Information Sciences 721 (2025) 122602

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

STEMLP: A spatial-temporal embedding multi-layer perceptron for traffic flow prediction

Liming Jiang, Baiyi Liu, Huanyu Wang *, Shaomiao Chen, Wei Liang

School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, 411100, Hunan, China

ARTICLE INFO

Keywords:
Traffic flow prediction
Spatial-temporal embedding
Multi-layer perceptron

ABSTRACT

The linear framework has become an alternative paradigm for the traffic flow prediction model with better predictive performance and less training cost, compared to Spatial-Temporal Graph Neural Networks. However, challenges such as insufficient spatial and temporal embeddings still persist. In this paper, we propose a novel spatial-temporal linear framework named STEMLP, to capture the complex spatial-temporal pattern by combining spatial-temporal adaptive embedding with Multi-Layer Perceptron. Specifically, the temporal periodicity patterns are treated as adaptive variables obtained by Discrete Fourier Transform, rather than determined values (daily, weekly) as in previous studies. Afterward, we construct spatial embedding based on the normalized Laplace Eigenvector Matrices of the predefined graph and adaptive graph to model spatial relationships between road nodes. Additionally, we use a mixed joint linear structure based on Multi-Layer Perceptron to integrate spatial and temporal embedding, achieving a better learned spatial-temporal patterns for traffic flow prediction. Experimental results on five real-world datasets show that STEMLP achieves improvements in all evaluation metrics, ranging from approximately 0.16% to 5.84%, compared to previous state-of-the-art models. In particular, when compared to Spatial-Temporal Graph Neural Network models, STEMLP is feasible to improve the performance by 0.92% to 9.76%, with a training time reduction from 3.19% to 60.30%.

1. Introduction

Rapid urbanization coupled with a rising number of motor vehicles places significant strain on urban traffic systems, intensifying challenges like congestion, accidents, and environmental pollution. Intelligent Transportation Systems (ITS) integrate advanced techniques in transportation to improve road safety, mitigate traffic congestion, and address mobility issues [1]. Effective traffic flow prediction constitutes an integral element in building ITS because it assists in traffic management and helps travelers plan for emergencies such as traffic accidents and unexpected celebrations, thus providing a smooth roadway [2]. However, achieving accurate and effective traffic flow prediction represents a pressing challenge, driven by the complex spatial-temporal dependencies inherent in traffic data.

Over the past decades, artificial intelligence has become a powerful ally for traffic prediction tasks, in which existing methods can be categorized into conventional techniques and advanced deep learning approaches. Conventional techniques are mainly based on statistical and machine learning models, including Auto-Regressive Integrated Moving Average (ARIMA) [3], struggle to handle high-

* Corresponding author.
E-mail addresses: liming@hnust.edu.cn (L. Jiang), 22010502012@mail.hnust.edu.cn (B. Liu), huanyu@hnust.edu.cn (H. Wang), csm1987@hnust.edu.cn (S. Chen), wliang@hnust.edu.cn (W. Liang).

https://doi.org/10.1016/j.ins.2025.122602
Received 1 April 2024; Received in revised form 22 July 2025; Accepted 13 August 2025

Available online 21 August 2025

dimensional spatial data and fail to capture intricate non-linear patterns in traffic datasets. Advances in deep learning, however, have provided solutions to these limitations. For instance, Zhao et al. [4] apply the Long Short Term Memory (LSTM) networks to capture temporal dependencies in traffic data for short-term prediction, though their approach does not account for spatial relationships. To address this, hybrid models such as the deep Spatial-Temporal Residual Network (ST-ResNet) [5] fuses Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to holistically model spatiotemporal dependencies, enhancing accuracy in traffic forecasting tasks. However, a problem raised by CNN models is that it is difficult to preserve translation invariance in non-Euclidean data structures, making them unsuitable for irregular road networks. When it comes to the case of irregular road structures, Graph Convolutional Networks (GCNs) have shown their priority in the prediction tasks for the traffic flow due to the capacity to deal with the graph-structured data. This advancement has spurred the development of high-performing Spatial-Temporal Graph Neural Networks (STGNNs). STGCN [6] utilizes stacked spatial-temporal convolutional blocks to capture dynamic correlations but relies on static predefined graphs for spatial modeling. In contrast, adaptive approaches like ST-RetNet [7] employ data-driven dynamic graph learning to minimize dependency on manually defined graph structures. Recent research has further refined STGNNs through innovations such as advanced graph convolution techniques [8], adaptive graph structure learning [9,10], efficient attention mechanisms [11,12], and other novel methodologies [13].

Although STGNNs have achieved great success among traffic flow prediction methods, it is important to note that certain issues with enhanced models predicated on STGNNs have been identified by the academic community [14,15]. On the one hand, model architectures grow substantially more complex, providing only relatively minor improvements in prediction performance. On the other hand, the computational demands of the model will also increase linearly or quadratically as the scale of datasets are becoming larger. This expansion leads to prolonged training times and increased memory consumption, making the model impractical for use in large-scale traffic network systems. Therefore, the methods based on Multi-Layer Perceptrons (MLPs) have been recently gained attention for their simplicity, low computational complexity, and superior prediction performance [16–20]. As an example, DLinear [16] employs a minimalistic architecture with just one linear layer to identify and extract the most significant periodic components from the temporal domain. More recently, the STHMLP framework [17] implements a more sophisticated frequency analysis approach using Discrete Fourier Transform (DFT) to detect multiple key frequencies rather than depending on a single periodic pattern, thereby enabling more effective modeling of long-range temporal relationships. When it comes to the case of spatiotemporal traffic data, existing works like STID [18], ST-MLP [19], NexuSQN [20], have investigated the impact of spatial embeddings and temporal periodicity embeddings on various prediction benchmarks. This class of methods improves the spatial-temporal contextualization representation through temporal and spatial embeddings. Consequently, they obtain certain advantages over STGNNs regarding operational efficiency and prediction accuracy. Despite the excellent performance of such methods, there are still some limitations in spatial-temporal embedding representation as follows.

First, in terms of temporal embedding, most methods, such as STID [18] and ST-MLP [19], directly use static time modes, such as daily and weekly periodicity, to construct temporal contextualization, ignoring the possible variability of periodicity patterns in different traffic datasets. Typically, temporal embedding should focus more on the apparent periodicity reflected by the data itself, helping the model to characterize the temporal contextualization more accurately.

Second, in terms of spatial embedding, STID [18] obtains embedding vectors of nodes by employing a self-learning matrix for adaptive graph information, excluding spatial relationships within predefined graph information. ST-MLP [19] further extends the spatial embedding representation of STID by dot-multiplication a self-learning matrix with the predefined adjacency matrix. Although this method considers two different spatial relationships for predefined graphs and adaptive graphs, the distinguish ability of the two spatial embeddings is affected because they share a self-learning matrix. Previous studies have emphasized that considering both adaptive graphs based on self-learning and predefined graphs can more completely encapsulate all the spatial complexity inherent in real scenes, which contribute significantly to enhancing the model's prediction accuracy.

To address the limitations of existing linear framework-based prediction methods mentioned above, this paper proposes a novel spatial-temporal embedding method. This method explicitly captures the heterogeneity in temporal periodic patterns while adapting to spatial complexity inherent in real scenes. Building on this embedding, we introduce STEMLP, a multi-layer perceptron network model specifically designed for traffic flow forecasting. The subsequent text summarizes the main contributions presented in this paper.

- We propose a novel temporal embedding method with periodical pattern perception by obtaining significant frequency components in traffic series data via DFT to effectively deal with the differentiated temporal periodicity under different datasets.
- We derive the normalized Laplace eigenvector matrix for the predefined graph and the adaptive graph to develop a spatial embedding approach. This method demonstrates an awareness of the static and adaptive spatial relationships, enabling it to adeptly handle the road networks adaptivity at different scales.
- We design an MLP-based mixed joint architecture which collaboratively integrates temporal and spatial embedding to facilitate its feature learning capacity. Experimental comparative analysis shows that the proposed MLP-based joint architecture outperforms the traditional simple connected and cascaded architecture.
- We perform comprehensive experiments evaluating the proposed model across five traffic datasets in comparison with nine baseline models. The outcomes demonstrate that our model consistently delivers superior predictive performance on all evaluated datasets, while also offering noteworthy benefits concerning model complexity and computational efficiency.

The main difference between STEMLP and the previous linear models is that we reconstructed the contextualized information of the traffic data itself to adaptive for the multi-periodicity of the time period pattern and the distinguishable between the predefined

and unknown relationships of the spatial dependence pattern. We hope that our work can inspire further efforts to adopt and expand the linear framework to more challenging urban spatial-temporal computing problems. The rest of this work consists of the following sections. Section 2 reviews the existing research methods for traffic flow prediction models. Section 3 presents the definitions of traffic flow prediction and the relevant theoretical background. Section 4 delineates the architecture and operational mechanisms of our newly developed STEMLP framework. Section 5 details the extensive evaluation of our model, including prediction performance experiments, efficiency analysis, ablation studies, hyper parameter tuning, and a range of visualizations. Section 6 discusses comparative results and model interpretations. Section 7 concludes with research contributions and future directions.

2. Related work

2.1. Spatial-temporal graph neural networks for traffic flow prediction

Advancements in artificial intelligence have driven a proliferation of traffic flow prediction methodologies over recent decades. Conventional stochastic methodologies such as ARIMA [3] and Kalman Filters (KFs) [21] can only extract linear relationships across time series. They cannot be applied directly to the processing of traffic data with complex nonlinear correlation. Given the inherent nonlinearity of high-dimensional traffic data, researchers employ machine learning frameworks including Random Forest Regression [22] to construct robust flow prediction models. However, these models fall short in capturing the complex spatial correlation present within road networks. In recent years, STGNNs [23,24] have attracted much attention for their ability to extract spatial-temporal correlation hidden in graph data by combining the spatial and temporal learning modules, which exhibit excellent prediction performance. The design elements of the existing STGNN-based traffic flow prediction solutions are summarized as seen in Fig. 1. STGNNs adopt a separate or coupled architecture to complete the interactive learning between temporal features and spatial features. In these STGNNs methods based on separated architecture, temporal processing occurs either before or after spatial processing, as exemplified by STGODE [25]. Conversely, some methods based on coupled architecture interleave spatial and temporal modules, as seen in DCRNN [26]. In terms of spatial learning, some existing works capture spatial correlation by modeling static predefined graph. However, the predefined graph may not be able to encompass the complete dynamic spatial complexity in real-world scenes [27]. Another research group developed adaptive graph convolution networks, such as AGCRN [28], GWNet [29], and MTGNN [30], which employ data-adaptive dynamic graph construction for spatial feature learning, thereby diminishing the model's dependency on predefined graphs. STMGCN [31] employs a multi-graph convolution network integrating neighborhood, functional similarity, and connectivity graphs. This approach enables the model to capture spatial relationships extending beyond topological neighborhoods. Various temporal learning architectures have been developed to profile time-dependent patterns in traffic flow data. Traditional solutions primarily employ recurrent network designs to capture and process inter-temporal dependencies in the time domain. A prominent example is DCRNN [26], which incorporates graph diffusion mechanisms within a GRU framework to handle both spatial and temporal relationships. Meanwhile, convolutional approaches have emerged as an effective alternative for temporal pattern extraction, forming the foundation of many current solutions. The STGCN model [6] demonstrates this through its innovative use of temporal gated convolution, where one-dimensional convolutional layers work in tandem with GLU activation to ensure stable model training. Recently, an increasing number of methodologies have turned to attention mechanisms, such as the Transformer's self-attention [32], to embed temporal correlations. Hybrid models are also widely explored for modeling inter-temporal dependencies in many practical applications, as seen in ASTGCN [23], which effectively merge attention mechanisms with convolutional operations for comprehensive temporal modeling. Subsequent researchers have combined some state-of-the-art techniques with STGNNs. For instance, the TFGAN framework [33] combines STGNNs with generative adversarial networks for traffic flow prediction, while STMetaNet [34] employed a meta-learner to extract additional spatial-temporal attributes, enhancing the model's predictive capabilities. However, most of existing models based on STGNNs often encounter difficulties due to the incorporation of excessively complex architectures and sophisticated techniques, which inadvertently constrain further enhancements in model performance. Besides, the computational costs of these models exhibit linear or quadratic growth relative to both the temporal sequence length and feature dimensionality, leading to substantial increases in processing time and memory demands. In contrast, the proposed STEMLP model demonstrates operational efficiency and predictive performance that can match or potentially outperform existing STGNN-based models.

