TLAST: A Time-Lag Aware Spatial-Temporal Transformer for Traffic Flow Forecasting

Qi Zheng, Minhua Shao, and Yaying Zhang, *Member, IEEE*

***Abstract*— Traffic flow forecasting is a strongly supportive component of intelligent transportation services. While in light of the expanding road networks or city grids, there is a critical concern to enhance both the accuracy and efficiency of prediction models. Despite the remarkable improvements in prediction accuracy, existing research continues to face three limitations in practical engineering scenarios. Firstly, current research often overlooks the time delay characteristics when capturing spatial relationships between global nodes. Secondly, most approaches have a quadratic computational complexity with respect to the number of nodes, resulting in significant training overhead and poor scalability. Furthermore, studies that do consider dynamic spatial relationships typically require complex model structures, resulting in higher computational costs. To address these issues, we propose a Time-Lag Aware Spatial-temporal Transformer (TLAST), a lightweight yet effective traffic flow forecasting model. TLAST introduces a cross-time strategy into the embedding stage and the attention extraction to capture the time-lag aware spatial-temporal features. Furthermore, we propose a Spatial Proxy Attention (SPA) module. It utilizes proxy representations to efficiently capture time-varying spatial dependencies with linear complexity, significantly reducing computational overhead. Extensive experiments on seven real-world traffic datasets demonstrate that TLAST consistently outperforms state-of-the-art baselines, achieving up to 7.84% improvement in prediction accuracy (MAE) while reducing memory usage and time cost by 85.21% and 75.14%, respectively. Results from the empirical analysis not only demonstrate the model's efficiency and scalability but also highlight its practical usability in real-world traffic forecasting scenarios.**

***Index Terms*— Spatial-temporal data mining, self-attention, linear transformer, traffic flow forecasting, time lag.**

I. Introduction

TRAFFIC flow forecasting aims to predict future traffic flow for spatial nodes (*e.g.*, road sensors, city grids) within a transportation network. As a strongly supportive component of intelligent transportation services, traffic prediction attracts more and more attention in both academics and industry. The recognition of the spatial-temporal relationship between traffic nodes is a key prerequisite of traffic prediction, and a thorough comprehension of spatial-temporal characteristics can guide the concise and effective design of model architecture. Furthermore, the traffic monitoring capabilities in cities are continuously evolving, driven by the rising deployment of road sensors and the improved granularity of urban area division. These factors contribute to the growing complexity of road networks or city grids. Consequently, meeting prediction needs in practical traffic scenarios requires a careful balance between algorithm accuracy and efficiency. Despite the remarkable progress in prediction accuracy within existing research, there are still two challenges that warrant further exploration: the modeling of time-lagged spatial-temporal patterns and the computational efficiency of prediction models. These challenges are described in detail below.

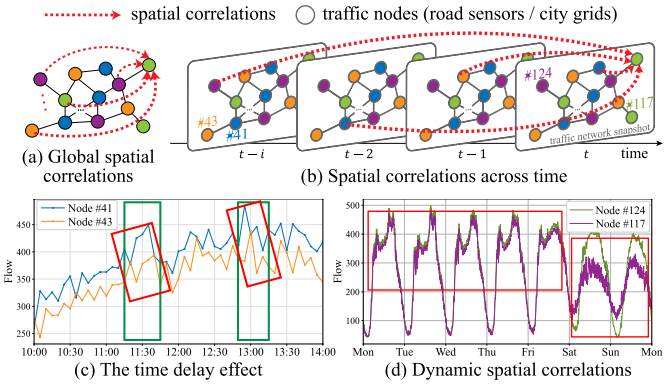


Fig. 1. Examples of global spatial associations with time entanglement. Data of nodes are from the PEMS08 dataset [1]. (a) Full global spatial correlations results in a quadratic complexity. (b) Global spatial relationships persist not only within the same time interval but also across time. (c) Traffic patterns of adjacent nodes (Node #41 and Node #43) exhibit similarity in a lagged time window (red rectangles) while being opposite in the same time periods (green rectangles). (d) Global nodes (Node #124 and Node #117) display a weekday similarity but diverge on weekends.

1) **The spatial interdependence of traffic nodes exhibits time-lagged effects and dynamic variations as traffic propagates.** First, the spatial relationships among global traffic nodes exhibit temporal continuity due to the traffic propagation. As illustrated in Figure 1(a)(b), global spatial relationships persist not only within the same time interval but also across time, particularly those between geographically distant nodes [2]. Intuitively, Figure 1(c) epitomizes this time

Received 29 November 2023; revised 13 June 2025; accepted 23 June 2025. Date of publication 10 July 2025; date of current version 16 September 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB4501704, in part by the National Natural Science Foundation of China under Grant 72342026, and in part by the Fundamental Research Funds for the Central Universities under Grant 2024-6-ZD-02. The Associate Editor for this article was N. Attoh-Okine. *(Corresponding author: Yaying Zhang.)*

Qi Zheng and Yaying Zhang are with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 200092, China (e-mail: zhengqi97@tongji.edu.cn; yaying.zhang@tongji.edu.cn).

Minhua Shao is with the School of Transportation, Tongji University, Shanghai 200092, China (e-mail: shaominhua@tongji.edu.cn).

Digital Object Identifier 10.1109/TITS.2025.3583391

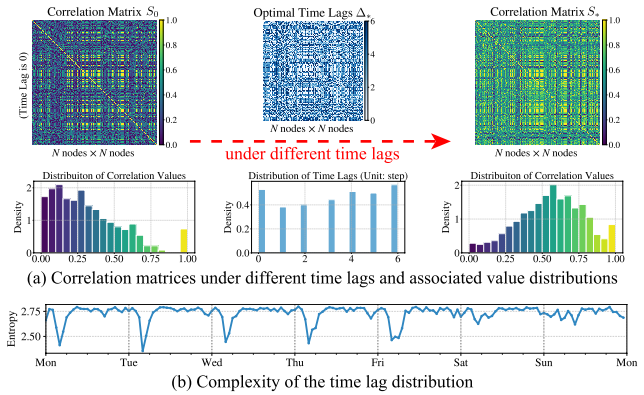


Fig. 2. Examples of the maximum spatial associations among various time lags on PEMS08 dataset. (a) The inter–node correlations increase markedly under time delays. (b) The complexity of time lags are also time-varying.

delay effect: two spatially adjacent nodes exhibit similar traffic patterns only when analyzed with a temporal offset (red rectangles), yet display opposite behaviors when compared synchronously (green rectangles). This observation indicates that Node 43 lags Node 41 by one time step. Motivated by this illustrative case, we systematically quantify the delay effect across the entire network based on the max-cross-correlation method [3] (formal details are provided in Section III). Figure 2(a) depict the inter–node correlation matrix and its distribution for the PEMS08 dataset between 10:00 am and 11:00 am, revealing that correlations increase substantially under various time lags. Thus, incorporating time lags into spatial–correlation modeling uncovers latent, effective dependencies and enables the model to learn spatio-temporal patterns with reduced bias.

Second, spatial correlations among traffic nodes are intrinsically time–varying. Figure 1(d) exemplifies this by showing a pair of global nodes whose patterns align on weekdays but diverge on weekends. When optimizing for maximal correlation via appropriate time lags, the complexity of these delays also varies across time, as shown in Figure 2(b). Accordingly, capturing the dynamic nature of spatial correlations is a critical component of accurate traffic forecasting.

2) **Modeling spatial correlations between all pairs of nodes lead to significant computational costs and poor scalability of the prediction model.** Beyond temporal entanglement, spatial scale presents another key challenge for inter-node correlation analysis. Predefined local correlations based on geographic distance or connectivity is insufficient to reveal the potential relationships of global nodes [4], [5]. Thus the focus of spatial mining research has expanded to a global scale. To perceive the global spatial correlation of traffic nodes, a common and intuitive approach is to model pairwise relationships between all nodes, as shown in examples of Figure 1(a). However, modeling the pairwise correlation often results in quadratic growth of training costs with the increasing number of traffic nodes, making it prohibitively expensive for large road networks or fine-grained urban grids. Moreover, to achieve a comprehensive understanding and prediction of traffic dynamics, the spatial correlations among nodes can often extend beyond the temporal perspective to a broader perceptual domain [1], potentially increasing the scale of global feature space. In such scenarios, traversing each pair of elements in the spatial-temporal domain for prediction becomes computationally demanding. Hence, the crucial challenge lies in finding ways to reduce computational complexity while preserving superior prediction accuracy.

Many deep-learning-based studies in the traffic flow forecasting field have turned the research perspective to dynamic spatial mining on a global scope. A number of research constructs multiple graphs at timeline levels or spatial semantic levels to reveal the time-varying and various correlations among traffic nodes. However, the global spatial relationships they establish are frequently limited to nodes within the same time interval and lack connectivity with temporal spans. This constraint poses a challenge in addressing the time lagged effect of spatial relationships. Additionally, the consideration of complex dynamic spatial-temporal dependencies in these methods has resulted in increasingly intricate model structures and escalating computational costs. Furthermore, certain studies rely on pre-analyzing the dataset to provide node-similarity priors, but this will introduce additional preprocessing overhead and lack the flexibility to directly capture the temporal delays of spatial dependencies from individual input samples.

To address these issues, we propose a **T**ime **L**ag **A**ware **S**patial-Temporal **T**ransformer for traffic flow forecasting (TLAST). To effectively capture time-lagged and dynamic spatiotemporal features, we introduce a cross-time spatial-temporal embedding method, within which a time-lag embedding module is introduced to model the gaps between time embeddings as a unique identifier. Second, during attention-based spatial-temporal modeling, we link the spatial features of the most recent time interval with those of all preceding time steps, enabling direct modeling of delay-aware dependencies. In terms of enhancing model efficiency, the aforementioned designs for node embedding and feature extraction are instrumental in streamlining model architectures and lowering computational overhead. In addition, we propose a Spatial Proxy Attention module for spatio-temporal feature extraction. It dynamically aggregates node features into a compact set of proxy node representations via learnable readout functions. By computing attention weights between proxy nodes and all nodes, we obtain full-resolution spatiotemporal features while reducing computational complexity to a linear scale relative to node count. This helps to preserve competitive prediction performance while significantly reduce computational overhead.

The main contributions of this work are as follows:

- A Time-Lag Aware Spatial-temporal Transformer (TLAST) model framework is proposed for traffic flow forecasting. We propose a new Time-Lag Embedding module to explicitly model the time-lagged feature as a unique temporal identified information. Meanwhile, we extract global node correlations between the most recent time interval and previous ones. These break the constraint of extracting global and dynamic spatial relationships in an isolated time interval. This cross-time setting helps to efficiently extract the spatial-temporal dependencies from a perspective of traffic propagation with light training costs.

- We propose a Spatial Proxy Attention (SPA) module that employs a two–stage multi–head attention mechanism to linearly capture spatiotemporal dependencies across time. Meanwhile, a linear prediction module compresses the extracted spatiotemporal features into a latent representation, from which time–interval–wise predictions are generated. Together, these components collectively achieve linear computational complexity with regard to both traffic node quantity and sequence length. This framework allows models to be feasibly trained on large traffic datasets while preserving superior prediction accuracy.
- Experiments results on seven real-world datasets show that TLAST outperforms state-of-the-art baselines in terms of prediction accuracy, while also significantly reducing computational consumption. Compared to the baseline method with the best accuracy, TLAST achieves up to 85.21%/ 75.14% improvement in memory overhead and training time cost. Furthermore, extensive empirical studies provide insights into the spatio-temporal characteristics of diverse traffic datasets, thereby aiding in the development of targeted traffic spatial-temporal mining.

