Learning All Dynamics: Traffic Forecasting via Locality-Aware Spatio-Temporal Joint Transformer

Yuchen Fang, Fang Zhao, Yanjun Qin, Haiyong Luo, *Member, IEEE*, and Chenxing Wang, *Graduate Student Member, IEEE*

***Abstract*— Forecasting traffic flow and speed in the urban is important for many applications, ranging from the intelligent navigation of map applications to congestion relief of city management systems. Therefore, mining the complex spatio-temporal correlations in the traffic data to accurately predict traffic is essential for the community. However, previous studies that combined the graph convolution network or self-attention mechanism with deep time series models (*e.g.*, the recurrent neural network) can only capture spatial dependencies in each time slot and temporal dependencies in each sensor, ignoring the spatial and temporal correlations across different time slots and sensors. Besides, the state-of-the-art Transformer architecture used in previous methods is insensitive to local spatio-temporal contexts, which is hard to suit with traffic forecasting. To solve the above two issues, we propose a novel deep learning model for traffic forecasting, named Locality-aware spatio-temporal joint Transformer (Lastjormer), which elaborately designs a spatio-temporal joint attention in the Transformer architecture to capture all dynamic dependencies in the traffic data. Specifically, our model utilizes the dot-product self-attention on sensors across many time slots to extract correlations among them and introduces the linear and convolution self-attention mechanism to reduce the computation needs and incorporate local spatio-temporal information. Experiments on three real-world traffic datasets, England, METR-LA, and PEMS-BAY, demonstrate that our Lastjormer achieves state-of-the-art performances on a variety of challenging traffic forecasting benchmarks.**

***Index Terms*— Traffic forecasting, spatio-temporal joint transformer, diffusion convolution network.**

I. Introduction

THE gradually increasing of vehicles and populations has brought many challenges for city management, such as annoying traffic jams and life-threatening traffic accidents. The accurate and efficient forecasting of the traffic flow and speed on the road networks can plan routes so that people can avoid traffic congestion and accidents. Therefore, traffic forecasting has attracted a lot of attention in the industrial and academic communities and many works have been proposed.

Most early methods treat traffic forecasting as the univariate time series forecasting that captures only the temporal dependence and ignores the spatial correlations in the traffic data, such as statistical method ARIMA [1], shallow machine learning method SVR [2], and deep time series model LSTM [3]. The subsequent works DCRNN [4] and STGCN [5] notice the shortcoming of the above methods and introduce the graph convolution network (GCN) [6] into traffic forecasting to capture the spatial dependence in the traffic data. Specifically, they combine GCN with deep time series models (*e.g.*, recurrent neural network (RNN) [7] and temporal convolution network (TCN) [8]) to separately extract spatio-temporal dependencies. However, the performance of DCRNN and STGCN is restricted by the local spatio-temporal representation power and the pre-defined static spatial correlations. Therefore, Graph WaveNet [9] and MTGNN [10] utilize the adaptive graph in GCN to capture the global spatial dependence and avoid bias in prior knowledge. Consequently, the state-of-the-art methods [11], [12], [13], [14] further replace the GCN and TCN in previous models with the self-attention [15] to dynamically reveal the global spatio-temporal correlations in the traffic data. While having proven to be effective for forecasting traffic, these approaches encounter two major limitations.

First, current methods still follow the paradigm of separately capturing the spatio-temporal dependencies and thus are insufficient to reveal all dynamic correlations in the traffic data. As shown in Fig. 1, a sensor on the road network is influenced by its historical records (*i.e.*, purple arrows), other sensors at the same time (*i.e.*, red and blue arrows), and sensors with cross-time spatial correlations and cross-sensor temporal correlations (*i.e.*, green and yellow arrows). Unfortunately, previous methods fail to capture the cross-time and cross-sensor dependencies in the traffic data. Second, few methods are suited to the traffic data, which contains both the local and global spatio-temporal dependencies. For the temporal dimension, traffic is affected by short-term events and the long-term trend simultaneously. For the spatial dimension, traffic is influenced by the neighbor sensors (*e.g.*, red arrows in Fig. 1) and the far away but have similar function sensors (*e.g.*, blue arrows in Fig. 1) concurrently. However, most convolution and recurrent networks fail to gain the global knowledge,

Manuscript received 18 June 2021; revised 25 March 2022 and 7 June 2022; accepted 20 July 2022. Date of publication 16 August 2022; date of current version 5 December 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61872046, in part by the Beijing Natural Science Foundation under Grant 4212024, and in part by the Science and Technology Plan Project of Inner Mongolia Autonomous Region under Grant 2019GG328. The Associate Editor for this article was X. Ma. *(Corresponding authors: Fang Zhao; Haiyong Luo.)*

Yuchen Fang, Fang Zhao, Yanjun Qin, and Chenxing Wang are with the School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: fangyuchen@bupt.edu.cn; zfsse@bupt.edu.cn; qinyanjun@bupt.edu.cn; wangchenxing@bupt.edu.cn).