2.2. Linear frameworks for multivariate time series prediction

With the continuous progress of the research on time series prediction, transformer-based time series prediction methods, such as Pyraformer [11], Autoformer [12], Fedformer [35], etc., have been widely used due to their ability to extract semantic associations between elements in a long sequence. However, transformers are characterized by their high computational cost and significant memory usage, particularly when processing long sequences. In contrast, linear models have gained renewed attention in applications such as computer vision [36] and attack detection [37]. For example, MLP-JCG introduced in [36] is an efficient multi-layer perceptron with joint-coordinate gating to leverage both local and global structural information for 3D pose estimation in vision tasks. Similarly, [37] proposed a deep neural network architecture based on a multi-head self-attention mechanism for the classification of network attacks. Inspired by these advancements, many researchers have recently developed linear prediction models for multivariate time series by combining data feature extraction and MLPs. We summarize the design elements of existing MLP-based multivariate time series prediction solutions, as seen in Fig. 2. Among them, Zeng et al. [16] decomposed the original data input into trend components and seasonal components, and then called Linear, DLinear, and NLinear, which series of linear models exhibits prediction

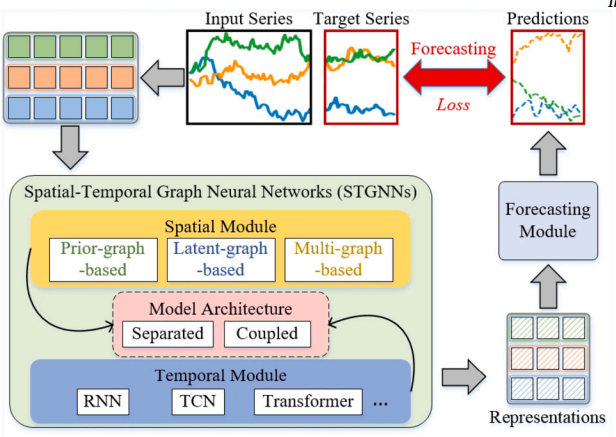


Fig. 1. Review of STGNN Methods for Traffic Flow Prediction.

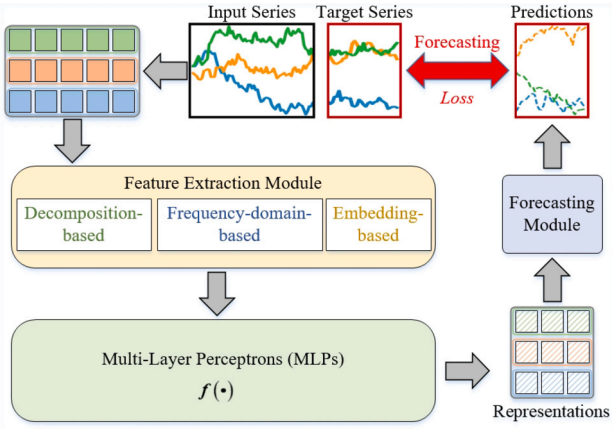


Fig. 2. Review of MLP-Based Methods for Multivariate Time Series Prediction.

performance beyond the advanced Transformer-based baseline models. Unlike the methods have been proposed for predicting in the time domain, FreTS [38] converts time-domain signals into complex-valued frequency representations. It employs redesigned MLPs to learn the real and imaginary components of frequency features at both inter-series and intra-series scales, enhancing channel-wise and temporal dependency modeling.

Traffic flow data represents a complex multivariate temporal sequence characterized by intricate spatial relationships. Current approaches face limitations in learning these spatial dependencies, consequently reducing their applicability for direct traffic prediction tasks. In spatial-temporal data analysis, researchers have proposed various simple and effective traffic flow prediction models by combining spatial-temporal embedding with the MLP structure. For example, the STID model [18] integrates spatial and temporal embeddings through a simple concatenation approach and leverages an MLP architecture. ST-MLP [19] incorporates spatial embedding from the predefined graph into existing methods, using an MLP cascade architecture to integrate spatial and temporal embeddings. In addition, STHMLP [17] employs a decomposition architecture that enables the model to learn the trend-cyclical and seasonal features of traffic time series, while utilizing a fine module and a coarse module to capture spatial-temporal correlations. NexuSQN [20] demonstrates a scalable MLP-Mixers in large-scale urban data prediction problems by customizing structured temporal and spatial mixing operations with learnable embedding. STLinear [39] advocates for a minimalist model architecture based on a fully localized strategy and a linear layer, designed to optimize efficiency and performance. However, these models do not adequately account for the variability in time periods and the dynamics of spatial relationships across different datasets when performing temporal and spatial embedding. In contrast, the proposed STEMLP model stands out by capturing multiple temporal periodicity information and more comprehensive spatial relationships from different traffic datasets. This is achieved through a novel mixed-joint MLP architecture specifically designed to merge temporal and spatial embeddings, thus improving the learning of spatial-temporal correlations.

3. Preliminaries

In this section, we introduce the fundamental terminology and formally define the traffic flow prediction problem. We present the method of DFT and process for extracting temporal periodic patterns. Table 1 lists some notations used in this paper.

Table 1
Notations.

Notation	Description
$X, \hat{X}$	Input data, Prediction result
E_d, E_t, E_s	Data embedding, Temporal embedding, Spatial embedding
E_{td}, E_{sd}	Temporal-data embedding, Spatial-data embedding
f_j	The significance frequency obtained by Discrete Fourier Transform
P_j	The corresponding time period of f_j
Γ_j	The temporal periodicity index of P_j
TP_j	Temporal periodicity information of traffic data X for time period P_j
E_j	The temporal embedding of time period P_j
B_j	The self-learning matrix for temporal embedding E_j
A_p, A_a	Adjacency matrix of predefined graph, Adjacency matrix of adaptive graph
B_{a_1}, B_{a_2}	Two different self-learning matrices for adaptive graph
E_p, E_a	Spatial embedding of predefined graph, Spatial embedding of adaptive graph

3.1. Problem formalization

Definition 1 *(Traffic network).* We define the traffic network as an undirected graph $G = (V, E, A)$, where $V = \{v_1, v_2, \ldots, v_N\}$ represents the set of N road nodes, E is the set of edges representing the road connections, and $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix that captures the static spatial relationships between the nodes.

Definition 2 *(Traffic flow feature).* The traffic flow feature (e.g. traffic speed or traffic flow) of road nodes v_i at time t is represented as $X_t^i \in \mathbb{R}^C$, where C represents the dimension of traffic flow feature. The traffic flow feature for all nodes at time t is expressed as $X_t = \left[X_t^1, X_t^2, \ldots, X_t^N\right] \in \mathbb{R}^{N\times C}$. The historical traffic flow for all nodes from time step $(t-T+1)$ to t is denoted as $X = \left[X_{(t-T+1)}, \cdots, X_t\right] \in \mathbb{R}^{N\times T\times C}$, where T is the number of past time steps considered. The predicted traffic flow for the next M time steps is denoted as $\hat{X} = \left[\hat{X}_{(t+1)}, \cdots, \hat{X}_{(t+M)}\right] \in \mathbb{R}^{N\times M\times C}$. In this paper, we set $C = 1$, resulting in $X \in \mathbb{R}^{N\times T}$ and $\hat{X} \in R^{N\times M}$.

Definition 3 *(Temporal periodicity index).* The significance frequency $f_j (1 \le j \le k)$ hidden in the traffic flow sequence data is extracted through the DFT [40], and its corresponding time period is P_j. We use $\tau(\cdot)$ to obtain the time index of each traffic flow feature $X_t^i \in \mathbb{R}^C$. Afterwards, we further derive the temporal periodicity index $\Gamma_{j,t}^i$ for X_t^i with P_j, as shown in Eq. (1).

$$\Gamma_{j,t}^i = \tau(t) \ mod \ \ P_j, \tag{1}$$

where $\tau(\cdot)$ represents the time index function and *mod* is the remainder operation. The normalization temporal periodicity index $\hat{\Gamma}_{j,t}^i$ is calculated as follows.

$$\hat{\Gamma}_{j,t}^i = \frac{\Gamma_{j,t}^i}{P_j}. \tag{2}$$

Definition 4 *(Temporal periodicity information).* The single-node temporal periodicity information of traffic flow feature $X^i = \left[X_{(t-T+1)}^i, \ldots, X_t^i\right] \in \mathbb{R}^{1\times T\times 1}$ for a specific time period P_j is defined as $TP_j^i = \left[\hat{\Gamma}_{j,(t-T+1)}^i, \ldots, \hat{\Gamma}_{j,t}^i\right] \in \mathbb{R}^{1\times T\times 1}$. Since the timestamps for each node are identical, TP_j^i is replicated N times to obtain the all-node temporal periodicity information TP_j, which is denoted as $TP_j \in \mathbb{R}^{N\times T\times 1}$. The complete temporal periodicity information is represented as $TP = \left[TP_1||TP_2||\ldots||TP_k\right] \in \mathbb{R}^{N\times T\times k}$.

Definition 5 *(Traffic flow prediction problem).* Traffic flow prediction aims to use the known historical traffic flow information X from time step $(t-T+1)$ to time step t, the traffic road network G, and the temporal periodicity information TP to learn a mapping function $f(\cdot)$ to predict future traffic flow information $\hat{X}$ for M time steps. Therefore, the traffic flow prediction problem can be formulated as follows:

$$\left[X_{(t-T+1)}, \cdots, X_t; G; TP\right] \overset{f(\cdot)}{\rightarrow} \left[\hat{X}_{(t+1)}, \cdots, \hat{X}_{(t+M)}\right]. \tag{3}$$

3.2. Temporal periodic pattern extraction using discrete Fourier transform

The DFT [40], an algorithm widely used in digital signal processing, converts time-varying signals from the time domain to the frequency domain to extract the magnitudes of their frequency components. Building on prior work [17], we apply the DFT to decompose traffic flow data, revealing periodic patterns hidden within the temporal sequence. The specific steps are as follows:

First, given a traffic series signal $x^i \in \mathbb{R}^L$ of road nodes v_i with length L, where $1 \le t \le L$, we apply the DFT to transform this time-domain signal $x^i \in \mathbb{R}^L$ into the frequency domain:

$$x_q^i = \sum_{t=1}^{L} x_t^i e^{-j(2\pi/L)qt}, \quad q = 1, 2, \ldots, L, \tag{4}$$

where $e^{-j(2\pi/L)qt}$ represents the complex exponential basis, and x_q^i indicates the spectrum of the signal x_t^i at the frequency $2\pi q/L$. Furthermore, x_q^i consists of L consecutive points, where $1 \le q \le L$.

Next, we calculate the magnitude AM_q^i of the q-th frequency component to obtain the frequency spectrum from x_q^i, as calculated as follows:

$$AM_q^i = \sqrt{(\mathrm{Re}(x_q^i))^2 + (\mathrm{Im}(x_q^i))^2}, \tag{5}$$

where $\mathrm{Re}(x_q^i)$ and $\mathrm{Im}(x_q^i)$ are the real and imaginary parts of x_q^i, respectively. The magnitude reflects the strength of the periodic signal at the corresponding frequency.

To extract global spectral patterns, we compute the average magnitude for each frequency component across all nodes. For N nodes, the global frequency magnitude AM_q at the q-th frequency is computed as follows:

$$AM_q = \frac{1}{N} \sum_{i=1}^{N} AM_q^i, \tag{6}$$

where the amplitude value AM_q of the q-th frequency component reflects the periodic signal strength of the corresponding frequency component.

Taking into account the sparsity within the frequency domain and mitigating the noise introduced by irrelevant high frequencies as noted in [35,41], we choose only the *top-k* amplitude values $\{AM_1, AM_2, \cdots, AM_k\}$ and calculate the corresponding frequency values $\{f_1, f_2, \cdots, f_k\}$, where k is the hyper parameter. Due to the conjugation of the frequency domain, we only take frequencies within $\{1, \ldots, \left\lceil \frac{L}{2} \right\rceil\}$ into consideration. The process can be represented as follows:

$$\{f_1, f_2, \ldots, f_k\} = \underset{f_* \in \{1, \ldots, \lfloor \frac{L}{2} \rfloor\}}{argTopk} (AM_k), \tag{7}$$

Finally, Eq. (8) is applied to these selected frequencies to obtain the corresponding time periodic lengths $\{P_1, P_2, \ldots, P_k\}$:

$$P_j = \left\lceil \frac{L}{f_j} \right\rceil, j \in \{1, 2, \ldots, k\}. \tag{8}$$

We visualize and analyze the periodicity of several public traffic datasets. Fig. 3 illustrates the spectrum of the PEMS04, PEMS07, PEMS08, and TFA datasets obtained using the described method. Notably, the top-5 significant frequencies and their corresponding time periods are identified and labeled for each dataset. Given the datasets' sampling interval of 5 minutes, a full 24 hours period corresponds to 288 sampling points. Consequently, the time period length of the daily periodicity mode is 288 (24 hours), and the time period length of other periodic modes can be interpreted similarly. From Fig. 3, several observations challenge existing research views. First, in most datasets, the weekly periodicity (7 days) does not appear. Second, all datasets exhibit not only the daily periodicity, but also shorter periodicity patterns, such as a time period length of 144 time slots (12 hours) and 96 time slots (8 hours). Third, even when different datasets share the same top-5 time period lengths, the amplitudes are different, indicating variations in the significance of the corresponding frequencies. Finally, we observe some drift in the time period lengths, which we attribute to missing data in the datasets.