II. Related Works

We aim to forecast future traffic conditions by analyzing the spatial-temporal connections between nodes using a Transformer-like architecture. However, faced with the burgeoning data scale in the real world, considering both the accuracy and scalability of spatial-temporal prediction algorithms has become an increasingly critical concern. We will first review the spatial-temporal modeling methods in existing traffic flow forecasting research and two groups of complexity reduction methods in Transformer models.

A. Spatial-Temporal Mining for Traffic Flow Forecasting

To achieve high forecasting accuracy, it is crucial to model latent spatial-temporal patterns of the traffic data. Early methods [6], [7] are utilized for summarizing the temporal characteristics of traffic flow changes. They are computationally efficient but labored to achieve good accuracy in complex scenarios. Deep learning methods exemplified by Convolution Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Graph Neural Networks (GNNs), and Attention-based models have showed great potential of spatial-temporal patterns mining.

Graph-based methods [1], [4], [8], [9], [10], [11], [12], [13], [14] are effective in modeling message propagation between spatial nodes, but learning reliable and time-coupled global graph structures remains challenging and requires a significant amount of time and effort. Many Attention-based studies [5], [15], [16], [17], [18], [19], [20] consider the non-locality and dynamics of spatial connections with the attention mechanism. Furthermore, ST-SSL [21] and SSTBAN [18] incorporate self-supervised learning methods into the field of traffic flow forecasting. In contrast, our TLAST focuses on the design of encoder and predictor, without relying on self-supervised methods, yet allowing for seamlessly integration with them.

Recent advancements in traffic prediction research remain centered on modeling dynamic inter-node correlations, where multi-view graph structures are constructed through dynamic graph learning modules or attention mechanisms [22], [23], [24], [25]. Building upon this foundation, additional traffic-specific considerations have been systematically incorporated, including hierarchical temporal decoupling of sequences [25], [26], prioritization of pivotal graph nodes [27], and time-delay feature modeling [17]. Concurrently, novel approaches adopt adaptive embeddings and hybrid neural network architectures (*e.g.*, MLPs, Mixture-of-Experts) for structural innovation [19], [20], [26], [28], effectively reducing computational overhead while maintaining performance.

Among them, PDFormer [17] distinctively introduces a feature transformation module to model the time delay in spatial propagation by clustering short-term historical traffic flow series in advance. STAEformer [19] proposes spatio-temporal adaptive embeddings, empowering vanilla Transformers to achieve state-of-the-art accuracy without complex architectural modifications. TESTAM [20] introduces a Mixture-of-Experts (MoE) architecture with dedicated modules to dynamically capture traffic patterns. Compared with these methods, our TLAST focuses on spatial mining by Transformer-type architecture with less computational complexity. It aims to capture the dynamic and time-lagged spatial-temporal features with a lightweight time lag aware modeling.

B. Complexity Reduction in Transformer Models

To capture inter-node spatial-temporal features for traffic forecasting, Transformer [29] is a common-used and effective tool based on the attention mechanism. A canonical Transformer explicitly extracts associations between all pairs of input tokens and generates a weighted representation, which results in quadratic complexity in both space and time level. In recent years, researchers have proposed numerous methods to reduce complexity in Transformer models. These methods can be broadly categorized into two groups: The first group focuses on leveraging mathematical methods [30], [31], [32] to approximate or replace the original attention mechanisms in the Transformer architecture. These studies primarily target graph nodes that do not have any inherent sequence relationships. In comparison, our TLAST also targets spatial nodes but we do not modify the attention architecture itself by any replacements.

The second group aims to reduce the number of input tokens for attention mechanism. It minimize the computations required for pair-wise attention score calculations [33], [34], [35] based on certain prior assumptions (*e.g.*, temporal order). In addition to treating temporal sequence elements as tokens, two studies also employ this approach to capture correlations between spatial representations: Crossformer [36] introduces a route mechanism with static learnable parameters to achieve linear complexity with respect to variable dimensions, which motivates our work. Similarly, SSTBAN [18] employs a bottleneck mechanism using fixed proxy tensors to reduce the complexity related to node count. Our TLAST falls within this group but differs by adopting a learnable function to

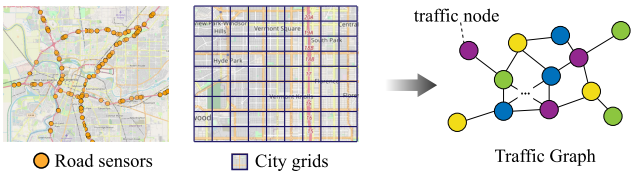


Fig. 3. Two types of traffic graphs: road networks (left) and city regions (middle). The traffic nodes are road sensors and city grids, respectively.

generate dynamic proxy nodes from input features. These proxies interact with representations from different time steps, highlighting their temporal adaptability.

III. Preliminary

Various types of transportation network systems can be represented as graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In terms of the spatial scope of a single observation point, traffic graphs can be divided into two types: road networks and city regions, as in Figure 3. Accordingly, the node set $\mathcal{V}$ represents a group of sensors in road segments or multiple city regions, where $|\mathcal{V}| = N$, and N is the number of traffic nodes. $\mathcal{E}$ contains edges between traffic nodes in terms of geographical distance or connectivity. Generally, city regions can be divided into Euclidean divisions and non-Euclidean divisions based on urban geographical spaces. In this work, we focus on the Euclidean division of city regions, i.e., city grids, for diversity. Usually, an adjacency matrix $\mathcal{A}$ can be used to present this graph.

The 24 hours of a day can be equally divided into $\mathcal{T}$ time intervals according to certain frequencies. Each time interval has two time indexes, identifying the time interval of the day and the day of the week. The traffic value generated by all nodes in a time interval t_i can be aggregated into a matrix $X_{t_i} \in \mathbb{R}^{N\times C}$, where C is the number of traffic features (*e.g.*, flow). The traffic flow forecasting problem can be formulated as: Given road network or city grids $\mathcal{G}$, T time intervals and their time indexes, we need to use the historical traffic tensor $\mathbf{X} = [X_{t-T+1}, \ldots, X_{t-1}, X_t] \in \mathbb{R}^{T\times N\times C}$ to predict the future data in the next consecutive T' time intervals $\mathbf{Y} = [X_{t+1}, X_{t+2}, \ldots, X_{t+T'}]$. Here, X_t represents the traffic data at the latest time interval t.

To quantify the time delay effect in traffic data, we define the maximum correlation matrix S_* and associated time lag matrix Δ_* based on the max-cross-correlation method from literature [3]. Specifically, for an input $\mathbf{X} \in \mathbb{R}^{T\times N\times C}$ within a given time window, the cross-correlation matrix $R^{(\delta)} \in \mathbb{R}^{N\times N}$ under time delay δ is formulated as:

$$R^{(\delta)}_{i,j} = Corr(\mathbf{X}[\delta:, i, :], \mathbf{X}[: T-\delta, j, :]), \delta \in [0, \delta_{\max}], \quad (1)$$

where $Corr$ denotes the correlation computation function, which can be implemented using metrics such as Spearman coefficient, Pearson coefficient. $\delta_{\max}$ is the manually configured upper threshold for the time delay. Then the maximum correlation matrix S_* and associated time lag matrix Δ_* are defined as:

$$(S_*)_{i,j} = \max_{\delta} R^{(\delta)}_{i,j}, \quad (\Delta_*)_{i,j} = \operatorname*{argmax}_{\delta} R^{(\delta)}_{i,j}. \quad (2)$$

When $\delta = 0$, $S_0 = R^{(0)}$ is the prior correlation matrix within the same time interval. Additionally, we define the Time Lag Entropy to measure the complexity of optimal time lags:

$$Entropy(\Delta_*) = -\sum_{\delta=0}^{\delta_{\max}} p(\delta) \log_2 p(\delta). \quad (3)$$

Here $p(\delta) = \frac{\sum_{i,j} \mathbb{I}((\Delta_*)_{i,j}=\delta)}{N^2}$. The empirical analysis of these metrics motivated our model design, which will provide new insights into traffic data analysis.

IV. Methodology

Figure 4 shows the overall framework of the proposed TLAST model. The main idea of our model is to directly capture the spatial correlation between nodes in the latest period and past ones, and use a proxy mechanism to reduce the time complexity with a Transformer-type architecture. TLAST model mainly consists of three parts, 1) the *Time Lag Aware Embedding* module, 2) the *Cross-time Spatial-Temporal Feature Extractor (CSTFE)* module, and 3) the *Time-interval-wise Prediction Module*.

The raw input tensor $\mathbf{X} \in \mathbb{R}^{T\times N\times C}$ is firstly transformed into a high dimensional representation $\mathbf{Z} \in \mathbb{R}^{T\times N\times d}$ through the *Time Lag Aware Embedding* module, where d is the embedding dimension. Then we use a readout function to convert the representation $Z_t \in \mathbb{R}^{N\times d}$ in the latest time interval t into $P_t \in \mathbb{R}^{m\times d}$, where m is a constant value extremely smaller than N. P_t potentially denotes a high-level node summary information. T time intervals share one *Cross-time Spatial-Temporal Feature Extractor (CSTFE)* with the same parameters. This module contains a Transformer-type encoder with a skip connection. Differently, the self-attention block in the encoder layer is replaced by a proposed *Spatial Proxy Attention (SPA)* block. This block contains two attention calculations where the first one uses P_t to produce a small query and the second one recovers full node features, which significantly reduces the number of query-key pairs in calculation and explicitly extracts the cross-time spatial correlation. The outputs of the CSTFE are then concatenated into $H^{ST} \in \mathbb{R}^{N\times Td}$. Finally, the *Time-interval-wise Prediction Module* transform H^{ST} into future traffic features $\hat{\mathbf{X}} \in \mathbb{R}^{T'\times N\times C}$ with individual decoder units and the time skip connection from X_t.