Haiyong Luo is with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China (e-mail: yhluo@ict.ac.cn).

Digital Object Identifier 10.1109/TITS.2022.3197640

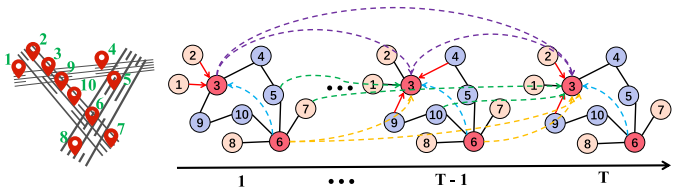


Fig. 1. The left picture shows a road network with ten sensors numbered from 1 to 10. The right picture denotes the intricate correlations between sensor 3 and other sensors in the traffic network over time from 1 to T. The color of a sensor in the right picture represents the degree of the sensor on the road network. Besides, in the right picture, the black lines denote the road network, red arrows denote correlations of neighbors at the same time, blue dotted arrows denote implicit spatial correlations between sensors with a similar function, green dotted arrows denote the cross-time correlations between remote sensors, purple and yellow dotted arrows denote the explicit and implicit temporal dependencies from the history of sensor 3 and sensors with similar temporal trends.

and the self-attention in the Transformer [15] architecture is insensitive to local spatio-temporal contexts.

To alleviate the above limitations, in this paper, we propose a novel Locality-aware spatio-temporal joint Transformer (Lastjormer) network for traffic forecasting, which is a Transformer-based encoder-decoder architecture. Compared with widely used convolution-based methods in traffic forecasting [4], [5], [9], [10], our Lastjormer can dynamically reveal the complex spatial and temporal correlations simultaneously. Moreover, in contrast to state-of-the-art Transformer-based approaches [11], [12] in traffic forecasting, our Lastjormer can further model all sophisticated relationships including the cross-sensor and cross-time correlations in the traffic data by replacing the stacked or paralleled spatial and temporal self-attention with our elaborately designed spatio-temporal joint attention.

The spatio-temporal joint attention flattens the 2D spatio-temporal traffic sequence as a 1D sequence and utilizes the dot-product self-attention on the sequence to dynamically extract all correlations among them. Besides, to maintain the global receptive field and enhance the locality representation power of the canonical dot-product self-attention, we introduce the convolution attention mechanism [16] into our model. Specifically, we employ the causal convolution and our proposed multi-hop diffusion convolution in self-attention to produce queries and keys with local spatio-temporal information. The local spatio-temporal information can help our spatio-temporal joint attention correctly match the query-key pairs with similar local temporal trends and in the same local sub-structure of the road network. However, the implementation of our spatio-temporal joint attention is difficult because of the heavy computation needs brought by the vanilla self-attention with the quadratic complexity. Inspired by the linear attention mechanism [17], we replace the dot-product self-attention in our spatio-temporal joint attention with the linear self-attention to afford the memory usage and speed up the calculation. Through the effective incorporation of these designs, our model outperforms previous models of traffic forecasting accuracy. The main contributions of our work are summarized as follows:

- To the best of our knowledge, we are the first to utilize a novel spatio-temporal joint attention in the Transformer architecture to capture all the dynamic dependencies in the traffic data, especially the cross-time spatial dependence and cross-sensor temporal dependence.
- For effective local and global modeling for traffic forecasting, we introduce a convolution attention mechanism into our model that leverages the causal convolution and multi-hop diffusion convolution to produce queries and keys in self-attention and can correctly match sensors.
- We introduce the linear attention mechanism into our model to reduce the time consumption in the training phase and afford the memory usage under the current computation resource.
- The experimental results on the three large-scale real-world traffic datasets of METR-LA, PEMS-BAY, and England show that our method significantly outperforms all the comparison state-of-the-art methods.

The rest of the paper is organized as follows. We first review the related works in Section II. Then the traffic forecasting task, diffusion convolution network, and dot-product self-attention are formulated in Section III. Section IV details our model, Section V introduces our experiments, and Section VI concludes the paper and discusses future work directions.

II. Related Works

We first review studies on traffic forecasting and then cover related work on the attention mechanism in this section.