4. Methodology

We introduce STEMLP, a novel traffic flow prediction model. As illustrated in Fig. 4, the model architecture integrates structured spatial and temporal embeddings to enhance feature representation, enabling more precise discrimination between spatial dependencies and temporal dynamics in traffic data. In summary, the proposed model includes three types of embeddings. 1) Temporal embedding $E_t \in \mathbb{R}^{N \times d_t}$ contains temporal periodicity information in the traffic data. 2) Spatial embedding $E_s \in \mathbb{R}^{N \times d_s}$ consists of spatial embedding of the predefined graph $E_p \in \mathbb{R}^{N \times d_p}$ and spatial embedding of adaptive graph $E_a \in \mathbb{R}^{N \times d_a}$, which expresses static spatial information and adaptive spatial relationships, respectively. 3) Data embedding $E_d \in \mathbb{R}^{N \times d_d}$ contains the original traffic data and the corresponding temporal periodicity information. We use a mixed joint MLP architecture to integrate spatial and temporal embeddings to achieve more effective spatial-temporal correlation learning. First, the joint embedding is obtained through the straightforward concatenation of two distinct types of embeddings. For example, the concatenation between the data embedding E_d and spatial embedding E_s can be effectively utilized to generate the joint spatial-data embedding E_{sd}. Similarly, the concatenation between data embedding E_d and temporal embedding E_t generates the joint temporal-data embedding E_{td}. Second, we use an MLP-based mixed joint architecture to learn the spatial-temporal correlation for the integrated temporal-data embedding E_{td} and spatial-data embedding E_{sd}. Finally, the prediction result $\hat{X}$ is obtained through the linear layer.

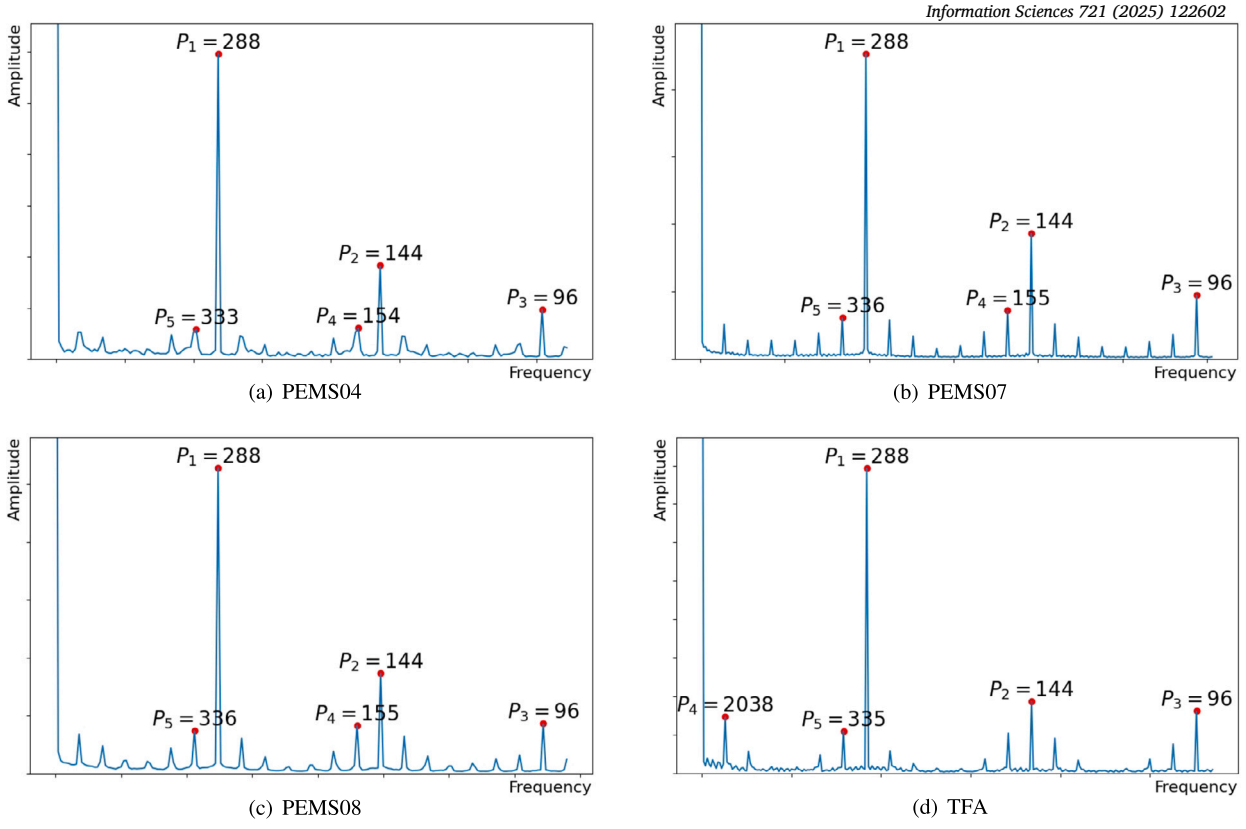


Fig. 3. Spectrograms of some traffic datasets.

4.1. Data embedding

Before obtaining the data embedding for traffic flow, we first extend it by incorporating temporal periodicity information. The historical traffic flow data of the node v_i in the past T time steps is expressed as $X^i \in \mathbb{R}^{1\times T\times 1}$, and the corresponding temporal periodicity information is $TP^i = \left[TP^i_1, TP^i_2, \ldots, TP^i_k\right] \in \mathbb{R}^{1\times T\times k}$. By concatenating X^i and TP^i together, we obtain the extension result $\left[X^i \parallel TP^i\right] \in \mathbb{R}^{1\times T\times(k+1)}$ of the traffic flow data. Regarding data embedding, we execute the $Flod(\cdot)$ operation on $\left[X^i \parallel TP^i\right]$ in the temporal dimension and get the data embedding $E^i_d \in \mathbb{R}^{1\times d_d}$ through the application of a linear layer, as shown in Eq. (9):

$$E^i_d = Linear\left(Flod\left(\left[X^i \parallel TP^i\right]\right)\right), \tag{9}$$

where $Flod(\cdot):\mathbb{R}^{1\times T\times(k+1)} \rightarrow \mathbb{R}^{1\times T(k+1)}$ denotes the flattening operation.

The traffic flow data for all nodes is processed according to Eq. (9) to derive the corresponding data embeddings. These embeddings are then combined to form the final data embedding $E_d \in \mathbb{R}^{N\times d_d}$ for the road network traffic flow X. Notice that all linear projection operations in this paper are applied in the temporal dimension. Specifically, all the nodes of the road network use the Channel-Independent (CI) strategy to obtain its respective embedding representation, thereby mitigating the distribution offset problem in the traffic datasets [19,42].

4.2. Temporal embedding

As shown in Fig. 4(b), we combine the temporal periodicity index sequence with the self-learning matrix to obtain the embedding vector. This vector is then replicated N times and concatenated across multiple temporal periods to form the final temporal embedding. Following Eq. (1), we generate the periodicity index sequence $\left[\Gamma^i_{j,(t-T+1)} : \Gamma^i_{j,t}\right]$ for traffic flow data $X^i = \left[X^i_{(t-T+1)}, \ldots, X^i_t\right] \in \mathbb{R}^{1\times T\times 1}$ within period P_j. This sequence forms the basis for temporal embedding construction. Second, we execute the lookup operation on matrix B_j using the sequence $\left[\Gamma^i_{j,(t-T+1)} : \Gamma^i_{j,t}\right]$ and obtain the embedding vector $E^i_j \in \mathbb{R}^{1\times d_j}$ through the application of a linear layer, as shown in Eq. (10):

$$E^i_j = Linear\left(Lookup\left(B_j, \left[\Gamma^i_{j,(t-T+1)} : \Gamma^i_{j,t}\right]\right)\right), \tag{10}$$

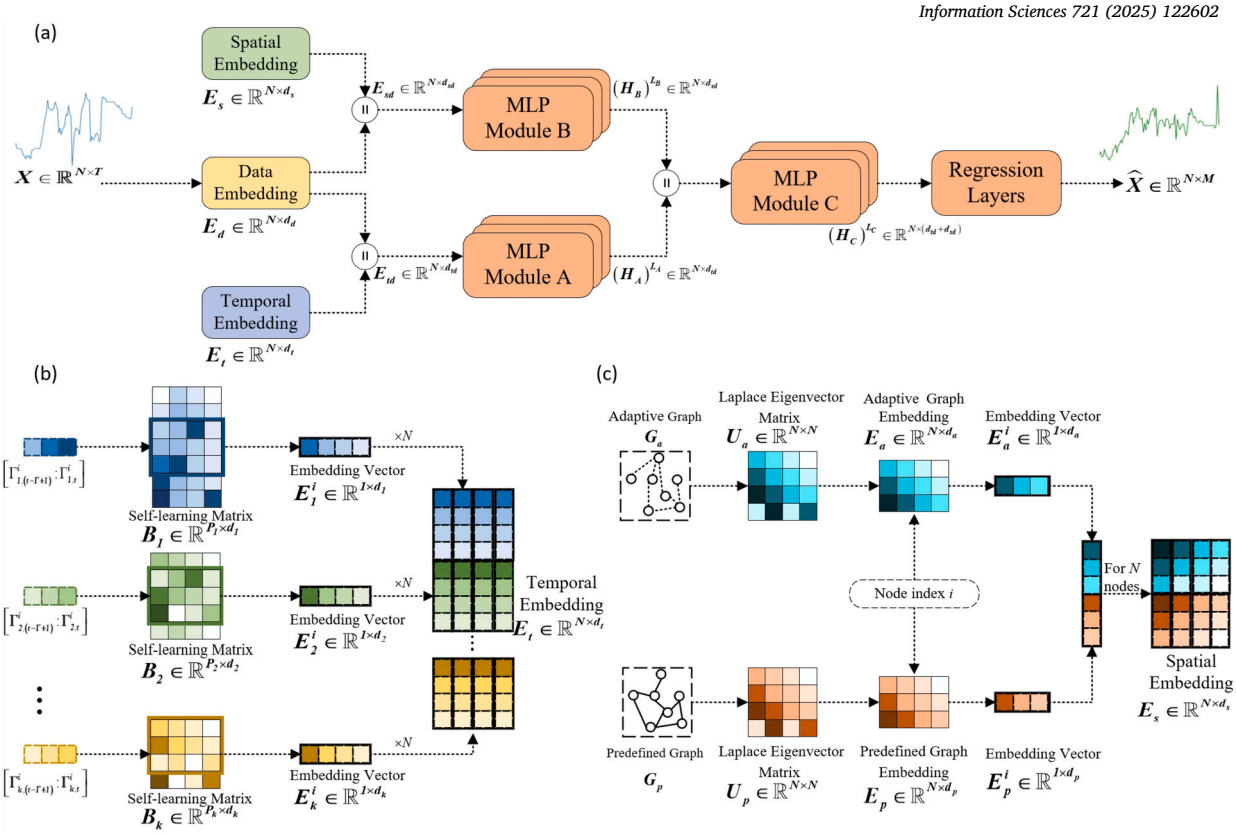


Fig. 4. (a) Structure of STEMLP. (b) Details of temporal embedding, which contains temporal periodicity information. (c) Details of spatial embedding, which include spatial embedding of predefined graph and spatial embedding of adaptive graph.

where $B_j \in \mathbb{R}^{P_j \times d_j}$ is a self-learning matrix, and the $Lookup(\cdot,\cdot) : \mathbb{R}^{P_j \times d_j} \to \mathbb{R}^{T \times d_j}$ is a function to query from matrix B_j with index sequence $\left[\Gamma^i_{j,(t-T+1)} : \Gamma^i_{j,t}\right]$.