A. Time Lag Aware Embedding

Our empirical analysis in Section III and Figure 2 shows that inter-node correlations increase markedly at particular time lags. Motivated by this, we hypothesize that incorporating delay information into the extraction of spatial dependencies among nodes can enhance predictive accuracy. However, explicitly enumerating all possible delay combinations would incur an unsustainable computational burden. To address this, we treat the features from the most recent time slice as an anchoring reference and encode delay information via the differences between this anchor and the features at various past time slices. The interaction between these two representations then serves to approximate the time-lag-aware spatial correlations. From a feature–encoding perspective, this embedding

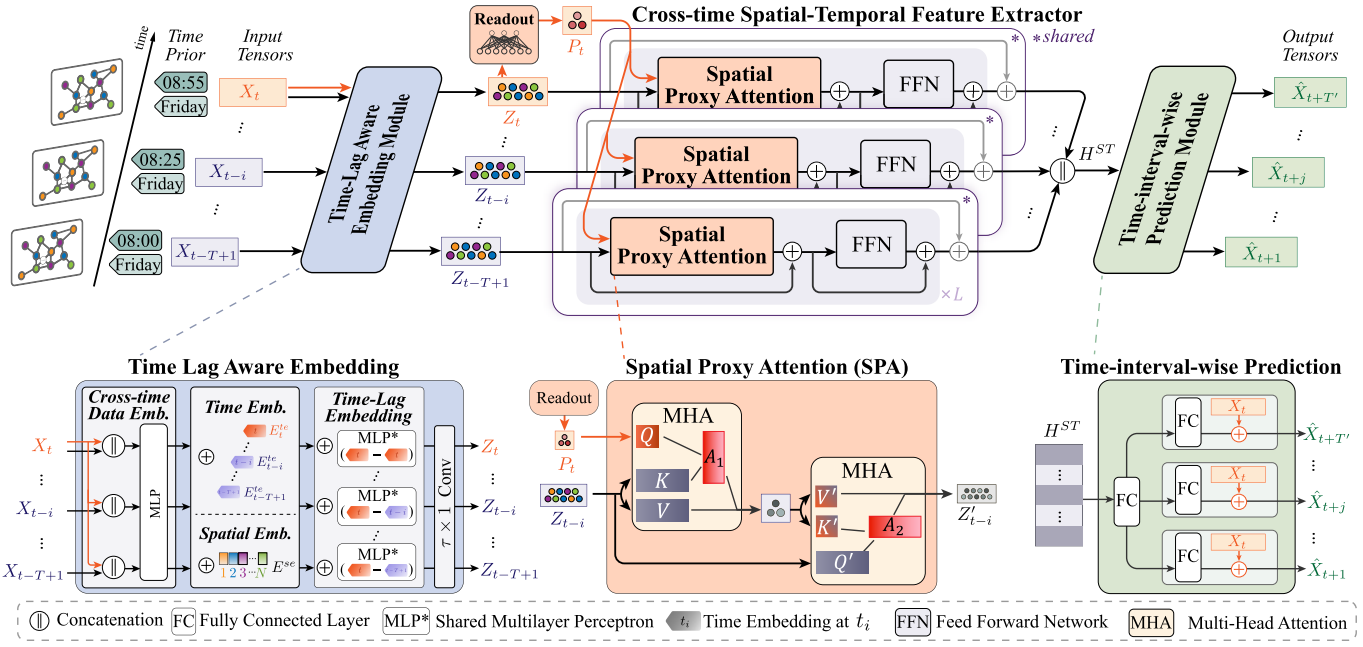


Fig. 4. The overall framework of the proposed TLAST model.

module enhances the model's sensitivity to time–lag information through three components.

The first component is the cross-time data embedding. We concatenate the input tensor in each time interval $t-i$ ($i \in \{0, 1, \cdots, T-1\}$) with that of the latest time interval t and convert them into a high-dimensional representation through an MLP layer:

$$X_{t-i}^{cross} = MLP_{\theta_1}([X_{t-i}||X_t]) \in \mathbb{R}^{N\times d}, \tag{4}$$

where $[\cdot||\cdot]$ denotes the concatenation along the feature dimension. We define MLP_θ as two fully-connected layers with a ReLU activation in between, θ is the corresponding parameters. MLP_{θ_1} here is shared on all time intervals. Collectively, the entire data embedding from T time intervals is $\mathbf{X}^{cross} = [X_{t-T+1}^{cross}, \ldots, X_{t-i}^{cross}, \ldots, X_t^{cross}] \in \mathbb{R}^{T\times N\times d}$.

The second component is the temporal embedding and spatial embedding. Traffic data generally show periodic changes on a daily and weekly scale. Time information (*i.e.*, the time interval of the day and the day of the week) is an intuitive, simple, and effective indicator of traffic flow changes. For example, traffic flow often peaks during *weekday rush hours*. In addition, traffic nodes distributed in urban space always show a certain degree of functional heterogeneity influenced by their geographical locations, although we usually only consider the same type of traffic data (*e.g.*, flow). This causes the difference in traffic patterns of different nodes. To make the model perceive the uniqueness of nodes in time and space, we introduce an embedding technique to represent these identifications. Specifically, we build three embedding tensors with learnable parameters: $\mathcal{D}^{time} \in \mathbb{R}^{\mathcal{T}\times d}$, $\mathcal{D}^{day} \in \mathbb{R}^{7\times d}$, and $\mathcal{D}^{space} \in \mathbb{R}^{N\times d}$ representing time intervals of a day, days of a week, and the space identification of all nodes. For the input tensor of T time intervals, by looking up with time indexes and node indexes, the temporal embeddings $X^{time} \in \mathbb{R}^{T\times d}$, $X^{day} \in \mathbb{R}^{T\times d}$ and the spatial embedding $E^{se} \in \mathbb{R}^{N\times d}$ can be obtained from $\mathcal{D}^{time}$, $\mathcal{D}^{day}$, and $\mathcal{D}^{space}$. The time embeddings are added into $E^{te} = X^{time} + X^{day}$.

The third component is the time-lag embedding. Building on the temporal embeddings, we compute the difference between the most recent time embedding and each historical one, and then encode these differences via another shared MLP to identify information of different time lags:

$$E_{t-i}^{tle} = MLP_{\theta_2}(E_t^{te} - E_{t-i}^{te}). \tag{5}$$

Note that all nodes share the same temporal embedding and time-lag embedding in each time interval and all time intervals share the same spatial embedding.

These embedding tensors are then added through a broadcasting mechanism with $\mathbf{X}^{cross}$ to generate $\mathbf{X}^{emb}$:

$$\mathbf{X}^{emb} = \mathbf{X}^{cross} + E^{te} + E^{se} + E^{tle} \in \mathbb{R}^{T\times N\times d}. \tag{6}$$

Here, during training phase, embeddings are followed by a dropout layer with rate ϕ to alleviate overfitting. Finally, inspired by [16], we use an equal-width convolution layer along with the time dimension to model the local trend with a $\tau \times 1$ kernel:

$$\mathbf{Z} = Conv_{\tau\times 1}(\mathbf{X}^{emb}) \in \mathbb{R}^{T\times N\times d}. \tag{7}$$

$\mathbf{Z}$ is the spatial-temporal context-aware data representation, and $\mathbf{Z} = [Z_{t-T+1}, \ldots, Z_{t-i}, \ldots, Z_t]$, where $Z_{t-i} \in \mathbb{R}^{N\times d}$.

B. Cross-Time Spatial-Temporal Feature Extractor

We aim to capture spatial correlations between global nodes in multiple time spans and then extract spatial-temporal features on different time lags. Meanwhile, time complexity reduction and redundancy alleviation are also important. To achieve these two objectives, we design a *Cross-time Spatial-Temporal Feature Extractor* module to model dynamic and time-lagged spatial-temporal dependencies in a linear time

complexity by three components: *Node Proxy Mechanism*, *Spatial Proxy Attention (SPA)* block, and a *Cross-Time Modeling* framework.

1) Node Proxy Mechanism: In the global space scale, it is not necessary to clearly explore the degree of spatial connection between each pair of nodes, because it is time-consuming and perhaps only a few nodes have important contributions. However, identifying these prominent nodes or pairs in advance is challenging. First, this selection process may introduce unwarranted induction bias. Second, the predefined node relationships may not always be valid over time. Instead of taking the entire set of nodes or a subset of them for attention computation, we first use a readout function to aggregate the node representation $Z_t \in \mathbb{R}^{N\times d}$ in the latest time interval into a fixed-size matrix $P_t \in \mathbb{R}^{m\times d}$, where m is a small constant value. This summary matrix P_t can be used to calculate attention scores with all nodes in each past time interval later. Specifically, we use a linear layer as the readout function:

$$P_t = (Z_t^\top W^P + b^P)^\top \in \mathbb{R}^{m\times d}, \tag{8}$$

where $W^P \in \mathbb{R}^{N\times m}$, $b^P \in \mathbb{R}^m$ are the parameters in the linear projection. P_t can be regarded as m proxy node representations summarizing various factors of entire nodes.

2) Spatial Proxy Attention (SPA) Block: We apply the Multi-Head Attention (MHA) mechanism [29] on the spatial dimension. This enables the input tokens to be spatial node representations on a single time interval, rather than a time series of a single node. The MHA mechanism we used in this work is formulated as:

$$\begin{aligned} MHA(Q,K,V) &= \text{Concat}(\text{head}_1, \text{head}_2 \ldots .\text{head}_h)W^O \\ \text{where head}_i &= \text{softmax}(\frac{(QW_i^Q)(KW_i^K)^\top}{\sqrt{d_k}})(VW_i^V), \end{aligned} \tag{9}$$

where $Q \in \mathbb{R}^{N_Q\times d}$, $K \in \mathbb{R}^{N_K\times d}$, $V \in \mathbb{R}^{N_K\times d}$ are the packed query, key, and value matrices, N_Q is the length of the query, N_K is the length of the key and value, $W_i^Q, W_i^K \in \mathbb{R}^{d\times d_k}$, $W_i^V \in \mathbb{R}^{d\times d_v}$, $W^O \in \mathbb{R}^{hd_v\times d}$ are parameters of linear projections, here bias vectors are omitted for brevity, h is the number of heads, and $d_k = d_v = d/h$.

An SPA block contains two MHA attention operations with different query-key-value combinations. For the first one, we use $P_t \in \mathbb{R}^{m\times d}$ as the query and $Z_{t-i} \in \mathbb{R}^{N\times d}$ as the key and value. The result of MHA_1 is used as the key and value in the second one, and Z_{t-i} is the query:

$$\mathcal{Z}_{t-i}^P = MHA_1(P_t, Z_{t-i}, Z_{t-i}) \in \mathbb{R}^{m\times d}, \tag{10}$$

$$Z'_{t-i} = MHA_2(Z_{t-i}, \mathcal{Z}_{t-i}^P, \mathcal{Z}_{t-i}^P) \in \mathbb{R}^{N\times d}. \tag{11}$$

The time complexity of these two attention operations is $\mathcal{O}(mNd)$. As m and d can be set to small constant values, this attention block has a linear time complexity with regard to the number of nodes. With Equation 10 and Equation 11, the SPA block can be formulated as:

$$Z'_{t-i} = SPA(P_t, Z_{t-i}) \in \mathbb{R}^{N\times d}. \tag{12}$$

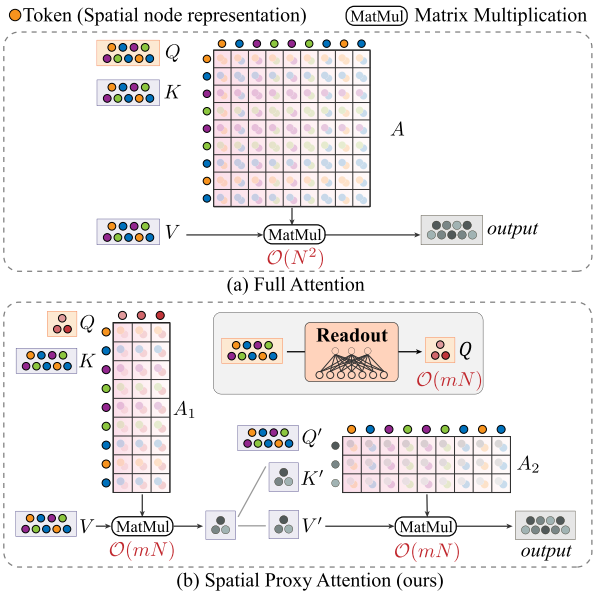


Fig. 5. Comparison of mechanism between the full attention and our spatial proxy attention.