A. Traffic Forecasting

Traffic forecasting has been studied for decades. At first, researchers used traditional statistical methods that were based on the stable assumption to forecast traffic, such as auto-regressive integrated moving average (ARIMA) [1] and vector auto-regression (VAR) [18]. These methods can only capture the linear temporal dependence and fail to achieve excellent results on unstable traffic sequences. Compared with traditional methods, shallow machine learning methods can capture non-linear dependence, *e.g.*, Wu *et al.* [2] used support vector regression (SVR), and Van *et al.* [19] used k-nearest neighbors (KNN) to predict traffic. However, shallow machine learning methods need to manually extract high-order features in the traffic data for specific scenarios, and the lack of generalization restricts their performance. With the brilliant success of deep learning in computer vision [20], natural language processing [21], and other fields [22], [23]. Some attempts based on deep learning have been made for traffic forecasting, [24], [25] applied RNN and its variants (*e.g.*, long short-term memory (LSTM) [26] and gated recurrent unit (GRU) [27]) to forecast traffic, which improves performance compared with previous methods but still ignores the spatial correlations between sensors in the traffic data. To capture spatial dependence, Zhang *et al.* [28] first used CNN to extract spatial correlations between pixels on the image-based traffic forecasting tasks based on the spatial aggregation ability of CNN. Subsequently, ConvLSTM [29] combined CNN with LSTM for traffic forecasting to capture spatial and temporal

dependencies simultaneously. ST-ResNet [30] further used the residual connection to increase the depth of the CNN model and capture the global spatial dependence. Although CNN has achieved excellent performance in image-based traffic forecasting tasks, it is not suitable for road network-based traffic forecasting tasks because the road network is non-Euclidean. GCN is a convolution operation defined on the non-Euclidean space [31], which is consistent with the role of CNN on the image and can capture the spatial dependence in the non-Euclidean structure by passing messages between nodes. DCRNN [4] was the first work that applied GCN on road network-based traffic forecasting tasks, which replaced the linear operation in the GRU with the diffusion convolution [32] and recurrently used the diffusion convolution to extract the spatial correlations at each time slice. Another strategy to use graph convolution for traffic forecasting is to combine GCN with causal convolution [33], such as STGCN [5]. Although follow-up works Graph WaveNet [9], MTGNN [10], and AGCRN [34] utilized the adaptive graph that learned through back-propagation to capture the global spatial dependence in the road network, they fail to reveal dynamic correlations in the traffic data. Different from GCN and TCN/RNN, spatial and temporal attention can dynamically calculate the weights between sensors and time slots on the spatial and temporal dimensions. Therefore, ST-GRAT [14] stacked the spatial and temporal attention in the vanilla Transformer architecture to predict traffic, GMAN [11] paralleled the spatial and temporal attention to forecast traffic and proposed transform attention between the encoder and decoder to reduce inference time of Transformer. To address the local spatial insensitivity of self-attention, Traffic Transformer [12] used the K-hop adjacent matrix as a mask to dynamically attend local spatial dependencies of traffic data. However, as shown in Fig. 1, the cross-time spatial correlations and cross-sensor temporal correlations are ignored in previous Transformer-based methods.

B. Attention Mechanism

The earliest addictive attention appeared in the encoder-decoder framework proposed by Bahdanau *et al.* [35]. Later, [36] proposed the most commonly used dot-product attention for natural language processing. Vaswani *et al.* [15] then designed the most famous Transformer architecture, which completely uses attention to solve tasks (*e.g.*, machine translation [37] and question answering system [38]) and achieves superior performance. Transformer makes the attention mechanism no longer just an auxiliary tool of the encoder-decoder framework. In the next period, Transformer-based works have sprung up in computer vision, signal processing [22], and bioinformatics [39] and have achieved the best results in these tasks. However, the quadratic time and space complexity of the dot-product self-attention in Transformer hinder model scalability in many settings. To address the quadratic complexity issue, Sparse Transformer [40] and Longformer [41] utilized pre-define fixed local patterns to optimize self-attention. Reformer [42] and Routing Transformer [43] designed a hash-based similarity measurement and an online k-means clustering, they are efficient in learning local patterns. Linformer [44] found that the feature map matrix in self-attention has low-rank property and mapped keys in self-attention into a low-rank matrix to transform the $N \times N$ calculation into $N \times k$. Performer [17] enabled clever mathematical re-writing of the self-attention and no longer calculates the $N \times N$ feature map matrix in self-attention. In this paper, we propose a novel spatio-temporal joint attention that can effectively and efficiently attend all dynamic relationships in traffic data.