Since all nodes have the same temporal periodicity information, we eventually replicate E^i_j N times to derive the embedding vector $E_j \in \mathbb{R}^{N \times d_j}$ of the traffic flow data X in time period P_j. Using the same operation, we can obtain the temporal embedding vector $\{E_1, E_2, \ldots, E_k\}$ of the traffic flow data X over time periods $\{P_1, P_2, \ldots, P_k\}$. Finally, all of them are concatenated together to generate the final temporal embedding E_t as shown in Eq. (11):

$$E_t = Concat\left[\left(E_1, E_2, \ldots, E_k\right), dim = 1\right], \tag{11}$$

where $E_t \in \mathbb{R}^{N \times d_t}$, $d_t = d_1 + d_2 + \cdots + d_k$.

4.3. Spatial embedding

As shown in Fig. 4(c), we can obtain graph Laplace eigenvectors for the predefined graph and adaptive graph to generate the corresponding spatial embedding. The purpose of spatial graph embedding is to project the nodes of the graph into a low-dimensional eigenvector space while preserving the strength of the connection relationships between them. For example, consider two node pairs (v_i, v_m) and (v_i, v_n), which are characterized by connection strengths S_{im} and S_{in}, such that $S_{im} > S_{in}$. In this case, node v_m will be projected closer to v_i in the embedding space compared to the projection of v_n. Previous studies have pointed out that the graph Laplacian eigenvectors are able to embed the graph into Euclidean space and retain the global graph structure information, which can be regarded as an optimal set of embedding representations of the graph [43,44]. Based on this, we leverage these eigenvectors for representation learning. Specifically, the first step is to compute the normalized Laplacian matrix, as defined in Eq. (12):

$$\widehat{\mathcal{L}} = D^{-1/2}\mathcal{L}D^{-1/2} = I - D^{-1/2}AD^{-1/2}, \tag{12}$$

where $D \in \mathbb{R}^{N \times N}$ is the degree matrix, $\mathcal{L} = D - A$ is the standard Laplace matrix, $I \in \mathbb{R}^{N \times N}$ is the unit matrix, $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix.

Next, through eigenvalue decomposition, we derive the eigenvalue matrix Λ and the eigenvector matrix U, as shown in Eq. (13):

$$\widehat{\mathcal{L}} = U\Lambda U^T, \tag{13}$$

As shown in Fig. 4(c), we can further obtain graph Laplace eigenvectors for the predefined graph G_p and the adaptive graph G_a to generate the corresponding spatial embedding E_p and E_a. These embeddings enhance the expressive power of the spatial embedding by capturing both the topological information of the traffic road network and the adaptive spatial relationships between the road nodes.

4.3.1. Spatial embedding of predefined graph

In existing works on traffic flow prediction, the adjacency matrix A_p of the predefined graph G_p can be calculated by using the distance function or similarity function [27]. In the following, we introduce the implementation of spatial embedding for predefined graph structures using the eigenvalue decomposition of the normalized Laplacian matrix and the smallest m non-trivial eigenvectors respectively.

First, we compute the Laplacian matrix $\mathcal{L}$ from the adjacency matrix A_p and the degree matrix D_p. The normalized Laplacian matrix $\widehat{\mathcal{L}}_p \in \mathbb{R}^{N\times N}$ is then obtained via symmetric normalization, as defined in Eq. (14):

$$\widehat{\mathcal{L}}_p = I - D_p^{-1/2} A_p D_p^{-1/2}, \tag{14}$$

Next, by conducting eigenvalue decomposition of the normalized Laplace matrix $\widehat{\mathcal{L}}_p$, we can get the eigenvectors matrix $U_p \in \mathbb{R}^{N\times N}$ as shown in Eq. (15):

$$\widehat{\mathcal{L}}_p = U_p \Lambda_p U_p^T. \tag{15}$$

Afterwards, we use the smallest m ($m = d_p$) non-trivial eigenvectors of the Laplace matrix as the spatial embedding $E_p \in \mathbb{R}^{N\times d_p}$ for all nodes on the graph. Eventually, we assign spatial embedding of the predefined graph $E_p^i \in \mathbb{R}^{1\times d_p}$ to the node v_i based on the node index i.

4.3.2. Spatial embedding of adaptive graph

Although the predefined graph G_p can encapsulate the static topology or functional connectivity relationships of traffic road networks, it has limitations in fully uncovering the hidden spatial details of traffic data in real-world scenarios [24,45]. Therefore, we introduce the Laplace matrix $\widehat{\mathcal{L}}_a$ of the adaptive graph G_a to model hidden spatial dependencies between nodes, as established in [27,28]. The construction of $\widehat{\mathcal{L}}_a$ involves the following step:

First, a symmetric adaptive normalized matrix $\widehat{A}_a = D_a^{-1/2} A_a D_a^{-1/2} \in \mathbb{R}^{N\times N}$ is computed. This matrix $\widehat{A}_a$ is obtained by multiplying two learnable matrices B_{a_1} and $B_{a_2}^T$ (where $B_{a_1}, B_{a_2} \in \mathbb{R}^{N\times d_b}$), as shown in Eq. (16):

$$D_a^{-1/2} A_a D_a^{-1/2} = Softmax(ReLU(B_{a_1} \cdot B_{a_2}^T)), \tag{16}$$

where $Softmax(\cdot)$ and $ReLU(\cdot)$, are used for normalization and negative removal of the adaptive adjacency matrix A_a, respectively. Instead of generating the matrix A_a and calculating the corresponding normalization matrix, we directly generate $D_a^{-1/2} A_a D_a^{-1/2}$ by performing nonlinear projection and normalization operations on A_a, reducing computational cost in the training process. On this basis, we can get the normalized Laplace matrix $\widehat{\mathcal{L}}_a \in \mathbb{R}^{N\times N}$ of the adaptive graph G_a, as shown in Eq. (17):

$$\widehat{\mathcal{L}}_a = I - \widehat{A}_a = I - D_a^{-1/2} A_a D_a^{-1/2}, \tag{17}$$

Next, by conducting eigenvalue decomposition of the normalized Laplace matrix $\widehat{\mathcal{L}}_a$, we can get the eigenvector matrix $U_a \in \mathbb{R}^{N\times N}$ as shown in Eq. (18):

$$\widehat{\mathcal{L}}_a = U_a \Lambda_a U_a^T, \tag{18}$$

Similarly to the spatial embedding of the predefined graph, we use the smallest m ($m = d_a$) non-trivial eigenvectors of the Laplace matrix as the spatial embedding $E_a \in \mathbb{R}^{N\times d_a}$ for all nodes on the graph. Subsequently, the spatial embedding $E_a^i \in \mathbb{R}^{1\times d_a}$ is assigned to node v_i by directly indexing its position in E_a. Specifically, for node v_i ($0 < i < N$), its spatial embedding $E_a^i \in \mathbb{R}^{1\times d_a}$ is assigned by selecting the i-th row vector of E_a.

4.3.3. Spatial embedding representation

Independent derivation of spatial embeddings E_p^i and E_a^i is shown in Fig. 4(c). The overall spatial embedding of node v_i is obtained by concatenating E_p^i and E_a^i as shown in Eq. (19):

$$E_s^i = Concat[(E_p^i, E_a^i), dim = 1]. \tag{19}$$

The process is eventually repeated for all N nodes to obtain the final spatial embedding $E_s \in \mathbb{R}^{N\times d_s}$, where $d_s = d_p + d_a$.

4.4. MLP based mixed joint architecture

4.4.1. MLP module

In Fig. 4(a), we design three MLP modules consisting of stacked MLP layers containing residual connections, denoted M_A, M_B, M_C, to encode the information. The number of MLP layers in the MLP module M_A, M_B, M_C are L_A, L_B, L_C, which are adjustable hyper parameters. Taking the MLP module M_A as an example, the operation of the $(l+1)$ - $th (0 \leq l < L_A)$ MLP layer can be expressed as:

$$(H_A)^{l+1} = Linear_2^l\left(\sigma\left(Linear_1^l\left((H_A)^l\right)\right)\right) + (H_A)^l, \tag{20}$$

where $(H_A)^l \in \mathbb{R}^{N \times d_{td}}$ is the feature of the l-th MLP layer of module M_A in the model and $\sigma(\cdot)$ is the $ReLU$ activation function. Note that when $l = 0$, $(H_A)^l = E_{td}$.

4.4.2. Mixed joint architecture

The MLP based mixed joint architecture is shown in Fig. 4(a). Firstly, we fuse data embedding separately with temporal and spatial embeddings to produce temporal-data embedding E_{td} and spatial-data embedding E_{sd}, as shown in Eq. (21) and Eq. (22):

$$E_{td} = Concat[(E_t, E_d), dim = 1], \tag{21}$$

$$E_{sd} = Concat[(E_s, E_d), dim = 1], \tag{22}$$

where $E_{td} \in \mathbb{R}^{N \times d_{td}}$, $d_{td} = d_t + d_d$, $E_{sd} \in \mathbb{R}^{N \times d_{sd}}$, $d_{sd} = d_s + d_d$.

Afterwards, we employ a mixed joint architecture to learn the temporal-related feature $(H_A)^{L_A}$, spatial-related features $(H_B)^{L_B}$, and spatial-temporal feature $(H_C)^{L_C}$. The feature learning process for all MLP modules can be expressed as follows.

$$(H_A)^{L_A} = MLP_A(E_{td}), \tag{23}$$

$$(H_B)^{L_B} = MLP_B(E_{sd}), \tag{24}$$

$$(H_C)^{L_C} = MLP_C([(H_A)^{L_A} \| (H_B)^{L_B}]), \tag{25}$$

where $(H_A)^{L_A} \in \mathbb{R}^{N \times d_{td}}$, $(H_B)^{L_B} \in \mathbb{R}^{N \times d_{sd}}$, $(H_C)^{L_C} \in \mathbb{R}^{N \times (d_{td}+d_{sd})}$, $MLP_A(\cdot), MLP_B(\cdot), MLP_C(\cdot)$ are MLP module functions.

Ultimately, the regression layer processes spatial-temporal feature $(H_C)^{L_C}$ to generate the final prediction output $\hat{X} \in \mathbb{R}^{N \times M}$, as shown in Eq. (26):

$$\hat{X} = Linear((H_C)^{L_C}). \tag{26}$$

4.5. Loss function

In this paper, the proposed model is trained by minimizing the mean absolute error loss between the true value X and the predicted value $\hat{X}$, as shown in Eq. (27):

$$L(\hat{X}, X) = \frac{1}{NM}\sum_{i=1}^{N}\sum_{t=1}^{M}\left|X_t^i - \hat{X}_t^i\right|, \tag{27}$$

where X_t^i is the true value of node v_i in the next t-th time step, $\hat{X}_t^i$ is the predicted value of node v_i in the next t-th time step, M is the predicted step size and N is the number of nodes.

5. Experiments

In order to assess the accuracy and validity of our model, we examine the following questions (RQs):

***RQ1.** What improvements in Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) does the proposed method achieve compared to the benchmark?

***RQ2.** What computational cost reduction per training epoch does it offer over the STGNNs baseline?

***RQ3.** What is the significance of spatial-temporal embedding components and the mixed joint architecture for model performance as measured by MAE?

***RQ4.** How do certain parameters impact model performance as evaluated through MAE, RMSE, and MAPE?

***RQ5.** Are spatial-temporal embedding learning results interpretable?

5.1. Datasets

Our evaluation utilizes five real-world large-scale traffic flow datasets, called PEMS04, PEMS07, PEMS08, PEMS-BAY, and TFA. The first three (PEMS04, PEMS07, and PEMS08) consist of high-resolution freeway traffic measurements obtained by the Caltrans Performance Measurement System, covering major metropolitan freeways in central California. The PEMS-BAY dataset incorporates six months of continuous traffic records gathered from 325 fixed sensor stations distributed throughout the San Francisco Bay Area's

Table 2
Description of the datasets used in our experiments.

Data Source	Nodes	Sample Rate	Time Steps	Time Stamp
PEMS04	307	5 minutes	16992	01/01/2018-02/28/2018
PEMS07	883	5 minutes	28224	05/01/2017-08/31/2017
PEMS08	170	5 minutes	17856	07/01/2016-08/31/2016
PEMS-BAY	325	5 minutes	52116	01/01/2017-05/31/2017
TFA	179	5 minutes	26496	07/01/2022-09/31/2022

southern region. The TFA dataset, collected from loop sensors on the Twin Cities (Minneapolis and St. Paul) highway network, is provided by the Regional Traffic Management Center (RTMC), a division of the Minnesota Department of Transportation (MnDOT). These datasets include traffic flow, speed, and density information. For PEMS04, PEMS07, PEMS08 and TFA, we select time-stamped traffic flow as the traffic flow feature of the nodes in the road network. For PEMS-BAY, we select time-stamped traffic speed. Additionally, we aggregate the traffic flow on each road every 5 minutes. Detailed statistical information on node size, time span, and data sampling rate in the four traffic flow datasets is shown in Table 2.