The SPA block can be regarded as following a mini encoder-decoder architecture. The first MHA attention block encodes input tokens into intermediate proxy ones and the second block recovers full node representations by looking up the proxy ones. Correspondingly, attention matrices of a single head in two MHA blocks can be formulated as $A_1 \in \mathbb{R}^{m\times N}$ and $A_2 \in \mathbb{R}^{N\times m}$. Unlike prior works such as Crossformer and SSTBAN, which use fixed, learnable proxy tensors after model training (with sizes $\mathbb{R}^{L_{patch}\times m\times d}$ and $\mathbb{R}^{m\times 2d}$, respectively), our proxy tensor P_t are dynamic mappings of node representations obtained through a learnable readout function applied on Z_t.

Figure 5 illustrates a comparison among the process of the Full Attention and our SPA. A Full Attention block enables full query-key pairs to be computed and holds an attention score matrix with N^2 size. By contrast, to output a N-length node representation while reducing the number of query-key pairs, we reduce the length of query tokens and combine two smaller attention computation blocks. In an SPA block, only $2mN$ pairs need to be processed. Notably, when the proxy representation P_t is defined as an identity mapping of Z_t and the SPA module is replaced by a standard MHA block, this module is equivalent to a Transformer encoder layer (i.e., Full Attention) extracting spatial features across different time intervals.

3) Cross-Time Modeling Framework: The core of this CSTFE module is encoder layers that contains a SPA block and a fully connected feed-forward network (FFN) block. Both blocks are followed by a dropout layer with rate ϕ during training phase. Residual connections are employed around both blocks.

$$FFN(x) = \text{GELU}(xW_1^f + b_1^f)W_2^f + b_2^f, \tag{13}$$

where GELU [37] is an activation function. To capture correlations in multiple time spans, we keep the input query fixed to be from the latest time interval t and perform computations with the key and value from each time interval. This allows

global nodes in each time interval to be directly linked with proxy nodes in the latest period. Specifically, as demonstrated in Figure 4, T time intervals share the same CSTFE with the same parameters and the same input query P_t. The benefits of this sharing mechanism are twofold: saving model parameters and reducing the risk of overfitting to some extent. A residual connection is also employed around the encoder layers:

$$H_{t-i} = ST\,Encoder(P_t, Z_{t-i}) + Z_{t-i} \in \mathbb{R}^{N\times d}. \tag{14}$$

Without loss of the generality, the $ST\,Encoder$ could contains L encoder layers, each comprising an SPA block and an FFN block with residual connections. Finally, the outputs from T time intervals are concatenated on feature dimension into $H^{ST} = [H_{t-T+1}||\ldots||H_{t-i}||\ldots||H_t] \in \mathbb{R}^{N\times Td}$. In this module, we do not perform separate analysis on the entire historical time series of each node. Instead, features combined across time are extracted via a shared spatial encoder. This structure could also support further reduction of training time consumption through matrix parallel computation.

C. Time-Interval-Wise Prediction Module

Once we obtain the merged spatial-temporal features $H^{ST} \in \mathbb{R}^{N\times Td}$ aggregated by the encoder, to avoid quadratic complexity such as $\mathcal{O}(NTT')$ in the predictor, we project the representation H^{ST} onto a d'-dimensional space. Here, d' is a constant value. Then we further map it to prediction values at each future time step. Following our recent study [38], we incorporate the most recent input X_t at each prediction step to generate future values that reflect traffic evolution.

$$H' = \mathrm{GELU}(H^{ST}W^{de} + b^{de}) \in \mathbb{R}^{N\times d'}, \tag{15}$$

$$\hat{X}_{t+j} = X_t + H'W^{de2}_{t+j} + b^{de2}_{t+j} \in \mathbb{R}^{N\times C_{out}}, \tag{16}$$

where $W^{de} \in \mathbb{R}^{Td\times d'}, b^{de} \in \mathbb{R}^{d'}$ are parameters of the nonlinear mapping, $W^{de2}_{t+j} \in \mathbb{R}^{d'\times C_{out}}, b^{de2}_{t+j} \in \mathbb{R}^{C_{out}}$ are parameters of linear projections at each time step, d' is the hidden dimension, and C_{out} is the number of predicted feature dimensions. The final prediction is $\hat{\mathbf{Y}} = [\hat{X}_{t+1}, \ldots \hat{X}_{t+j}, \ldots, \hat{X}_{t+T'}] \in \mathbb{R}^{T'\times N\times C_{out}}$. Unlike our earlier work [38], which introduced time-aware adjacency learning using GNNs, TLAST differs in both modeling design and computational strategy. Specifically, we introduce the time-lag-aware embedding module, the SPA mechanism, and the merge of spatial-temporal features to reduce complexity while preserving spatio-temporal expressiveness.

D. Complexity Analysis

We conduct the theoretical complexity analysis of our TLAST model in this subsection. TLAST processes the input tensor $\mathbf{X} \in \mathbb{R}^{T\times N\times C}$ through a three-stage architecture:

1) Time-Lag Embedding Module: Initially, we concatenate X_{t-i} with X_t for all $i \in [0, T-1]$, followed by a linear projection to $\mathbf{X}^{cross} \in \mathbb{R}^{T\times N\times d}$, incurring $\mathcal{O}(TNCd)$ time complexity and $\mathcal{O}(Cd)$ parameter space. Subsequently, three learnable parameters are queried to generate temporal and spatial embeddings, introducing $\mathcal{O}((\mathcal{T}+7+N)d)$ parameter. Next, time-lag embeddings are computed by a shared two-layer MLP ($\mathcal{O}(Td^2)$ time, $\mathcal{O}(d^2)$ parameters). All embeddings are then broadcast-added to $\mathbf{X}^{cross}$ ($\mathcal{O}(TNd)$), followed by a local temporal convolution, contributing $\mathcal{O}(TN\tau d^2)$ time complexity and $\mathcal{O}(\tau d^2)$ parameters.

2) Cross-Time Spatial-Temporal Feature Extractor: The embedded feature representations $\mathbf{Z} \in \mathbb{R}^{T\times N\times d}$ are subsequently fed into this Transformer-like architecture. In this design, we treat only the spatial variable features as tokens and employ a shared extractor across all time intervals. By introducing Queries and Keys from different time steps, the model captures time-lagged spatial dependencies. Under a full attention mechanism (i.e., Transformer encoder layer), this design incurs a time complexity of $\mathcal{O}(LT(N^2d + Nd^2))$ and a parameter complexity of $\mathcal{O}(Ld^2)$. In contrast, when our proposed SPA module is used as the attention mechanism, the total time complexity is reduced to $\mathcal{O}(mN+LT(mNd+md^2+Nd^2))$, and the parameter complexity becomes $\mathcal{O}(mN+Ld^2)$, which includes an additional readout function with a time and space complexity of $\mathcal{O}(mN)$. By setting m as a constant, the module demonstrates linear complexity relative to the number of nodes N.

3) Time-Interval-Wise Prediction Module: Finally, the merged spatiotemporal features $H^{ST} \in \mathbb{R}^{N\times Td}$ are used to predict future values at each time step. A fully connected layer first maps these features to a shape of (N, d'), introducing a time complexity of $\mathcal{O}(NTdd')$ and a parameter complexity of $\mathcal{O}(Tdd')$. Subsequently, T' independent fully connected layers are employed to project these representations into the final future values, resulting in a time complexity of $\mathcal{O}(NT'd'C_{\text{out}})$ and a parameter complexity of $\mathcal{O}(T'd'C_{\text{out}})$.

In summary, the overall time complexity of the proposed TLAST model (with SPA attention block) is $O(TNCd + TN\tau d^2 + TNd^2 + mN + TL(mNd + md^2 + Nd^2 + d^2) + NTdd' + NT'd'C_{\text{out}})$. Assuming constant input feature dimension C, output dimension C_{out}, and setting m as a small constant, under fixed configurations of L, d, and d', the model's complexity asymptotically approaches $\mathcal{O}(TN+T'N)$. Regarding the memory and parameter requirements, the overall space complexity of the model is $\mathcal{O}((C+\mathcal{T}+N)d+(\tau+L)d^2+LTmN + Tdd' + T'd'C_{out})$, which approximates to $\mathcal{O}(\mathcal{T} + N+TN+T')$ under the same constant assumptions. It is worth noting that while the above analysis addresses the theoretical complexity, the actual computational cost during inference and training is equally critical. We report the empirical runtime and memory consumption of the model in Section V to validate its practical efficiency advantages.

V. Experiments

We conduct experiments to answer the following 5 research questions:

- RQ1: How is the prediction accuracy of TLAST compared to various baselines?
- RQ2: How is the computational efficiency of TLAST in terms of memory usage and time cost?
- RQ3: How does the number of proxy nodes in SPA affect the performance of TLAST?
- RQ4: How do core components in the model framework of TLAST benefit the model performance?

TABLE I
THE DETAILED STATISTICS OF DATASETS AND THE HYPERPARAMETER SETTINGS

Data Type		highway traffic flow				city grid inflow and outflow		
Datasets		PEMS04	PEMS07	PEMS08	CA	NYCTaxi	CHIBike	T-Drive
#Nodes N		307	883	170	8600	75 (15×5)	270 (15×18)	1024 (32×32)
#Time steps		16992	28224	17856	35040	17520	4416	3600
time span per step		5min	5min	5min	15min	30min	30min	60min
#steps per day $\mathcal{T}$		288	288	288	96	48	48	24
Missing rate		1.59%	0.45%	0.35%	2.32%	12.01%	86.33%	22.75%
hyper-parameter	m	8	8	8	2	4	2	4
	d	64	64	64	64	64	32	64
	d'	1024	1024	1024	1024	1024	16	512
	h	1	1	2	2	4	1	4
	L	1	1	1	1	3	3	3

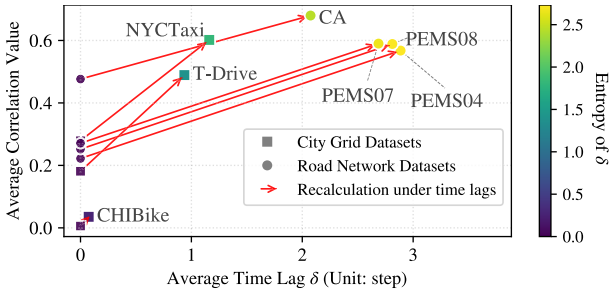


Fig. 6. Comparison of average correlations and time lag complexity across two dataset types.

- RQ5: Can SPA learn time-varying spatial correlations to represent traffic patterns?

A. Datasets and Baselines

We evaluate the performance of our TLAST on seven real-world datasets, including three small or medium sized road network datasets [1] (PEMS04, PEMS07 and PEMS08), one large road network dataset (CA [39]), and three city grid datasets (NYCTaxi [40], T-Drive [41], and CHIBike [42]). The four road network datasets record the highway traffic flow data of nodes denoting road sensors. The input and predicted feature dimensions are both 1, *i.e.*, $C = C_{out} = 1$. The other three citywide datasets contain a group of grids, each of them being a city area with inflow and outflow data. For city grid datasets, $C = C_{out} = 2$, *i.e.*, inflow and outflow. The details of seven datasets are shown in Table I.

We further conducted a statistical analysis of time-delay phenomena on the training sets of all datasets from two different categories, using the method introduced in Section III. As shown in Figure 6, we report the average values of the correlation matrices S_* and S_0, computed with and without considering time lags, respectively, along with the entropy of the corresponding optimal time lags Δ_*. $\delta_{\max}$ is set to 6 and 3 for road network datasets and city grid datasets. Results indicate that incorporating time lags consistently enhances the correlation across all datasets. Notably, road network datasets exhibit more complex and pronounced time delays compared to urban grid datasets. The lower delays observed in grid-based datasets may be attributed to the coarser granularity of spatial partitioning.