III. Preliminaries

In this section, we define traffic forecasting as a mathematical problem and then introduce the two algorithms involved in our method, *i.e.*, diffusion convolution network and dot-product self-attention.

A. Problem Definition

The goal of traffic forecasting is to predict the future traffic data (*e.g.*, flow and speed) by giving previously observed traffic data from N correlated sensors on the road network. We define N correlated sensors as a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where $\mathcal{V}$ is a set of vertices representing roads and $|\mathcal{V}| = N$, $\mathcal{E}$ is a set of edges, and $A \in \mathbb{R}^{N \times N}$ is a weighted directed adjacency matrix of $\mathcal{G}$ that represents the proximity of sensors. Particularly, the adjacency matrix used in our paper is constructed according to the historical real-world road network, and weights in the adjacency matrix are derived through distance on the road network. Denote the traffic data observed by the sensor i at time t as a graph signal $x_t^i \in \mathbb{R}^C$, where $C = 1$ (*i.e.*, traffic flow or speed). The traffic forecasting task aims to learn a function f for forecasting the T_p future graph signals $\hat{\mathcal{Y}} \in \mathbb{R}^{T_p \times N \times C}$ from the known T_h graph signals $\mathcal{X} \in \mathbb{R}^{T_h \times N \times C}$ and the graph $\mathcal{G}$:

$$[X_1, \ldots, X_{T_h}; \mathcal{G}] \underset{f(\cdot)}{\rightarrow} [\hat{Y}_{T_h+1}, \ldots, \hat{Y}_{T_h+T_p}] \tag{1}$$

B. Diffusion Convolution Network

DCRNN [4] proposed a diffusion convolution network for suiting the nature of traffic. Specifically, [45] proved that the diffusion process of information on a graph $\mathcal{G}$ can be formulated as a process of random work with the restart probability $\alpha \in [0, 1]$ and transition matrix $D_O^{-1}A$. $D_O = diag(rowsum(A))$ is the out-degree diagonal matrix. The stationary distribution of the diffusion process can be represented as a weighted combination of infinite random walks on the graph and calculated in closed form:

$$\mathcal{P} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k (D_O^{-1}A)^k, \tag{2}$$

where $\mathcal{P} \in \mathbb{R}^{N \times N}$ denotes the stationary distribution of the Markov process of diffusion. In DCRNN, a finite S scale truncation of the diffusion process with the trainable weights are used. The diffusion process of S scales can be written as:

$$X_{\star G}\Theta = \sum_{s=0}^{S} (\theta_{s,1}(D_O^{-1}A)^s + \theta_{s,2}(D_I^{-1}A^T)^s)X, \tag{3}$$