In the experiments, we used the Z-score normalization method to preprocess the data. For the PEMS04, PEMS07, PEMS08 and TFA datasets, we divided them into training, validation, and testing sets in a 6:2:2 ratio, while the PEMS-BAY dataset was divided in a 7:1:2 ratio.

5.2. Baseline model and metrics

We evaluate the proposed STEMLP model by comparing it with the following methods for traffic prediction. First, GWNet [29], STGCN [6] and DGCRN [46] are prior-graph-based STGNNs, which require pre-defined graphs to indicate spatial dependencies among time series. Second, StemGNN [8], GTS [9] and MTGNN [30] are latent-graph-based STGNNs, which propose to jointly learn graph structures and optimize STGNNs. Besides, we also adopt three up-to-date MLP methods, STID [18], TSMixer [47] and ST-MLP [19], for the comparison. In summary, the proposed model is compared against a set of representative baseline methods.

- STGCN [6]: it captures global spatial-temporal correlations by combining graph convolution and gated temporal convolution.
- DGCRN [46]: it uses dynamic graph convolution to learn spatial-temporal features, and generates multi-step traffic flow prediction by sequence generation.
- StemGNN [8]: it captures inter-sequence correlation and time dependence in the spectral domain.
- GTS [9]: it is an improvement on DCRNN by replacing static predefined graph with a probability graph.
- MTGNN [30]: it combines the stacking and jumping connection of spatial-temporal convolutional blocks to realize the learning of multi-scale spatial-temporal correlation.
- GWNet [29]: it combines graph convolution and expansion causal convolution to capture spatial-temporal correlation.
- STID [18]: it integrates temporal embedding and spatial embedding by a simple connected architecture and achieves spatial-temporal correlation learning through MLP.
- TSMixer [47]: it uses a fully-connected MLP architecture with time and feature mixing mechanisms to capture temporal and feature correlations in time series data.
- ST-MLP [19]: it integrates temporal and spatial embedding through a cascading MLP architecture and learns the spatial-temporal correlation of traffic flow data.

Performance evaluation is based on MAE, RMSE, and MAPE, defined as follows:

$$MAE = \frac{1}{NM}\sum_{i=1}^{N}\sum_{t=1}^{M}|X_t^i - \hat{X}_t^i|, \tag{28}$$

$$MAPE = \frac{1}{NM}\sum_{i=1}^{N}\sum_{t=1}^{M}\left|\frac{X_t^i - \hat{X}_t^i}{X_t^i}\right|, \tag{29}$$

$$RMSE = \sqrt{\frac{1}{NM}\sum_{i=1}^{N}\sum_{t=1}^{M}\left(X_t^i - \hat{X}_t^i\right)^2}, \tag{30}$$

where X_t^i is the true value of node v_i in the next t-th time step, $\hat{X}_t^i$ is the predicted value of the node v_i in the next t-th time step, M is the predicted step size and N is the number of nodes.

5.3. Experimental setup

In this paper, the STEMLP model and baseline models are implemented using BasicTS [48] and Easytorch [49]. All experiments are conducted on a computing server equipped with an NVIDIA RTX 3090 graphics card, an AMD EPYC 7502 32-Core Processor

Table 3
Parameter settings for different datasets.

Parameter	Description	Value				
		PEMS04	PEMS07	PEMS08	PEMS-BAY	TFA
d_p	Hidden dimension of spatial embedding of predefined graph E_p	64	96	32	64	32
d_a	Hidden dimensions of spatial embedding of adaptive graph E_a	64	96	32	64	32
d_b	Hidden dimensions of self-learning matrix B_{a_1}, B_{a_2}	16	32	32	16	8

Table 4
Comparative performance evaluation on the PEMS04, PEMS07, PEMS08, PEMS-BAY and TFA datasets.

Data	Metrics	STGNNs						MLPs				Promotion	
		STGCN	DGCRN	StemGNN	GTS	MTGNN	GWNet	STID	TSMixer	ST-MLP	STEMLP	Best baseline	Best STGNNs baseline
PEMS04	MAE	19.76	18.84	22.98	21.23	19.13	18.80	18.44	19.81	18.15	**18.10**	+0.28%	+3.72%
	RMSE	31.51	30.48	36.00	33.55	31.03	30.14	30.05	31.68	29.82	**29.74**	+0.27%	+1.32%
	MAPE	13.45%	12.92%	16.56%	14.85%	13.22%	13.19%	12.47%	13.40%	12.28%	**12.26%**	+0.16%	+7.05%
PEMS07	MAE	22.25	20.04	22.50	22.47	21.01	20.44	19.59	21.81	19.51	**18.89**	+3.18%	+7.58%
	RMSE	35.83	32.86	36.41	35.42	34.14	33.38	32.77	34.58	32.61	**32.22**	+1.20%	+3.48%
	MAPE	9.47%	8.63%	9.57%	9.56%	8.92%	8.71%	8.30%	8.98%	8.26%	**7.86%**	+4.84%	+9.76%
PEMS08	MAE	16.19	14.77	16.90	16.92	15.25	14.67	14.23	17.00	14.03	**13.32**	+5.06%	+9.20%
	RMSE	25.51	23.81	26.30	26.68	24.22	23.55	23.48	26.43	23.07	**22.75**	+1.39%	+3.40%
	MAPE	10.82%	9.77%	11.89%	10.88%	10.66%	9.46%	9.40%	12.37%	9.25%	**8.71%**	+5.84%	+7.93%
PEMS-BAY	MAE	1.63	1.58	1.99	1.68	1.60	1.59	1.56	1.73	1.56	**1.55**	+0.64%	+2.52%
	RMSE	3.72	3.65	4.49	3.79	3.71	3.68	3.60	3.86	3.55	**3.48**	+1.97%	+5.43%
	MAPE	3.69%	3.52%	4.61%	3.68%	3.59%	3.60%	3.51%	3.91%	3.50%	**3.49%**	+0.29%	+3.06%
TFA	MAE	13.47	13.17	15.31	13.71	13.23	13.05	13.01	13.79	13.11	**12.93**	+0.62%	+0.92%
	RMSE	20.02	19.73	23.18	20.43	19.70	19.72	19.63	20.65	19.35	**19.41**	+1.12%	+1.60%
	MAPE	16.54%	16.32%	17.02%	16.71%	18.82%	15.97%	15.92%	16.83%	16.32%	**15.47%**	+2.83%	+3.13%

Note: **Bold text** denotes the best performance, underlined text indicates the second-best, and green text represents the optimal performance among STGNNs. 'Promotion' refers to the improvement margin of STEMLP over the best baseline model or the best STGNNs baseline model. (For interpretation of the colors in the table, the reader is referred to the web version of this article.)

operating at 2.50 GHz, and 256 GB of RAM. Adam is selected as the optimizer for training, with the initial learning rate set to 0.002 and milestones configured as $[1, 50, 80, 100, 150]$. The training process is stopped after 200 epochs and the Batchsize is set to 32. We used MAE as the training loss function and selected hyperparameters based on MAE, RMSE, and MAPE. Specifically, the number of layers of three MLP modules M_A, M_B, M_C are set to $L_A = 3$, $L_B = 3$ and $L_C = 1$, respectively. The number of significant time periods k is set to 3. The hidden dimension d_d of the data embedding E_d is set to 96, and the hidden dimension d_1, d_2, d_3 of the temporal embedding E_1, E_2, E_3 for all three time periods is set to 32. The settings of the other hyper parameters are shown in Table 3. The code is publicly available at https://github.com/liu681/STEMLP.

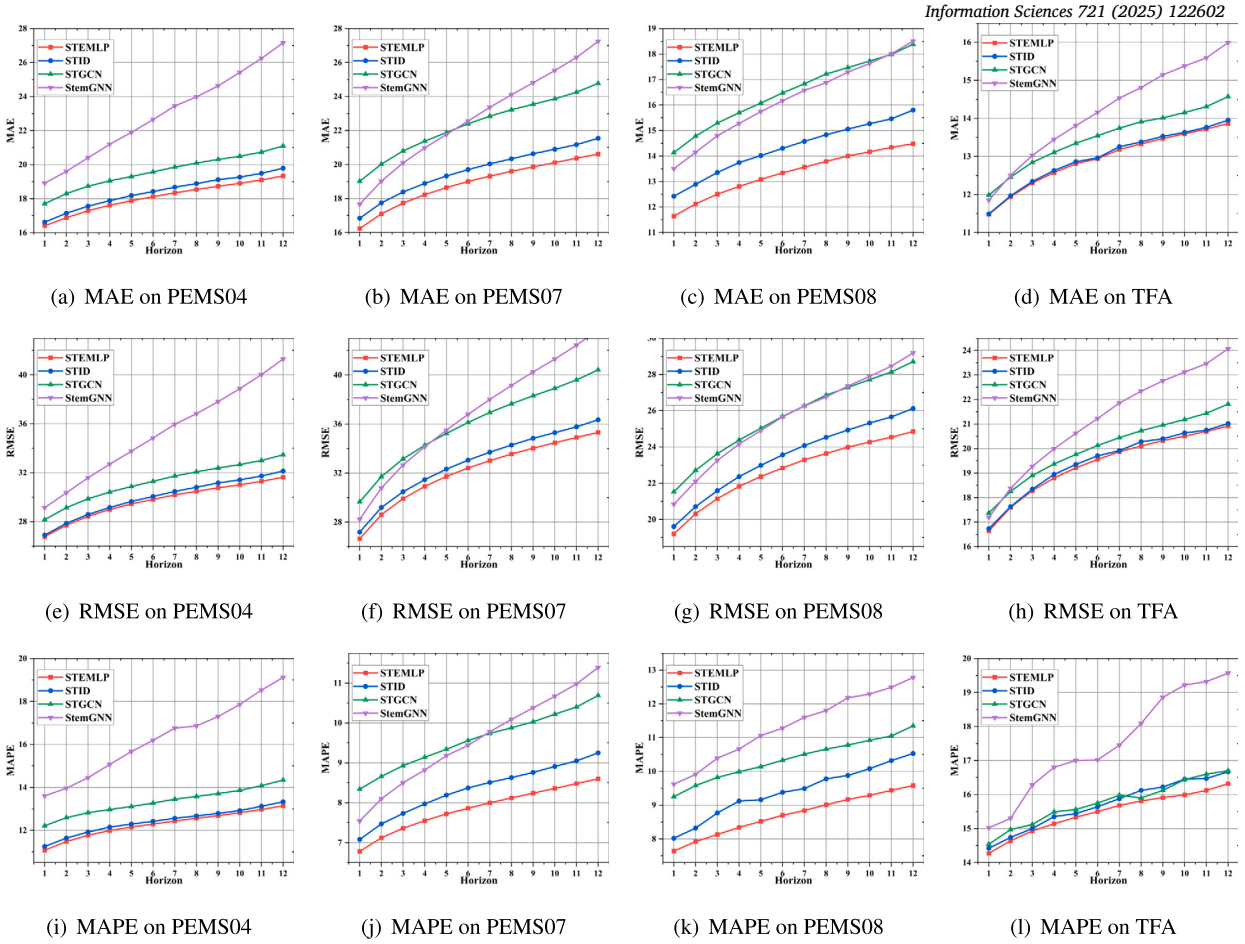
5.4. Comparison of predictive performance with the baseline model (RQ1)

This section presents a comparative evaluation of our STEMLP model against baseline models using five traffic flow datasets. Table 4 presents the average MAE, RMSE, and MAPE results of the models in the 60-minute traffic flow prediction task.

According to Table 4, the STEMLP model outperforms all other models, achieving an average improvement of approximately 2% across all evaluation metrics when compared to the best-performing baseline. STGCN, GTS, DGCRN, and other STGNN-based models benefit from the ability to learn more complex spatial-temporal correlation features in traffic flow sequence data by utilizing graph neural networks combined with temporal learning networks, which improves prediction performance. Prior-graph-based methods generally perform better than latent-graph-based or non-graph-based approaches, owing to the incorporation of prior knowledge. Additionally, learning the graph structure remains a challenging task. Among the methods, only MTGNN successfully learns effective graph structures, although this does not significantly improve prediction performance. In general, more intricate network structures offer limited improvements. In contrast, deep learning models based on linear frameworks, such as STID and ST-MLP, achieve better prediction accuracy than STGNN-based models for traffic flow prediction tasks by integrating effective temporal and spatial embeddings. In particular, the proposed STEMLP model achieves lower prediction errors than STID and ST-MLP.