We compare TLAST with 1 naive method Historical Last (HL) [43] and 11 representative deep-learning based methods. HL simply use the last observation as all the future predictions. Deep-learning based baselines can be divided into two groups, including six graph-based methods (DCRNN [8], STGCN [9], GWNET [4], STSGCN [1], STGODE [10], STGNCDE [11]), and five self-attention based methods (GMAN [5], AST-GNN [16], PDFormer [17], STAEformer [19], TESTAM [20]). Besides, we introduce a variant of our TLAST model, denoted as TLAST(full), in which the proposed SPA module is replaced with a full attention mechanism.

B. Experiment Setting

The implementation[1] of the proposed model is under the PyTorch framework on a Linux server with one Intel(R) Xeon(R) Gold 5220 CPU and one 32GB NVIDIA Tesla V100-SXM2 GPU card. For a fair comparison, we maintain the consistency with the dataset-setting used in previous methods [17], [39]. Specifically, we split four road network datasets into the training, validation, testing sets with a ratio of 6:2:2 and 7:1:2 for the other three grid datasets. For PEMS datasets, the input length T and the predicted length T' are both 12, leading to a one hour ahead prediction (5 minutes for one time step). For CA dataset, T and T' are both 12, leading to a three hour ahead prediction (15 minutes for one time step). Following [39], one year of traffic data from 2019 is used. For three grid datasets, T is 6, and T' is 1. We employ standard normalization to standardize input features and recover them to real values for the loss calculation. We choose the Huber Loss [44] as the loss function. We use AdamW [45] as the optimizer during the model training with a learning rate of 0.001 for 100 epochs.

We use the grid search technique to set the hyper-parameters of the model. Five important hyper-parameters m, d, d', h, L are shown in Table I. The kernel size of time convolution τ in the embedding module is set to 3, and the dropout rate ϕ is 0.1. Since m is critical to the efficiency of the model implementation. The sensitivity of the TLAST model to m is further discussed in Subsection V-E later. We use three metrics to evaluate the prediction performance: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE). All missing values are excluded from the evaluation for road network datasets. Following previous methods [17], [46], values below a threshold are not included in the evaluation for grid datasets. The filter thresholds are 10, 10, and 5 for NYCTaxi, T-Drive, and CHIBike respectively. We report the average results obtained after repeating the experiments no less than five times. Besides, for a fair comparison with baselines [17], the results on grid datasets are average values of the inflow and outflow evaluation indicators.

C. Prediction Accuracy Comparison (RQ1)

Table II and Table III show the prediction accuracy comparison for average results of all predicted horizons

[1]The code is available at https://github.com/zhuoshu/TLAST

TABLE II
COMPARISON OF PREDICTION ACCURACY ON THREE ROAD NETWORK DATASETS

Method	PEMS04			PEMS07			PEMS08		
	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE
HL	31.56 ±0.00	21.39 ±0.00	47.59 ±0.00	35.45 ±0.00	15.66 ±0.00	52.57 ±0.00	25.28 ±0.00	15.62 ±0.00	37.79 ±0.00
DCRNN	20.34 ±0.07	13.98 ±0.06	32.20 ±0.13	22.45 ±0.03	9.74 ±0.02	35.41 ±0.03	15.94 ±0.10	10.19 ±0.06	25.12 ±0.12
STGCN	22.36 ±0.25	15.00 ±0.11	34.62 ±0.34	24.91 ±0.40	10.90 ±0.21	38.21 ±0.40	17.96 ±0.08	11.33 ±0.07	27.58 ±0.15
GWNET	19.04 ±0.04	13.40 ±0.13	30.43 ±0.04	20.79 ±0.09	9.00 ±0.12	33.76 ±0.09	15.11 ±0.12	10.09 ±0.65	24.02 ±0.17
STSGCN	22.30 ±0.32	15.23 ±0.59	35.18 ±0.43	25.27 ±0.24	10.92 ±0.15	40.43 ±0.48	17.95 ±0.18	11.82 ±0.24	27.55 ±0.22
STGNCDE	20.04 ±0.15	13.76 ±0.14	31.73 ±0.19	21.95 ±0.14	9.33 ±0.06	34.95 ±0.14	16.52 ±0.26	10.41 ±0.15	25.88 ±0.40
GMAN	19.67 ±0.10	14.89 ±0.71	31.05 ±0.19	21.03 ±0.15	9.50 ±0.18	33.98 ±0.28	15.65 ±0.14	12.27 ±1.29	24.55 ±0.10
ASTGNN	19.21 ±0.12	12.96 ±0.11	30.90 ±0.10	21.08 ±0.27	8.85 ±0.14	34.44 ±0.26	15.36 ±0.15	9.65 ±0.13	24.73 ±0.14
PDFormer	18.37 ±0.03	12.17 ±0.07	30.00 ±0.04	19.82 ±0.06	8.44 ±0.04	32.84 ±0.05	13.79 ±0.09	9.22 ±0.07	23.42 ±0.15
STAEformer	18.22 ±0.03	12.03 ±0.04	30.20 ±0.16	19.23 ±0.09	8.03 ±0.04	32.75 ±0.13	13.51 ±0.04	8.89 ±0.05	23.27 ±0.09
TESTAM	18.53 ±0.14	12.49 ±0.14	30.49 ±0.51	20.11 ±0.25	8.36 ±0.10	33.64 ±0.43	14.72 ±0.13	9.58 ±0.17	24.32 ±0.31
*TLAST(full)	**17.89** ±0.02	11.80 ±0.01	29.57 ±0.07	**18.94** ±0.02	**7.93** ±0.02	**31.95** ±0.04	13.16 ±0.01	8.65 ±0.02	22.63 ±0.04
*TLAST	17.90 ±0.01	**11.78** ±0.03	**29.53** ±0.02	18.97 ±0.02	**7.93** ±0.02	32.03 ±0.01	**13.13** ±0.01	**8.63** ±0.01	**22.59** ±0.03

* Results of TLAST(full) and TLAST are statistically significant compared to the best baseline (t-test with p-value<0.05).

TABLE III
COMPARISON OF PREDICTION ACCURACY ON THREE CITY GRID DATASETS

Method	CHIBike			NYCTaxi			T-Drive		
	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE
HL	5.34 ±0.00	40.91 ±0.00	7.57 ±0.00	20.95 ±0.00	21.74 ±0.00	34.56 ±0.00	38.68 ±0.00	28.23 ±0.00	73.43 ±0.00
DCRNN	4.39 ±0.02	32.69 ±0.13	6.32 ±0.03	15.55 ±0.03	16.07 ±0.02	25.18 ±0.07	24.33 ±0.57	18.34 ±0.35	42.42 ±1.20
STGCN	4.01 ±0.04	30.14 ±0.21	5.81 ±0.05	14.69 ±0.08	15.21 ±0.10	23.97 ±0.14	23.98 ±0.32	19.40 ±0.34	41.56 ±0.61
GWNET	4.58 ±0.08	33.21 ±0.73	6.73 ±0.12	17.17 ±0.17	16.44 ±0.14	29.72 ±0.37	27.93 ±0.58	20.54 ±0.49	48.25 ±1.11
STSGCN	4.12 ±0.03	31.54 ±0.24	5.87 ±0.05	14.24 ±0.10	14.40 ±0.08	23.72 ±0.24	25.27 ±1.58	19.77 ±1.06	43.35 ±3.18
STGNCDE	4.58 ±0.12	34.77 ±1.00	6.52 ±0.15	14.67 ±0.17	14.83 ±0.13	24.37 ±0.25	29.94 ±1.18	21.73 ±0.91	56.73 ±2.19
GMAN	4.01 ±0.08	30.21 ±0.43	5.78 ±0.10	13.08 ±0.16	13.86 ±0.18	21.05 ±0.26	18.79 ±0.80	16.48 ±0.91	31.23 ±1.09
ASTGNN	4.40 ±0.04	29.75 ±0.30	6.16 ±0.05	13.50 ±0.11	14.33 ±0.11	22.20 ±0.17	27.13 ±0.10	22.13 ±0.10	51.85 ±0.19
PDFormer	3.98 ±0.03	30.69 ±0.29	5.61 ±0.04	12.62 ±0.07	**12.96** ±0.03	20.78 ±0.16	17.86 ±0.32	14.69 ±0.31	31.82 ±0.43
STAEformer	4.03 ±0.04	31.07 ±0.25	5.67 ±0.06	12.40 ±0.03	13.24 ±0.04	20.47 ±0.04	17.73 ±0.12	14.89 ±0.13	31.92 ±0.25
TESTAM	3.90 ±0.07	29.71 ±0.36	5.56 ±0.14	12.89 ±0.06	13.50 ±0.04	21.67 ±0.18	19.49 ±0.21	15.33 ±0.13	34.69 ±0.42
*TLAST(full)	3.84 ±0.05	29.69 ±0.34	5.43 ±0.08	12.29 ±0.02	13.10 ±0.05	20.11 ±0.07	16.51 ±0.03	13.88 ±0.01	30.16 ±0.02
*TLAST	**3.81** ±0.03	**29.50** ±0.31	**5.36** ±0.05	**12.22** ±0.02	13.06 ±0.04	**19.99** ±0.07	**16.34** ±0.09	**13.79** ±0.05	**29.88** ±0.23

* Results of TLAST(full) and TLAST are statistically significant compared to the best baseline (t-test with p-value<0.05).

on PEMS datasets and grid datasets. The best results are shown in bold and the second best results are underlined. On PEMS datasets, our TLAST consistently outperforms all baselines. Compared to the state-of-art baseline, TLAST achieves an average improvement of 1.99%/2.06%/2.44% in terms of MAE/MAPE/RMSE. For grid datasets, TLAST achieves an average improvement of 3.86%/4.14% in terms of MAE/RMSE. On NYCTaxi dataset, TLAST shows slightly higher MAPE than PDFormer but outperforms the other baselines. This is primarily due to a noticeable distribution shift in the NYCTaxi dataset, where the test set values are more concentrated in the lower range. As a result, the overall MAPE of our model increases, while MAE and RMSE remain relatively low. This observation highlights the importance of developing models that are robust to distribution shifts, which is a key direction for future research in traffic prediction.

Table IV showcases the results on the largest dataset CA (8600 nodes), encompassing three specific horizons (3, 6, and 12) and the average values across all predicted horizons. On this dataset, many recent advanced methods encounter challenges in training within acceptable time cost on devices with limited space. This is attributed to their higher complexity and intricate structures. In comparison to the second best results, our approach TLAST achieves an improvements of 12.64%/21.13%/16.88% for MAE/MAPE/RMSE on the average metric.