We set a 5-minute prediction time step and gradually increased the prediction horizon from 1 to 12 (i.e., from 5 minutes to 1 hour). The experimental results are presented in Fig. 5. As the prediction time step increases, the uncertainty and complexity in traffic flow prediction also rise, leading to an overall increase in prediction error. Our STEMLP model achieves superior prediction results in short-term prediction alone, demonstrating that the strategy of spatial-temporal embedding combined with the MLP-based mixed joint architecture can thoroughly learn the spatial-temporal correlation of the traffic flow data. As the time step increases, our STEMLP model exhibits a comparatively slower growth in prediction error, compared to alternative methods. This is due to the model's capacity to explicitly capture multiple time-period information within its temporal embedding, giving it a significant advantage in medium- and long-term prediction scenarios.

To comprehensively evaluate the temporal prediction capabilities of STEMLP at different time steps, we visualize the prediction results of STEMLP and baseline models (STID, StemGNN, and STGCN) across 15-, 30-, and 60-minute horizons during a representative workday. Fig. 6 shows the prediction results of different models for sensor #149 of the PEMS08 dataset on August 30, 2016. We can



(a) MAE on PEMS04 (b) MAE on PEMS07 (c) MAE on PEMS08 (d) MAE on TFA

(e) RMSE on PEMS04 (f) RMSE on PEMS07 (g) RMSE on PEMS08 (h) RMSE on TFA

(i) MAPE on PEMS04 (j) MAPE on PEMS07 (k) MAPE on PEMS08 (l) MAPE on TFA

Fig. 5. Prediction performance at each time step for different on the PEMS04, PEMS07, PEMS08, and TFA datasets.

find that the STEMLP model outperforms spatial-temporal graph neural network-based models like StemGNN and STGCN, as well as the deep network model based on linear framework, STID, for prediction horizons of 15 minutes, 30 minutes, and 60 minutes. There is also the case that the performance of the STEMLP model prediction remains optimal in both the peak and the trough time ranges.

5.5. Efficiency study compared to STGNNs methods (RQ2)

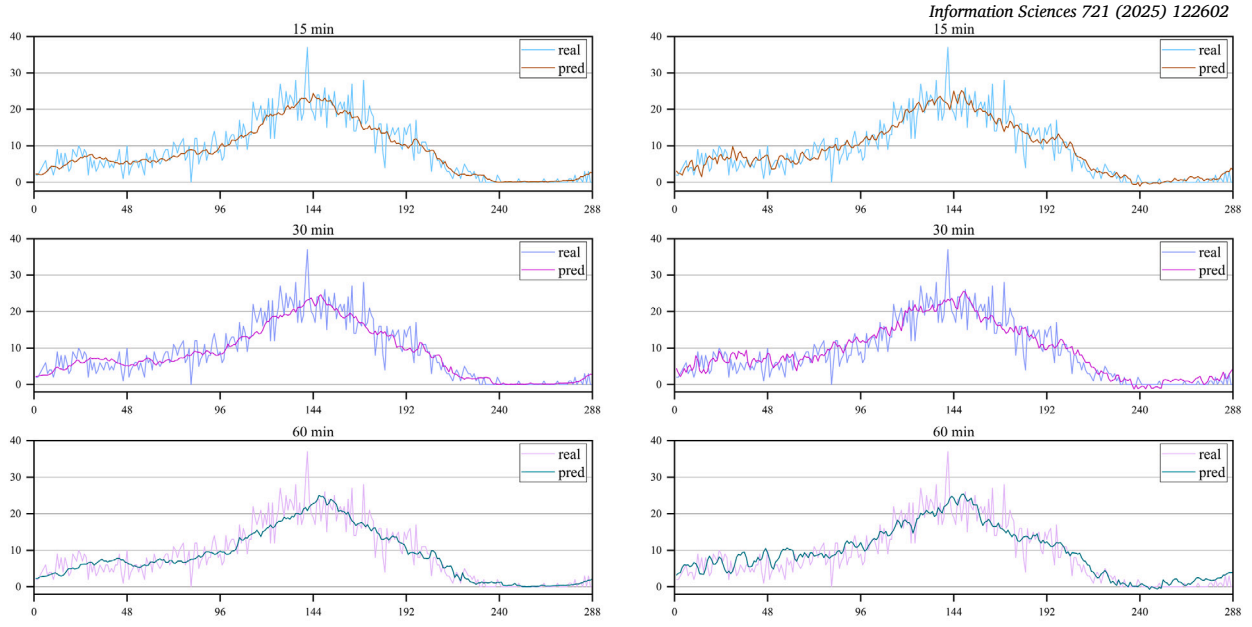
Given STEMLP's superior prediction performance compared to STGNN-based approaches, we conduct a further analysis on the complexity and average training time for each epoch required by the proposed model and other baseline models, including GWNet, STGCN, StemGNN, GTS, MTGNN, and DGCRN. The computational efficiency comparison is presented in Table 5. To ensure consistency in the experiments, the time overheads of all models on the four datasets are measured under identical experimental conditions.

As shown in Table 5, compared to the baseline model based on STGNNs, the proposed STEMLP model achieves an average reduction of approximately 20.45% in training time per epoch, attributed to its streamlined architecture that incorporates MLP and linear layers. GTS and DGCRN are based on the sequence generation approach to obtain multi-step prediction results, which requires more training time due to its inherent recurrent structure, and therefore, DGCRN has the time overhead among all the models as the largest. At the same time, we can also see that the training time overhead increases linearly for graph neural network models that stack multiple spatial-temporal graph convolution modules, such as GTS, GWNet, and DGCRN models, in larger datasets (such as PEMS07). In general, the STEMLP model maintains a minimum training time overhead under datasets with different node sizes. Therefore, the efficiency study of the model shows the advantages of STEMLP.

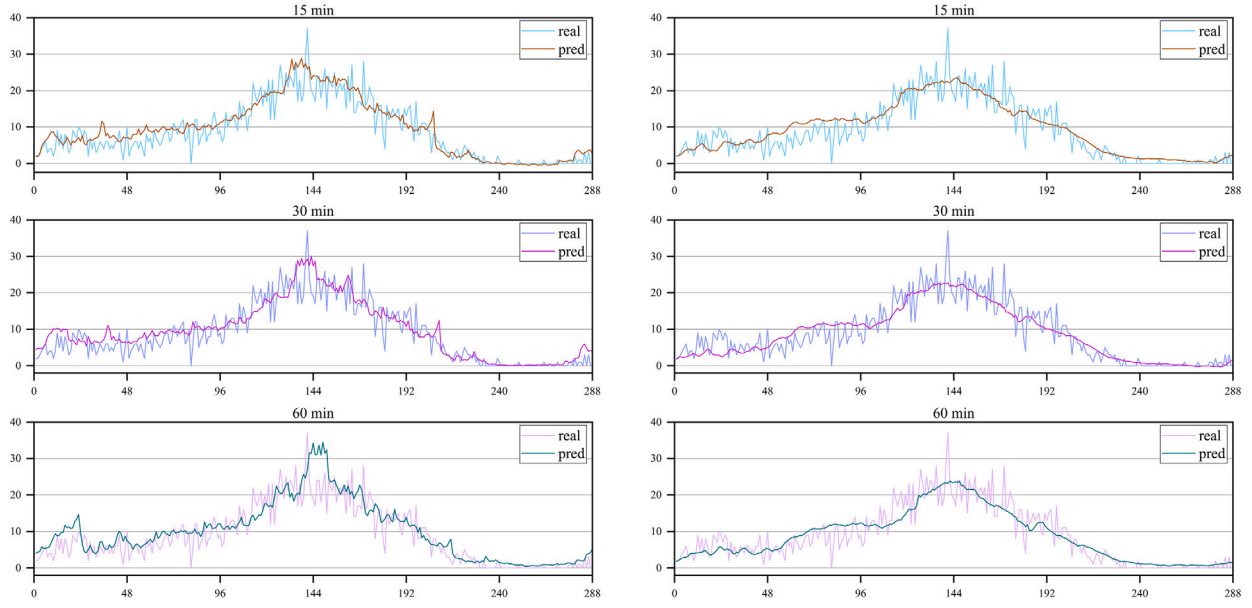
5.6. Importance of spatial-temporal embedding components and mixed joint architecture (RQ3)

In order to further evaluate the effectiveness of the temporal embedding and spatial embedding in our model, we conducted and analyzed ablation experiments on the PEMS08 dataset and designed the following model variants:

- $w/o\ E_1$: It is a variant of STEMLP without temporal embedding for periodicity information TP_1 ($P_1 = 288$).



(a) STEMLP (b) STID



(c) StemGNN (d) STGCN

Fig. 6. The comparison of predicted traffic flow for different models and the actual traffic flow for sensor #149 on August 30th, 2016.

- $w/o\ E_2$: It is a variant of STEMLP without temporal embedding for periodicity information TP_2 ($P_2 = 144$).
- $w/o\ E_3$: It is a variant of STEMLP without temporal embedding for periodicity information TP_3 ($P_3 = 96$).
- $w/o\ E_a$: It is a variant of STEMLP without spatial information E_{ada} for adaptive graph G_a.
- $w/o\ E_p$: It is a variant of STEMLP without spatial information E_{pre} for predefined graph G_p.

The prediction performance of the STEMLP model and its variants for prediction horizons of 15 minutes, 30 minutes and 60 minutes are shown as green bars in Fig. 7. We can get the following conclusions:

Table 5
Absolute training time for one epoch in different datasets.

Methods	Training (second/epoch)				
	PEMS04	PEMS07	PEMS08	PEMS-BAY	TFA
STGCN	18.26	51.32	11.53	51.21	17.86
DGCRN	82.95	502.96	81.68	525.29	115.95
StemGNN	10.85	48.76	11.87	31.51	12.92
GTS	31.07	434.28	29.81	249.94	42.68
MTGNN	17.36	52.25	15.07	58.29	26.42
GWNet	16.46	91.23	10.35	63.13	17.99
STEMLP (ours)	**9.12**	**19.36**	**10.03**	**29.13**	**10.95**
Promotion	+15.94%	+60.30%	+3.19%	+7.55%	+15.25%

Note: Bold text denotes the best performance, while underlined text indicates the second-best. 'Promotion' refers to the improvement margin of STEMLP over the top-performing STGNNs baseline model.

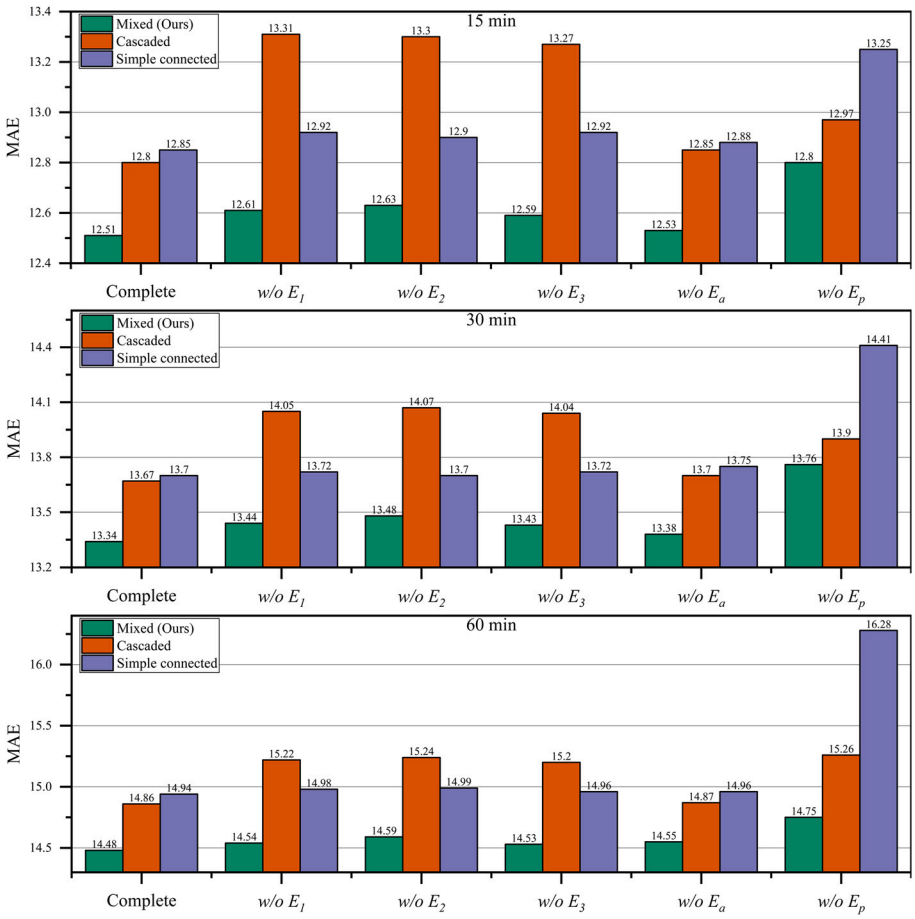
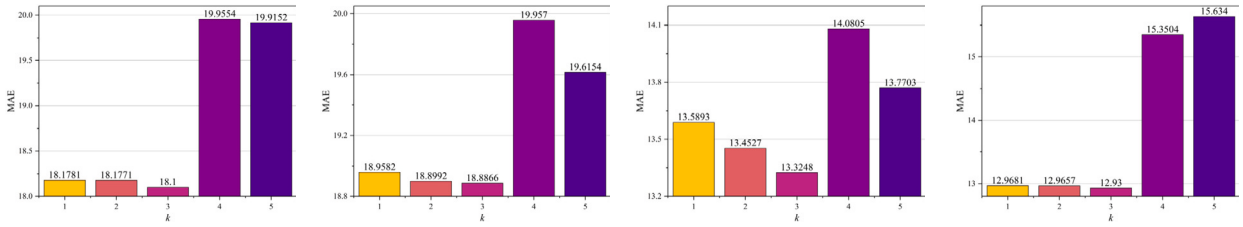


Fig. 7. Test MAE of variants of STEMLP on PEMS08.

(1) All embeddings play an active role in improving the model's prediction accuracy. Among these five different spatial and temporal embeddings, eliminating E_p leads to significant performance degradation for all prediction horizons, indicating that the capture of spatial dependencies related to the road network is very critical to achieve accurate prediction.
(2) In terms of temporal embeddings, the E_* variant exhibits a decrease in prediction performance compared to the STEMLP model. This also suggests that the integration of temporal embedding for time periods explored by using Fourier transform is effective to capture of traffic patterns, thereby further achieving a better accuracy of traffic flow prediction.

Table 6
MAE, RMSE, MAPE for different L_A, L_B, L_C on dataset PEMS08.

		$L_A=1, L_B=1$	$L_A=1, L_B=2$	$L_A=1, L_B=3$	$L_A=2, L_B=1$	$L_A=2, L_B=2$	$L_A=2, L_B=3$	$L_A=3, L_B=1$	$L_A=3, L_B=2$	$L_A=3, L_B=3$
MAE	$L_C=1$	13.79	13.59	13.48	13.59	13.48	13.43	13.43	13.45	**13.32**
	$L_C=2$	13.48	13.41	13.37	13.48	13.43	13.35	13.42	13.36	13.34
	$L_C=3$	13.46	13.34	13.36	13.39	13.38	13.33	13.40	13.37	13.41
RMSE	$L_C=1$	22.94	22.77	22.78	22.85	22.79	22.77	22.79	22.80	**22.75**
	$L_C=2$	22.82	22.81	22.95	22.88	22.90	22.91	22.95	22.90	22.89
	$L_C=3$	22.99	22.98	23.10	22.94	23.05	23.08	23.02	23.03	23.21
MAPE	$L_C=1$	9.01%	8.91%	8.75%	8.90%	8.79%	8.80%	8.80%	8.79%	**8.71%**
	$L_C=2$	8.81%	8.74%	8.78%	8.77%	8.75%	8.76%	8.77%	8.73%	8.74%
	$L_C=3$	8.77%	8.79%	8.70%	8.76%	8.76%	8.74%	8.75%	8.75%	8.76%



(a) PEMS04 (b) PEMS07 (c) PEMS08 (d) TFA

Fig. 8. Sensitivity analysis of hyper parameters k on the PEMS04, PEMS07, PEMS08 and TFA datasets.

To illustrate the effectiveness of the MLP-based mixed joint architecture in this paper, we proposed two variants of STEMLP which are simple connected and cascaded for comparison under the same temporal embedding, spatial embedding, and data embedding expression in Fig. 4.

- Cascaded: The architecture focuses on hierarchical integration of temporal embedding E_t, spatial embedding E_s, and data embedding E_d. The final feature expression is obtained as $MLP(MLP(MLP(E_t) \parallel E_s) \parallel E_d)$.
- Simple connected: Simple concatenation of temporal embedding E_t, spatial embedding E_s, and data embedding E_d. The final feature expression is obtained as $MLP(E_t \parallel E_s \parallel E_d)$.

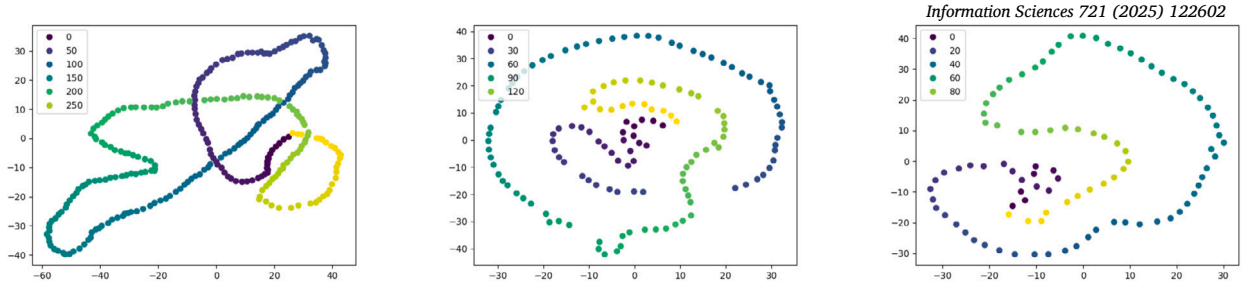
The prediction performance of the cascaded architecture variant and simple connected architecture variant of the STEMLP model are represented by the red and blue bars in Fig. 7, respectively. As shown in the results, our mixed joint architecture consistently outperforms the two alternative architectures across the three prediction horizons, highlighting its significant role in enhancing model performance. Notably, the MLP-based mixed joint architecture still yields improvements over the other two architectures, even without the use of specific embedding.

5.7. Analysis of the effect of hyper parameters on model performance (RQ4)

In order to observe the effect of the variation of the number of linear layers L_A, L_B, L_C in the three MLP modules of our model on prediction performances, we set each parameter to $L_A \in \{1,2,3\}, L_B \in \{1,2,3\}, L_C \in \{1,2,3\}$ on the PEMS08 dataset and conducted experimental comparisons. The results are shown in Table 6. Variations in L_A and L_B have a more significant impact on the prediction performances, while L_C has a minor impact. We selected $L_A=3, L_B=3, L_C=1$ as the best combination.

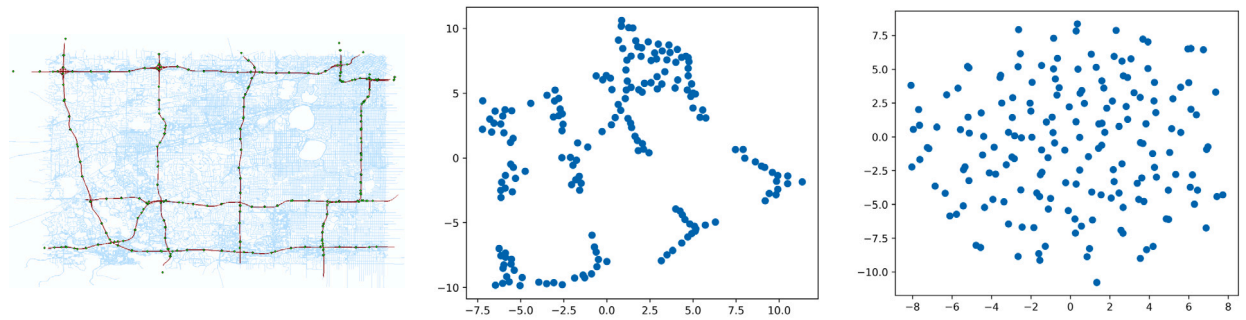
We use DFT to capture the *top*-5 significant time periods for PEMS04, PEMS07, PEMS08, PEMS-BAY and TFA datasets, which are {288, 144, 96, 154, 333}, {288, 144, 96, 155, 336}, {288, 144, 96, 155, 336}, {288, 144, 96, 2004, 336}, and{288, 144, 96, 2038, 335}. On this basis, we use the first k time periods to construct temporal embedding. Fig. 8 illustrates the variation of the averaged MAE of the proposed STEMLP model over 12 prediction time steps for temporal embedding with different k values.

As the value of k grows, the temporal embedding encodes richer periodicity information, thereby strengthening its ability to model temporal patterns in the data and ultimately improving traffic flow prediction accuracy. Experimental results demonstrate that setting $k=3$ achieves optimal prediction accuracy across four benchmark datasets. However, when $k>3$, the model's prediction performance on all four datasets suffers a significant decrease. This is mainly because the more temporal periodicity information there are, the more the temporal embedding process inevitably integrates part of the non-significant temporal periodicity information, reducing the quality of temporal embedding. In our experiments, we set k to 3, thereby selecting the initial three significant time periods from our datasets, which align with real-world time spans of 24 hours, 12 hours, and 8 hours, respectively. We can find that time spans as short as 2-3 months, as encountered in actual traffic datasets, do not align with the weekly periodicity typically discerned through intuitive analysis.



(a) Self-learning matrix B_1 (b) Self-learning matrix B_2 (c) Self-learning matrix B_3

Fig. 9. Visualization of learned temporal embedding under the PEMS08 dataset.



(a) Real traffic network (b) Spatial embedding of predefined graph E_p (c) Spatial embedding of adaptive graph E_a

Fig. 10. Visualization of real road networks and spatial embedding under the TFA dataset.

5.8. Spatial-temporal embedding interpretability analysis (RQ5)

To further understand the temporal embedding learning mechanism, we visualize the self-learning matrices $B_1 \in \mathbb{R}^{P_1 \times d_1}$, $B_2 \in \mathbb{R}^{P_2 \times d_2}$ and $B_3 \in \mathbb{R}^{P_3 \times d_3}$ of temporal embedding for three different temporal periodicities on the PEMS08 dataset, where $P_1 = 288$, $P_2 = 144$, $P_3 = 96$, $d_1 = d_2 = d_3 = 32$. Here, we use t-SNE [50] to visualize B_1, B_2 and B_3. Fig. 9 (a), (b), and (c) show the visualization of temporal embedding for 288-time slots (24 h) in time periodic length of P_1, 144-time slots (12 h) in time periodic length of P_2, and 96-time slots (8 h) in time periodic length of P_3, respectively. The key observation is that in each subfigure, the embedding vectors corresponding to different time points within one complete cycle form a distinct, continuous sequential arrangement in the low-dimensional projection space. Crucially, embedding vectors representing adjacent time points in the original sequence (e.g., time t and $t+1$) are positioned close together in the projection space. This phenomenon provides strong visual evidence that the self-learning matrices of the STEMLP model successfully encode the inherent sequential relationships within each periodicity into geometric proximity within the embedding space. Significantly, while the three subfigures correspond to different cycle lengths, they all exhibit this characteristic ordered sequential structure. This offers compelling visual confirmation that significant daily (24 h), semi-daily (12 h), and higher-frequency (8 h, likely capturing peak-valley patterns) periodic patterns coexist within the PEMS08 dataset.

In order to give a better physical interpretation of the spatial embedding, we chose to visualize the spatial embedding $E_p \in R^{N \times d_p}$ and $E_a \in R^{N \times d_a}$ of two different graph structures on the TFA dataset, where $N = 179$, $d_p = d_a = 32$. We have processed the TFA dataset from the raw traffic data, allowing us to gain a more precise and comprehension of the distribution of road sensors. This enhanced understanding enables us to examine the relationship of spatial embedding learning more effectively. Similarly to the application of temporal embedding, in this context, we use t-SNE to visualize E_p and E_a. Fig. 10(a) displays the physical topology of the Twin Cities highway network, where nodes represent traffic sensors and edges reflect geospatial connectivity. Fig. 10(b) visualizes node embeddings derived from the normalized Laplacian eigenvectors of the predefined graph, of which the adjacency matrix is constructed using distance-based spatial relationships (e.g., road network topology). Predefined graph spatial embedding preserves geospatial proximity, that is to say nodes close in the physical road network remain adjacent in the embedding space. This confirms that Laplacian eigenvector decomposition successfully encodes structural road connectivity. Fig. 10(c) illustrates embeddings from the data-adaptive graph, whose adjacency matrix is learned via nonlinear transformations of trainable matrices. Unlike predefined graph spatial embedding, adaptive graph spatial embedding reveals functional clustering patterns. It indicates nodes group based on semantic similarities (e.g., road type, traffic function), irrespective of physical proximity. For instance, sensors in residential zones or arterial roads form distinct clusters, reflecting latent traffic dynamics not captured by the predefined topology.