For DCRNN and STGCN, using the predefined and fixed adjacency matrices hinders them to obtain good prediction performance. Other four graph based methods explicitly introduce auxiliary graph structures to facilitate traffic forecasting, yet these graphs fail to reflect connections across time. By contrast, our proposed model TLAST does not rely on predefined adjacency matrix but can capture the spatial correlation among distant nodes across arbitrary time spans. Among the attention-based methods, TESTAM incorporates a MoE structure to capture dynamic and latent patterns in spatial dependencies. However, its design remains confined to a single time slice, limiting its ability to model temporal dynamics and resulting in suboptimal performance. PDFormer employs the self-attention mechanism in both space and time dimensions, and the introduced delay-aware feature transformation module and two attention masks enable it to achieve better results. STAEformer introduces adaptive spatial-temporal embeddings and applies transformer-based

TABLE IV
COMPARISON OF PREDICTION ACCURACY ON THE LARGE DATASET CA

Dataset	Method	Horizon 3			Horizon 6			Horizon 12			Average		
		MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE
CA	HL	30.71	20.40	46.94	51.55	37.18	76.44	89.32	76.79	125.68	54.10	41.61	78.97
	DCRNN	17.52	12.55	28.18	21.72	16.56	34.19	28.45	23.57	44.23	21.81	16.92	34.35
	STGCN	19.14	14.23	32.64	21.65	16.09	36.94	24.86	19.14	42.61	21.48	16.16	36.69
	GWNET	16.93	13.14	27.53	21.08	16.73	33.52	27.37	22.50	42.65	21.08	16.86	33.43
	STGODE	17.59	13.28	31.04	20.92	16.23	36.65	25.34	20.56	45.10	20.72	16.19	36.65
	*TLAST	**15.29** ±0.06	**10.32** ±0.00	**25.75** ±0.06	**18.24** ±0.13	**12.73** ±0.00	**30.58** ±0.13	**22.20** ±0.24	**16.47** ±0.00	**37.55** ±0.26	**18.10** ±0.14	**12.77** ±0.12	**30.46** ±0.13

* Results of TLAST model is statistically significant compared to the best baseline (t-test with p-value <0.05). Results of baseline methods are derived from [39], the absence of other advanced baselines is due to the out-of-memory issues.

TABLE V
COMPARISON OF COMPUTATIONAL COSTS AND EFFECTIVENESS ON THREE TRAFFIC FLOW FORECASTING SCENARIOS

Scenario ($T \rightarrow T'$)	Model	Time complexity w.r.t. N, T, T'	GPU Usage (MB)	Training time (second/epoch)	Inference time (second)	MAE
1 hour ahead prediction on T-Drive ($6 \rightarrow 1$) batch size: 16	ASTGNN	$\mathcal{O}(TN + NT^2 + TN^2 + NT'^2)$	9373	29.87	2.73	18.80
	PDFormer	$\mathcal{O}(TN + TN^2 + NT^2 + NTT')$	16599	93.02	2.08	17.83
	STAEformer	$\mathcal{O}(TN + NT^2 + TN^2 + NTT')$	15006	47.74	2.13	17.73
	TLAST	$\mathcal{O}(TN + T'N)$	**2220** (↓85.21%)	**11.87** (↓75.14%)	**0.82** (↓61.50%)	**16.34** (↓7.84%)
3 hour ahead prediction on PEMS08 ($36 \rightarrow 36$) batch size: 16	PDFormer	$\mathcal{O}(TN + TN^2 + NT^2 + NTT')$	10892	216.45	19.99	15.71
	SSTBAN	$\mathcal{O}(TN + T'N + NTT')$	22519	516.90	36.76	16.50
	STAEformer	$\mathcal{O}(TN + NT^2 + TN^2 + NTT')$	6544	121.29	11.96	15.39
	TLAST	$\mathcal{O}(TN + T'N)$	**1364** (↓79.16%)	**48.93** (↓59.66%)	**5.14** (↓57.02%)	**15.34** (↓0.32%)
3 hour ahead prediction on CA ($12 \rightarrow 12$) batch size: 8	DCRNN	$\mathcal{O}((T + T')\lvert\mathcal{E}\rvert)$	21269	5501.75	952.03	21.81
	GWNET	$\mathcal{O}(TN^2 + NT + NT')$	14179	5911.67	830.63	21.08
	STGODE	$\mathcal{O}(TN^2 + NT^2 + NTT')$	30729	5764.61	923.74	20.72
	TLAST	$\mathcal{O}(TN + T'N)$	**7436** (↓75.80%)	**717.01** (↓87.56%)	**85.56** (↓90.74%)	**18.10** (↓12.64%)

computations along both temporal and spatial dimensions, achieving the best results on all datasets except CHIBike. This suggests that adaptive embeddings spanning both time and space can partially capture time-delay-related patterns.

In contrast to these models, our approach TLAST introduces cross-time mechanisms in both the embedding stage and the spatial dependency modeling stage. This design enables the model to extract spatial features associated with varying time delays more effectively, leading to superior prediction performance. Compared to its variant TLAST(full), TLAST achieves comparable results on road network datasets. TLAST(full) shows a 0.25% improvement in RMSE on PEMS07 but performs slightly worse on PEMS04 and PEMS08. On the grid-based datasets, TLAST(full) consistently underperforms compared to TLAST. This may be due to the relatively weak delay effects in these datasets, where modeling dependencies across all spatial points introduces noise and leads to overfitting. These show that TLAST with SPA achieves accuracy on par with full attention, while enabling a more efficient design.

D. Efficiency Study (RQ2)

We conduct two groups of experiments to assess the scalability of TLAST in terms of computational efficiency. The computational costs include GPU memory consumption, training time per epoch, and inference time on the entire test set.

1) Prediction Efficiency: In the first group of experiments, we compare the efficiency of TLAST with baseline models on three scenarios: the short-term forecasting on medium dataset (T-Drive), the long-term forecasting on small dataset (PEMS08), and the long-term forecasting on large dataset (CA). In the comparison on PEMS08 dataset, we introduce a baseline method SSTBAN [18] that have superior prediction accuracy on long-term forecasting and has a linear computation complexity with respect to the number of nodes. Table V presents a comparison of the training costs. Results demonstrate that TLAST attains significant improvements in computational consumption compared to the second-best method while simultaneously achieving superior prediction accuracy. On the T-Drive dataset, the reduction in the three cost metrics reaches 85.21%, 75.14%, and 61.50%, respectively. Furthermore, Figure 7 illustrates the convergence curves. TLAST and STAEformer achieve low validation set error in over twenty epochs. This highlights the superior convergence speed of embedding-based methods. On the other hand, unlike methods requiring precomputed data like clustering [17] or

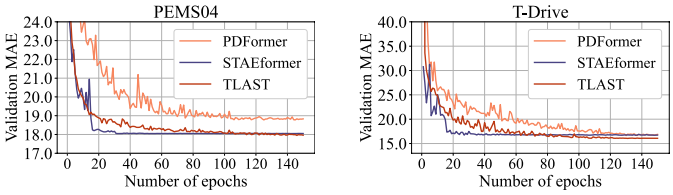


Fig. 7. Comparison of convergence curves on PEMS04 and T-Drive datasets.

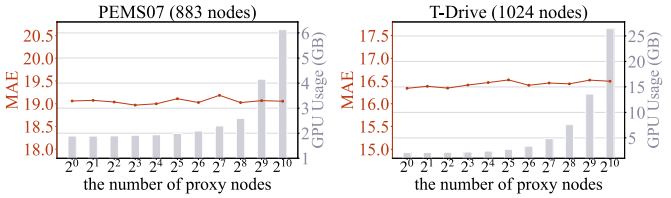
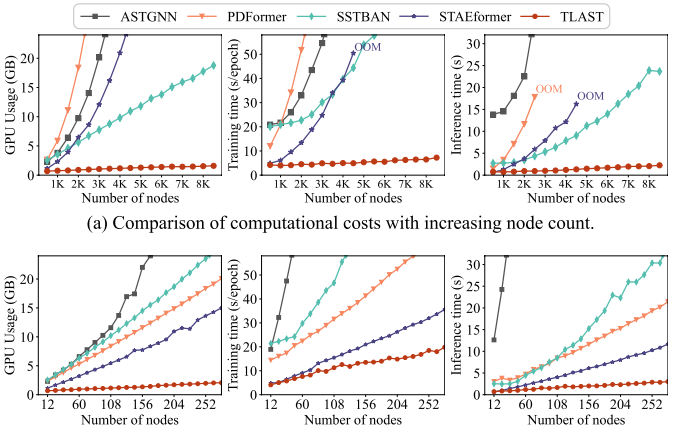


Fig. 9. The impact of varying numbers of proxy nodes on model performance.



(a) Comparison of computational costs with increasing node count.

(b) Comparison of computational costs with increasing time-step count.

Fig. 8. Comparison of computational costs with increasing node count and time step length (OOM: out-of-memory).

DTW matrices [10] for training, our approach avoids such additional overhead, enabling more flexible data usage.

2) Scalability With Respect to the Number of Nodes and Time Steps: In the second group of experiments, we simulate an increase in the number of nodes and the sequence length by extracting a subset from dataset CA, respectively. For instance, experiments are conducted using the data from the first 1000 nodes out of a total of 8600. We focus on comparing the computational efficiency of each method under these conditions. To facilitate a clear comparison, we standardized the parameters for this experiment, fixing both input and prediction time steps at 12. Additionally, the number of samples for training, validation, and testing is set at 100, with a batch size of 1. For the group of sequence length, we fix the number of nodes as 500 and increase the input and predict length gradually.

Figure 8(a) shows the computational cost of models as the number of nodes increases. It can be observed that two state-of-the-art methods PDFomer and STAEformer encountered out-of-memory (OOM) errors, rendering them unable to continue running when the number of nodes exceeded 2500 and 4500, respectively. In contrast, SSTBAN and TLAST demonstrate lower computational overhead due to their linear algorithmic complexity. As the number of nodes grows, the computational consumption of TLAST continues to exhibit a steady linear growth trend, while computational costs consistently remain at relatively low levels. Figure 8(b) presents the performance as the sequence length increases. In contrast to baseline methods, our model demonstrates a smooth linear growth trend in computational consumption as both input and target time steps increase, all while maintaining a notably low level. These results emphasize the favourable scalability of TLAST, thereby offering a potential solution for prediction tasks involving large sets of sensors or finer grid partitions in real-world applications.

E. Hyperparameter Sensitivity Study (RQ3)

As an important component of our proposed TLAST model, the SPA aims to achieve more efficient feature extraction by reducing the query-key pairs needed to be computed. Therefore, the number of proxy nodes m, as the length of the input query, plays a vital role in the model efficiency. To investigate the sensitivity of TLAST to m, we conduct a group of experiments setting exponentially increasing values of m. The prediction accuracy and GPU memory usage in training are monitored. Figure 9 presents the experimental results on the T-Drive and PEMS07 datasets, and similar results are observed on other datasets as well.

We observe that the model does not exhibit much sensitivity to m. Despite the exponential increase in m values, the MAE curves do not demonstrate a noticeable upward or downward trend. In other words, increasing the number of proxy nodes m, even up to the original number of nodes, does not necessarily improve performance; however, it does indeed increase computational costs. This aligns with our initial expectations that a small m is adequate for summarizing node representation without sacrificing performance, while also providing efficiency gains. This also serves as a reminder that the spatial-temporal correlations among traffic nodes can be adequately captured and analyzed in a lower-dimensional space. In our experiments on seven datasets, m is set to be 2, 4, 8 respectively as in Table I.

Additionally, the hidden dimension d' used to map the spatio-temporal features H^{ST} in the prediction module is determined based on performance on the validation set. On datasets with different number of nodes, d' reaches a bound of 1024. However, in tasks with longer prediction horizons, a larger d' may be required to construct a more expressive spatio-temporal latent space.

F. Ablation Study (RQ4)

To verify the effectiveness of three modules in TLAST, we design two groups of ablation study. The first group includes five variants: **(1) w/o Temporal Embedding**: It removes the temporal embedding. **(2) w/o Spatial Embedding**: It removes the spatial embedding. **(3) w/o ST-Encoder**: It removes the entire CSTFE module. **(4) w/o Cross-time Setting**: It removes three parts: cross-time connections in the embedding module, the time-lag embedding block,

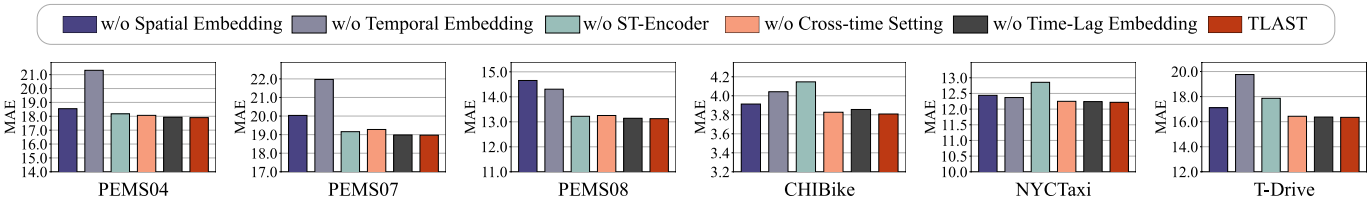


Fig. 10. Performance comparison on ablation experiments.

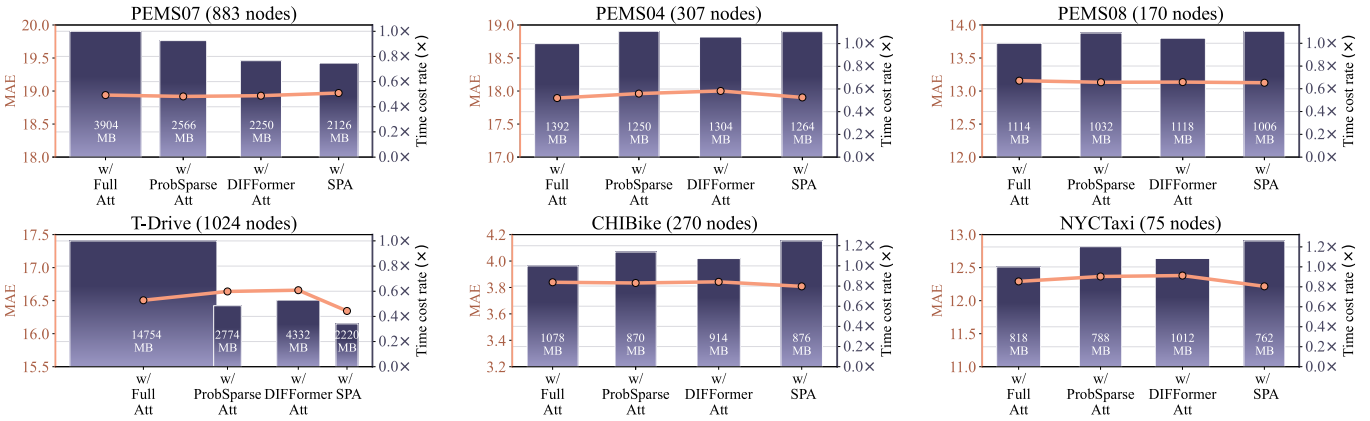


Fig. 11. Performance comparison on four attention mechanisms (The width of the dark-purple bins denotes memory usage and the height denotes comparative training time). The batch size is 16.

and the readout function in SPA is learned from the whole $\mathbf{Z}$ rather than Z_t. **(5) w/o Time-Lag Embedding**: It removes the time-lag embedding block but remains other cross-time setting. Results are shown in Figure 10. We can obtain the following observations: **First, the absence of spatial or temporal embedding causes significant performance degradation.** Two embeddings jointly enhance the model's ability to distinguish spatiotemporal semantics, which is particularly crucial for road network datasets with high time-lag complexity. The role of embeddings is consistent with the findings in STAEformer. **Second, the CSTFE module contributes notably to prediction accuracy.** Removing it forces the model to rely solely on embeddings for spatial-temporal feature extraction, leading to performance degradation, particularly on grid-based datasets. This may be because road networks allow time-lag patterns to be more easily encoded via embeddings, while grid datasets—with shorter delays and lower complexity—require more direct modeling of spatio-temporal dependencies. **Third, both the cross-time setting and time-lag embedding improve prediction accuracy**. Removing all cross-time components reduces the encoder to a standard Transformer relying only on spatial-temporal embeddings, leading to performance drops. While the time-lag embedding block also enhances performance, its impact varies depending on the dataset's delay characteristics. Nevertheless, the model maintains competitive results, likely due to the mixed mapping of spatio-temporal features in the time-interval-wise prediction module.

The second group of ablation studies compares TLAST (*i.e*, w/ SPA) against three attention variants: **(1)w/ Full Att**: It replaces the SPA block with a full attention block, which has a complexity of $\mathcal{O}(N^2)$. **(2) w/ ProbSparse Att**: It replaces the SPA block with a *ProbSparse* attention block from Informer [33] which exhibits $\mathcal{O}(N \log N)$ complexity. **(3) w/ DIFFormer Att**: It replaces the SPA block with a simple diffusivity model from DIFFormer [32], a recent advanced method demonstrating a linear complexity of $\mathcal{O}(N)$. Figure 11 presents the performance comparison of four attention variants. The width of bins denotes the amount of GPU memory usage, while the height represents comparative training time costs. Therefore, the area size of the bins intuitively represents the temporal and spatial consumption required by models. Results of this variants experiment reveal that on road network datasets, our TLAST model performs comparably to the state-of-the-art linear Transformer. On grid datasets, our TLAST model outperforms all variants. A possible reason is that the proxy mechanism reduces the number of node interactions that need to be learned, which helps mitigate overfitting, especially in datasets with higher missing rates. Furthermore, when the number of nodes increases significantly beyond the levels of other hyperparameters, the model's consumption can be observed to be greatly reduced.

G. Case Study (RQ5)

In this section, we present a case study to demonstrate the practical effectiveness of our model's predictions and the ability of the proposed SPA module to capture dynamic dependencies. First, we show the actual prediction results of the model on Node #117 from PEMS08 dataset, as illustrated in Figure 12(a). The predicted curve closely follows the ground truth, especially in segments where the original values exhibit minor fluctuations. This trend-fitting ability offers valuable

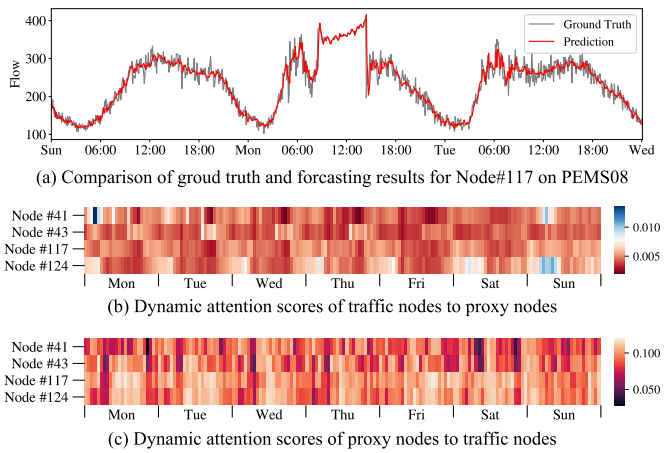


Fig. 12. Practical forecasting results and dynamic attention scores of the proposed TLAST model.

insights for practitioners analyzing traffic patterns in real-world applications.

In addition, Figures 12(b) and 12(c) visualize the dynamic variations of two attention matrices (A_1 and A_2) learned by our SPA module on the test set of PEMS08. Specifically, we examine one week of data for two adjacent nodes (41 and 43) and two distant nodes (124 and 117). For each node, the two attention scores with respect to the learned proxy nodes (from A_1 and A_2) are aggregated into 7×24-dimensional vectors, representing the attention scores across 24 hours over 7 days. These visualizations lead to the following observations:

(1) The degree of correlation between the four traffic nodes and proxy nodes exhibits periodic variations over time. It reveals latent semantics of cyclic traffic patterns, such as repetitive distributions distinguishing between weekdays and weekends. (2) Distinct attention patterns emerge for different traffic nodes at different times, highlighting the heterogeneity among nodes. For example, node 43 exhibits a higher contribution on weekday evening rush hours and a lower contribution during daytime. This contrast emphasizes the effective discriminatory capability of TLAST. (3) The model is capable of identifying nodes with similar spatial-temporal patterns from traffic data itself, with no need for time-consuming pre-analytical operations such as clustering.

H. Robust Analysis

To investigate the robustness of TLAST, we evaluate its prediction performance under missing data conditions in this subsection. Specifically, we introduce random missing values into the training set of the PEMS08 dataset and compare the prediction performance of TLAST with the two state-of-the-art baseline methods, PDFormer and STAEformer, under the same missing data settings. As shown in Figure 13, at lower missing rates ($\leq$40%), PDFormer's performance degrades rapidly as the missing rate increases, while STAEformer and TLAST demonstrate similar levels of robustness, with TLAST achieving the best prediction performance. Under high missing rates ($\geq$50%), all models experience significant performance degradation. However, TLAST maintains lower prediction errors than the other two, indicating superior robustness.

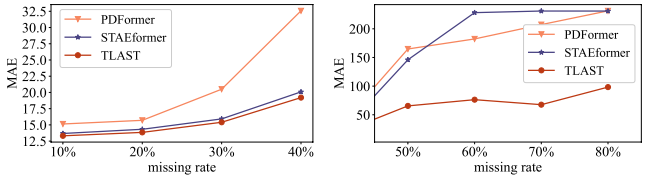


Fig. 13. Comparison of prediction performance under increasing missing rate in the training set.

I. Discussion

To provide deeper insights beyond empirical results, we organize the discussion into four aspects: the contribution of time-lag analysis to traffic modeling, the key factors underlying the improvements in prediction accuracy and computational efficiency, the limitations of TLAST, and practical implications for real-world deployment.

1) The Time-Lag Analysis Could Benefit Traffic Forecasting Research: By calculating the optimal time lag and the associated complexity for each node pair, an inherent inductive bias embedded in the spatial-temporal structure of traffic data can be uncovered. This bias reflects the latent dynamic spatial relationships and it can be a guiding principle for model design. For traffic data characterized by short time lags and low time-lag complexity (*e.g.*, city grid data), it is more important to model the spatial-temporal interactions between nodes than learned embeddings. Conversely, for traffic data with longer time lags and higher complexity (*e.g.*, road network data), effective spatial and temporal embeddings become essential to maintain the model's expressiveness.

2) The Key Factors Contributing to the Accuracy and Efficiency Gains of TLAST: TLAST achieves better performance by explicitly incorporating a direct link between different time intervals to capture the time-lag aware spatial-temporal features in the embedding stage and the attention extraction. Through the cross-time data embedding and the time-lag embedding, TLAST is adaptively learn discriminative patterns associated with different delays. Through the interaction between spatial Queries and Keys from different time step, TLAST avoids redundant sliding-window computations and captures essential patterns under varying delays.