6. Discussion

The experimental validation of STEMLP reveals two insights for spatial-temporal traffic prediction. First, STEMLP outperforms both STGNNs and existing linear models, demonstrating the efficacy of its adaptive embedding strategy. Unlike STGNNs relying on complex graph convolutions and recurrent mechanisms, STEMLP achieves competitive accuracy through temporal periodicity information representation and eigenvector-based spatial encoding. This efficiency comes from decoupling complex spatiotemporal dependencies into modular embeddings, reducing computational overhead while preserving expressive power. Second, the ablation studies and visualizations confirm that both adaptive temporal periodicity and dual-graph spatial embeddings are pivotal to STEMLP's success. The DFT-driven temporal embedding mitigates the rigidity of fixed-period assumptions. Similarly, separating predefined and adaptive spatial embeddings avoids the representational ambiguity observed in prior work, where shared parameters diluted distinct spatial relationships. The t-SNE visualizations further validate that Laplacian eigenvectors translate topological proximity into geometric coherence in the embedding space, while adaptive embeddings uncover latent functional clusters beyond physical connectivity.

Despite these advantages, STEMLP's reliance on predefined graphs for spatial embedding presents a limitation. While adaptive graphs partially mitigate this by learning latent relationships, the initial dependency on distance-based adjacency matrices may not generalize to dynamically evolving road networks. Future work could explore online graph structure learning to further enhance adaptability. Additionally, while STEMLP excels in traffic flow and speed prediction, its applicability to other spatial-temporal tasks (e.g., anomaly detection) warrants investigation. The interpretability of its embeddings, as evidenced by the t-SNE analysis, opens avenues for domain-driven refinement, such as incorporating semantic attributes (e.g., road type) into the adaptive graph module.

Overall, STEMLP's success advocates reevaluating complexity in spatiotemporal models. It demonstrates that foundational techniques like DFT for temporal analysis, spectral embedding for spatial relationships, and MLPs for fusion, can achieve state of the art results when thoughtfully orchestrated. This shift toward simplicity, efficiency, and interpretability aligns with real world ITS requirements where deployability and maintainability are as crucial as accuracy.

7. Conclusion

In this paper, we present STEMLP, an advanced model for traffic flow prediction that addresses the challenges of spatial-temporal contextualization within a linear framework. STEMLP captures complex spatial-temporal patterns by introducing novel adaptive temporal embeddings derived from Discrete Fourier Transform and spatial embeddings based on the Laplacian eigenvectors of predefined and adaptive graphs. Additionally, it employs a mixed joint MLP architecture to effectively integrate spatial-temporal context embeddings for superior spatial-temporal correlation learning. Extensive experiments on five real-world datasets demonstrate that STEMLP significantly outperforms STGNN-based models and other MLP-based models in prediction accuracy and computational efficiency. Future work focuses on exploring the interpretability of STEMLP and its scalability to diverse datasets for broader spatial-temporal series prediction tasks.

CRediT authorship contribution statement

Liming Jiang: Writing – review & editing, Writing – original draft, Validation, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation. **Baiyi Liu:** Writing – original draft, Software, Resources, Methodology, Formal analysis, Data curation. **Huanyu Wang:** Writing – review & editing, Supervision, Project administration. **Shaomiao Chen:** Formal analysis, Conceptualization. **Wei Liang:** Supervision, Resources, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to my data/code at the manuscript subsection 5.3.

References

[1] C. Yin, Z. Xiong, H. Chen, J. Wang, D. Cooper, B. David, A literature survey on smart cities, Sci. China Inf. Sci. 58 (2015) 1–18.
[2] J. Wang, J. Jiang, W. Jiang, C. Li, W.X. Zhao, Libcity: an open library for traffic prediction, in: Proceedings of the 29th International Conference on Advances in Geographic Information Systems, SIGSPATIAL '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 145–148.
[3] B.M. Williams, L.A. Hoel, Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: theoretical basis and empirical results, J. Transp. Eng. 129 (2003) 664–672.
[4] Z. Zhao, W. Chen, X. Wu, P.C. Chen, J. Liu, Lstm network: a deep learning approach for short-term traffic forecast, IET Intell. Transp. Syst. 11 (2017) 68–75.
[5] J. Zhang, Y. Zheng, D. Qi, Deep spatio-temporal residual networks for citywide crowd flows prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, 2017.
[6] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18, AAAI Press, 2018, pp. 3634–3640.

[7] B. Long, W. Zhu, J. Xiao, St-retnet: a long-term spatial-temporal traffic flow prediction method, in: Chinese Conference on Pattern Recognition and Computer Vision (PRCV), Springer, 2024, pp. 3–16.
[8] D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu, C. Huang, Y. Tong, B. Xu, J. Bai, J. Tong, et al., Spectral temporal graph neural network for multivariate time-series forecasting, Adv. Neural Inf. Process. Syst. 33 (2020) 17766–17778.
[9] C. Shang, J. Chen, Discrete graph structure learning for forecasting multiple time series, in: Proceedings of International Conference on Learning Representations, 2021.
[10] W. Zhu, B. Long, J. Xiao, Spatial-temporal retentive heterogeneous graph convolutional network for traffic flow prediction, in: 2024 International Joint Conference on Neural Networks (IJCNN), IEEE, 2024, pp. 1–8.
[11] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A.X. Liu, S. Dustdar, Pyraformer: low-complexity pyramidal attention for long-range time series modeling and forecasting, in: International Conference on Learning Representations, 2021.
[12] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: decomposition transformers with auto-correlation for long-term series forecasting, Adv. Neural Inf. Process. Syst. 34 (2021) 22419–22430.
[13] W. Zhang, L. Zhang, J. Han, H. Liu, Y. Fu, J. Zhou, Y. Mei, H. Xiong, Irregular traffic time series forecasting based on asynchronous spatio-temporal graph convolutional networks, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 4302–4313.
[14] X. Liu, Y. Xia, Y. Liang, J. Hu, Y. Wang, L. Bai, C. Huang, Z. Liu, B. Hooi, R. Zimmermann, Largest: a benchmark dataset for large-scale traffic forecasting, in: A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems, vol. 36, Curran Associates, Inc., 2023, pp. 75354–75371.
[15] M. Jin, H.Y. Koh, Q. Wen, D. Zambon, C. Alippi, G.I. Webb, I. King, S. Pan, A survey on graph neural networks for time series: forecasting, classification, imputation, and anomaly detection, arXiv preprint, arXiv:2307.03759, 2023.
[16] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting?, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 11121–11128.
[17] Y. Qin, H. Luo, F. Zhao, Y. Fang, X. Tao, C. Wang, Spatio-temporal hierarchical MLP network for traffic forecasting, Inf. Sci. 632 (2023) 543–554.
[18] Z. Shao, Z. Zhang, F. Wang, W. Wei, Y. Xu, Spatial-temporal identity: a simple yet effective baseline for multivariate time series forecasting, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 4454–4458.
[19] Z. Wang, Y. Nie, P. Sun, N.H. Nguyen, J. Mulvey, H.V. Poor, ST-MLP: a cascaded spatio-temporal linear framework with channel-independence strategy for traffic forecasting, arXiv preprint, arXiv:2308.07496, 2023.
[20] T. Nie, G. Qin, L. Sun, W. Ma, Y. Mei, J. Sun, Contextualizing mlp-mixers spatiotemporally for urban traffic data forecast at scale, IEEE Trans. Intell. Transp. Syst. (2024).
[21] Y. Wang, M. Papageorgiou, Real-time freeway traffic state estimation based on extended Kalman filter: a general approach, Transp. Res., Part B, Methodol. 39 (2005) 141–167.
[22] G. Leshem, Y. Ritov, Traffic flow prediction using adaboost algorithm with random forests as a weak learner, Int. J. Math. Comput. Sci. 1 (2007) 1–6.
[23] S. Guo, Y. Lin, N. Feng, C. Song, H. Wan, Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 922–929.
[24] S. Guo, Y. Lin, H. Wan, X. Li, G. Cong, Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting, IEEE Trans. Knowl. Data Eng. 34 (2021) 5415–5428.
[25] Z. Fang, Q. Long, G. Song, K. Xie, Spatial-temporal graph ode networks for traffic flow forecasting, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 364–373.
[26] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: data-driven traffic forecasting, in: International Conference on Learning Representations (ICLR '18), 2018.
[27] J. García-Sigüenza, F. Llorens-Largo, L. Tortosa, J.F. Vicent, Explainability techniques applied to road traffic forecasting using graph neural network models, Inf. Sci. 645 (2023) 119320.
[28] L. Bai, L. Yao, C. Li, X. Wang, C. Wang, Adaptive graph convolutional recurrent network for traffic forecasting, Adv. Neural Inf. Process. Syst. 33 (2020) 17804–17815.
[29] Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph wavenet for deep spatial-temporal graph modeling, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 1907–1913.
[30] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, C. Zhang, 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, 2020, pp. 753–763.
[31] R.W. Liu, M. Liang, J. Nie, Y. Yuan, Z. Xiong, H. Yu, N. Guizani, STMGCN: mobile edge computing-empowered vessel trajectory prediction using spatio-temporal multigraph convolutional network, IEEE Trans. Ind. Inform. 18 (2022) 7977–7987.
[32] J. Jiang, C. Han, W.X. Zhao, J. Wang, PDFormer: propagation delay-aware dynamic long-range transformer for traffic flow prediction, in: AAAI, AAAI Press, 2023.
[33] Q. Tian, Y. Chen, Z. Zhang, H. Lu, L. Chen, L. Xie, S. Liu, TFGAN: time and frequency domain based generative adversarial network for high-fidelity speech synthesis, arXiv preprint, arXiv:2011.12206, 2020.
[34] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, J. Zhang, Urban traffic prediction from spatio-temporal data using deep meta learning, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1720–1730.
[35] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, Fedformer: frequency enhanced decomposed transformer for long-term series forecasting, in: International Conference on Machine Learning, PMLR, 2022, pp. 27268–27286.
[36] Z. Tang, J. Li, Y. Hao, R. Hong, Mlp-jcg: multi-layer perceptron with joint-coordinate gating for efficient 3d human pose estimation, IEEE Trans. Multimed. 25 (2023) 8712–8724.
[37] F.J. Rendón-Segador, J.A. Álvarez-García, A.J. Varela-Vaca, Paying attention to cyber-attacks: a multi-layer perceptron with self-attention mechanism, Comput. Secur. 132 (2023) 103318.
[38] K. Yi, Q. Zhang, W. Fan, S. Wang, P. Wang, H. He, N. An, D. Lian, L. Cao, Z. Niu, Frequency-domain mlps are more effective learners in time series forecasting, Adv. Neural Inf. Process. Syst. 36 (2024).
[39] W. Duan, H. Rao, W. Huang, X. He, Minimalist traffic prediction: linear layer is all you need, arXiv preprint, arXiv:2308.10276, 2023.
[40] B. Osgood, The Fourier transform and its applications, Lect. Notes EE 261 (2009) 20.
[41] C. Chatfield, H. Xing, The Analysis of Time Series: an Introduction with R, Chapman and Hall/CRC, 2019.
[42] L. Han, H.-J. Ye, D.-C. Zhan, The capacity and robustness trade-off: revisiting the channel independent strategy for multivariate time series forecasting, arXiv preprint, arXiv:2304.05206, 2023.
[43] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (2003) 1373–1396.
[44] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and deep locally connected networks on graphs, in: 2nd Int. Conf. On Learning Representations, ICLR 2014—Conference Track Proceedings, 2014.
[45] Q. Zhang, J. Chang, G. Meng, S. Xiang, C. Pan, Spatio-temporal graph structure learning for traffic forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 1177–1185.
[46] F. Li, J. Feng, H. Yan, G. Jin, F. Yang, F. Sun, D. Jin, Y. Li, Dynamic graph convolutional recurrent network for traffic prediction: benchmark and solution, ACM Trans. Knowl. Discov. Data 17 (2023) 1–21.

[47] V. Ekambaram, A. Jati, N. Nguyen, P. Sinthong, J. Kalagnanam, Tsmixer: lightweight mlp-mixer model for multivariate time series forecasting, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 459–469.
[48] Y. Liang, Z. Shao, F. Wang, Z. Zhang, T. Sun, Y. Xu, BasicTS: an open source fair multivariate time series prediction benchmark, in: International Symposium on Benchmarking, Measuring and Optimization, Springer, 2022, pp. 87–101.
[49] Y. Wang, EasyTorch: simple and powerful pytorch framework, https://github.com/cnstark/easytorch, 2020.
[50] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008).