Besides, the proposed SPA module further improves efficiency by introducing a proxy mechanism that reduces node interactions, lowering time complexity and helping to prevent overfitting in datasets with weak time-lag dependencies. Finally, the prediction module fuses multi-delay spatial-temporal features and performs step-wise forecasting, effectively leveraging time-lag-aware representations to enhance accuracy.

3) Current Imitations of TLAST: While TLAST demonstrates strong performance in short-term forecasting, its ability to handle distribution shifts remains limited—an issue shared by many traffic prediction models. In real-world applications, traffic conditions can vary significantly across time and regions, requiring models to be more adaptive to dynamic environments. Moreover, the current evaluation primarily focuses on short-term prediction. As shown in the second row of Table V, the performance improvement becomes less

pronounced in long-term forecasting scenarios. This suggests that further investigation is needed to understand how time-lag modeling interacts with long-range temporal dependencies and to explore strategies for enhancing model generalization under extended horizons.

4) The Practical Insights for Real-World Intelligent Transportation Systems by TLAST: The lightweight and time-lag-aware design of TLAST makes it well-suited for real-world intelligent transportation systems. In scenarios such as adaptive signal control or real-time congestion monitoring, its linear complexity and low memory footprint enable efficient deployment on edge devices. By explicitly modeling time-lagged spatial correlation, TLAST achieves better accuracy with less computational cost, supporting proactive traffic management.

Notably, TLAST retains the original Transformer architecture, achieving linear complexity by reducing input token length rather than modifying the attention mechanism. This ensures compatibility with existing hardware optimizations, allowing for faster and more scalable inference. Moreover, the SPA module's flexible readout mechanism enables targeted analysis of key traffic nodes (*e.g.*, critical intersections), enhancing the model's interpretability and practical utility.

VI. Conclusion

In this work, we proposed a Time-Lag Aware Spatial-temporal Transformer (TLAST) model for traffic flow forecasting. Inspired by empirical findings that spatial correlations among traffic nodes can strengthen under specific time lags, TLAST explicitly integrates cross-time structures into both the embedding and attention modules. This design allows the model to capture time-lag-aware spatial-temporal features more effectively. Furthermore, we introduce the SPA module to model dynamic spatial dependencies with linear complexity.

Extensive experiments on seven real-world datasets demonstrate that TLAST achieves superior prediction accuracy while significantly reducing computational overhead, underscoring its practicality for real-time intelligent transportation systems. Empirical analysis further shows that the cross-temporal design and the SPA module enable efficient learning of time-varying spatial correlations. Despite its effectiveness, TLAST currently focuses on short-term forecasting and faces challenges under distribution shifts. Future work will aim to enhance its robustness to such shifts and investigate the integration of time-lag modeling into long-term prediction tasks.

References

[1] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial–temporal synchronous graph convolutional networks: A new framework for spatial–temporal network data forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 914–921.

[2] B. Yu, Y. Lee, and K. Sohn, "Forecasting road traffic speeds by considering area-wide spatio-temporal dependencies based on a graph convolutional neural network (GCN)," *Transp. Res. C, Emerg. Technol.*, vol. 114, pp. 189–204, May 2020.

[3] M. Azaria and D. Hertz, "Time delay estimation by generalized cross correlation methods," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-32, no. 2, pp. 280–285, Apr. 1984.

[4] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for deep spatial–temporal graph modeling," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Jul. 2019, pp. 1907–1913.

[5] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A graph multi-attention network for traffic prediction," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 1, pp. 1234–1241.

[6] E. Zivot and J. Wang, "Vector autoregressive models for multivariate time series," in *Modeling Financial Time Series With S-PLUS*. New York, NY, USA: Springer, 2003, pp. 369–413, doi: 10.1007/978-0-387-21763-5_11.

[7] G. Ristanoski, W. Liu, and J. Bailey, "Time series forecasting using distribution enhanced linear regression," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining (PKADD)*, Jan. 2013, pp. 484–495.

[8] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–10.

[9] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2018, pp. 3634–3640.

[10] Z. Fang, Q. Long, G. Song, and K. Xie, "Spatial–temporal graph ODE networks for traffic flow forecasting," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Min.*, 2021, pp. 364–373.

[11] J. Choi, H. Choi, J. Hwang, and N. Park, "Graph neural controlled differential equations for traffic forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 6, 2022, pp. 6367–6374.

[12] M. Li and Z. Zhu, "Spatial–temporal fusion graph neural networks for traffic flow forecasting," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 4189–4196.

[13] Q. Zheng and Y. Zhang, "DSTAGCN: Dynamic spatial–temporal adjacent graph convolutional network for traffic forecasting," *IEEE Trans. Big Data*, vol. 9, no. 1, pp. 241–253, Feb. 2023.

[14] G. Jin et al., "Spatio-temporal graph neural networks for predictive learning in urban computing: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 10, pp. 5388–5408, Oct. 2024.

[15] H. Yan, X. Ma, and Z. Pu, "Learning dynamic and hierarchical traffic spatiotemporal features with transformer," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 22386–22399, Nov. 2022.

[16] S. Guo, Y. Lin, H. Wan, X. Li, and G. Cong, "Learning dynamics and heterogeneity of spatial–temporal graph data for traffic forecasting," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 11, pp. 5415–5428, Nov. 2022.

[17] J. Jiang, C. Han, W. X. Zhao, and J. Wang, "PDFormer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 4365–4373.

[18] S. Guo et al., "Self-supervised spatial–temporal bottleneck attentive network for efficient long-term traffic forecasting," in *Proc. IEEE 39th Int. Conf. Data Eng. (ICDE)*, Apr. 2023, pp. 1585–1596.

[19] H. Liu et al., "Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting," in *Proc. 32nd ACM Int. Conf. Inf. Knowl. Manage.*, 2023, pp. 4125–4129.

[20] H. Lee and S. Ko, "TESTAM: A time-enhanced spatio-temporal attention model with mixture of experts," in *Proc. Int. Conf. Learn. Represent.*, Oct. 2024, pp. 1–12.

[21] J. Ji et al., "Spatio-temporal self-supervised learning for traffic flow prediction," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, 2023, pp. 4356–4364.

[22] Z. Diao et al., "DMSTG: Dynamic multiview spatio-temporal networks for traffic forecasting," *IEEE Trans. Mobile Comput.*, vol. 23, no. 6, pp. 6865–6880, Jun. 2024.

[23] T. Dan, X. Pan, B. Zheng, and X. Meng, "ByGCN: Spatial temporal byroad-aware graph convolution network for traffic flow prediction in road networks," in *Proc. 33rd ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2024, pp. 415–424.

[24] Y. Zhou et al., "Make graph neural networks great again: A generic integration paradigm of topology-free patterns for traffic speed prediction," in *Proc. 33rd Int. Joint Conf. Artif. Intell.*, Aug. 2024, pp. 2607–2615.

[25] Q. Yu, W. Ding, H. Zhang, Y. Yang, and T. Zhang, "Rethinking attention mechanism for spatio-temporal modeling: A decoupling perspective in traffic flow prediction," in *Proc. 33rd ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2024, pp. 3032–3041.

[26] K. Sun, P. Liu, P. Li, and Z. Liao, "ModWaveMLP: MLP-based mode decomposition and wavelet denoising model to defeat complex structures in traffic forecasting," in *Proc. AAAI Conf. Artif. Intell.*, Mar. 2024, vol. 38, no. 8, pp. 9035–9043.

[27] W. Kong, Z. Guo, and Y. Liu, "Spatio-temporal pivotal graph neural networks for traffic flow forecasting," in *Proc. AAAI Conf. Artif. Intell.*, Mar. 2024, vol. 38, no. 8, pp. 8627–8635.
[28] Z. Shao, Z. Zhang, F. Wang, W. Wei, and Y. Xu, "Spatial–temporal identity: A simple yet effective baseline for multivariate time series forecasting," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA, Oct. 2022, pp. 4454–4458.
[29] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 1–11.
[30] K. Choromanski et al., "Rethinking attention with performers," in *Proc. 9th Int. Conf. Learn. Represent.*, Virtual Event, Austria, Jan. 2020, pp. 1–14.
[31] Q. Wu, W. Zhao, Z. Li, D. Wipf, and J. Yan, "NodeFormer: A scalable graph structure learning transformer for node classification," in *Proc. Adv. Neural Inf. Process. Syst.*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., Jan. 2023, pp. 27387–27401.
[32] Q. Wu, C. Yang, W. Zhao, Y. He, D. Wipf, and J. Yan, "DIFFormer: Scalable (Graph) transformers induced by energy constrained diffusion," in *Proc. 11th Int. Conf. Learn. Represent.*, Kigali, Rwanda, Jan. 2023, pp. 1–13.
[33] H. Zhou et al., "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, May 2021, pp. 11106–11115.
[34] S. Liu et al., "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *Proc. 10th Int. Conf. Learn. Represent.*, 2022, pp. 1–11.
[35] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *Proc. 11th Int. Conf. Learn. Represent.*, Kigali, Rwanda, Jan. 2022, pp. 1–12.
[36] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *Proc. 11th Int. Conf. Learn. Represent.*, Kigali, Rwanda, 2023, pp. 1–12.
[37] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.
[38] Q. Zheng and Y. Zhang, "TAGnn: Time adjoint graph neural network for traffic forecasting," in *Proc. 28th Int. Conf. Database Syst. Adv. Appl.*, Tianjin, China, Jan. 2023, pp. 369–379.
[39] X. Liu et al., "LargeST: A benchmark dataset for large-scale traffic forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2023, pp. 75354–75371.
[40] L. Liu et al., "Dynamic spatial–temporal representation learning for traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 7169–7183, Nov. 2021.
[41] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, "Urban traffic prediction from spatio-temporal data using deep meta learning," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1720–1730.
[42] J. Wang, J. Jiang, W. Jiang, C. Li, and W. X. Zhao, "LibCity: An open library for traffic prediction," in *Proc. 29th Int. Conf. Adv. Geographic Inf. Syst.*, 2021, pp. 145–148.
[43] Y. Liang et al., "Revisiting convolutional neural networks for city-wide crowd flow analytics," in *Proc. Mach. Learn. Knowl. Discovery Databases*, Jan. 2021, pp. 578–594.
[44] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in Statistics*. New York, NY, USA: Springer, 1992, pp. 492–518.
[45] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in Adam," 2017, *arXiv:1711.05101*.
[46] H. Yao et al., "Deep multi-view spatial–temporal network for taxi demand prediction," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, Apr. 2018, pp. 2588–2595.

Qi Zheng received the B.S. and M.S. degrees in computer science from Tongji University, Shanghai, China, in 2019 and 2022, respectively, where he is currently pursuing the Ph.D. degree in computer science with the Key Laboratory of Embedded System and Service Computing. His research interests include intelligent transportation systems and spatial–temporal data mining.

Minhua Shao received the B.S. and Ph.D. degrees in traffic engineering from Tongji University, Shanghai, China, in 2001 and 2006, respectively. She is currently a Professor with the School of Transportation, Tongji University. Her research interests include traffic flow theory, intelligent transportation systems, and traffic network analysis.

Yaying Zhang (Member, IEEE) received the B.S. degree in computer science and the M.S. degree in electrical engineering from Shandong University of Science and Technology, Shandong, China, in 1996 and 1999, respectively, and the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2004. She is currently a Professor with the Key Laboratory of Embedded System and Service Computing, Tongji University, Shanghai. Her research interests include intelligent transportation systems, data integrity, and data mining.