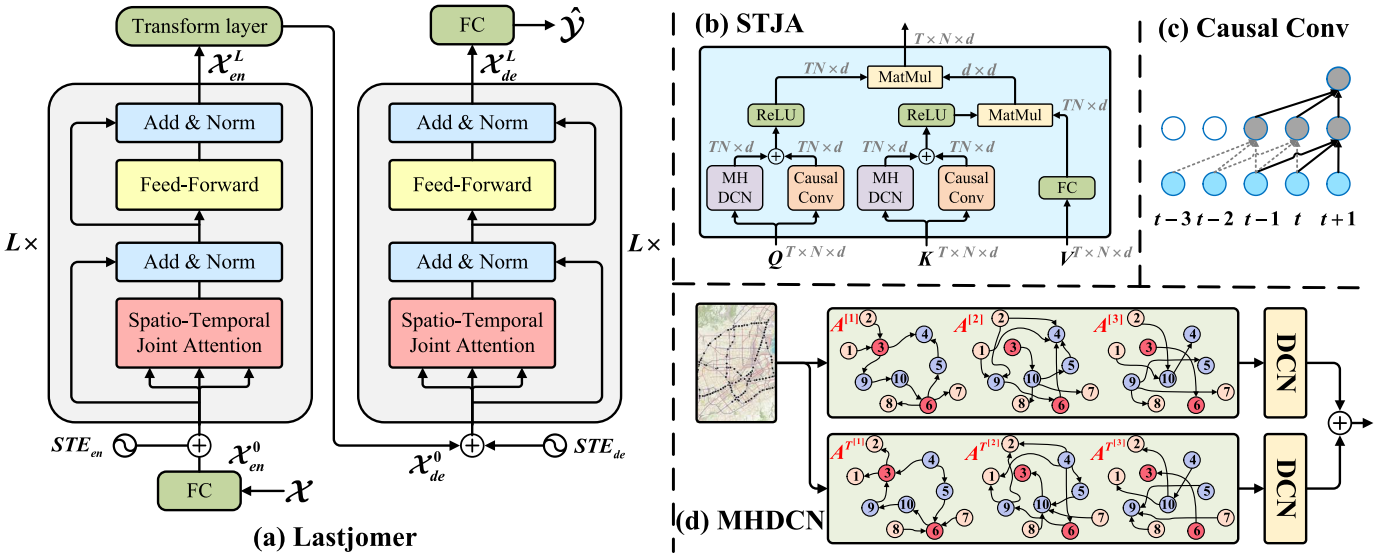


Fig. 2. (a) is the overall architecture of our Lastjormer. (b) is the detailed overview of our spatio-temporal joint attention (STJA). (c) is the illustration of the casual convolution (causal conv) with kernel size 3. (d) is the illustration of our multi-hop diffusion convolution network (MHDCN) with hop 3, where DCN indicates the diffusion convolution network.

where $X \in \mathbb{R}^{N\times d}$ is the input signal, $\theta_{s,1}, \theta_{s,2} \in \mathbb{R}^{d\times d}$ are parameters for the filter, and $D_O^{-1}A$, $D_I^{-1}A^T$ represent the bidirectional transition matrices of the diffusion process.

C. Dot-Product Self-Attention

The dot-product self-attention mechanism first calculates correlations between each tokens in the sequence, and then uses the calculated feature map matrix to aggregate information from other tokens. Formally, the input sequence $X \in \mathbb{R}^{N\times d}$ is projected by three matrices $W^Q, W^K, W^V \in \mathbb{R}^{d\times d}$ to corresponding representations Q, K and V:

$$Q = W^Q X,\ K = W^K X,\ V = W^V X, \tag{4}$$

then the self-attention is computed as follows:

$$Att(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d}})V, \tag{5}$$

From [46], we can write self-attention in a more general form by replacing the softmax with a similarity function:

$$Att(Q,K,V)_i = \frac{\sum_{j=1}^{N} sim(Q_i,K_j)V_j}{\sum_{j=1}^{N} sim(Q_i,K_j)}, \tag{6}$$

when we substitute the similarity function with $sim(q,k) = exp(\frac{qk^T}{\sqrt{d}})$, the Eq. (6) is equivalent to Eq. (5).

IV. Methodology

In this section, we will detail our method. Fig. 2 illustrates the framework of our proposed Lastjomer, which comprises the encoder, decoder, and Transform layer. The encoder and decoder have the same structure that stacks the spatio-temporal joint attention and feed-forward network L times with the spatio-temporal embedding. The Transform layer is in the middle of the encoder and decoder and can convert the historical information of the encoder into the future. We will detail the encoder, Transform layer, decoder, spatio-temporal joint attention, feed-forward network, and spatio-temporal embedding in order next.

A. Encoder Stacks

Before the encoder, we use a fully-connected layer to project the input traffic data $\mathcal{X} \in \mathbb{R}^{T_h\times N\times C}$ into a high-dimensional space $\mathcal{X}_{en}^0 \in \mathbb{R}^{T_h\times N\times d}$ to enhance the representation power of our model. The encoder consists of L stacked identical layers, and each layer includes two sub-layers. The first is our spatio-temporal joint attention (STJA) layer, and the second is a simple fully-connected feed-forward layer. Following the vanilla Transformer, we adopt the residual connection and layer normalization for each sub-layer. The overall equations for l-th encoder layer can be formulated as:

$$\begin{aligned}\mathcal{D}_{en}^l &= LN(STJA([\mathcal{X}_{en}^{l-1}, STE_{en}]) + \mathcal{X}_{en}^{l-1})\\ \mathcal{X}_{en}^l &= LN(FeedForward(\mathcal{D}_{en}^l) + \mathcal{D}_{en}^l),\end{aligned} \tag{7}$$

where $LN(\cdot)$, $[\cdot,\cdot]$, and $STJA(\cdot)$ represent the layer normalization, concatenate operation, and spatio-temporal joint attention, respectively. $STE_{en} \in \mathbb{R}^{T_h\times N\times d}$ is the historical spatio-temporal embedding. We will give a detailed description of STE_{en} and $STJA(\cdot)$ in the next.

B. Transform Layer

After the encoder, we first utilize a Transform layer between the encoder and decoder to transform the useful historical information from the encoder to expected future sequences, and the future sequences can be further fed into the decoder to capture the intra-dependencies of the future. Different from the previous work GMAN [11] uses Transform attention to convert traffic, we leverage the GRU to generate future sequences and our method can avoid the strong assumption in GMAN that the spatio-temporal embedding can be seen as traffic in Transform

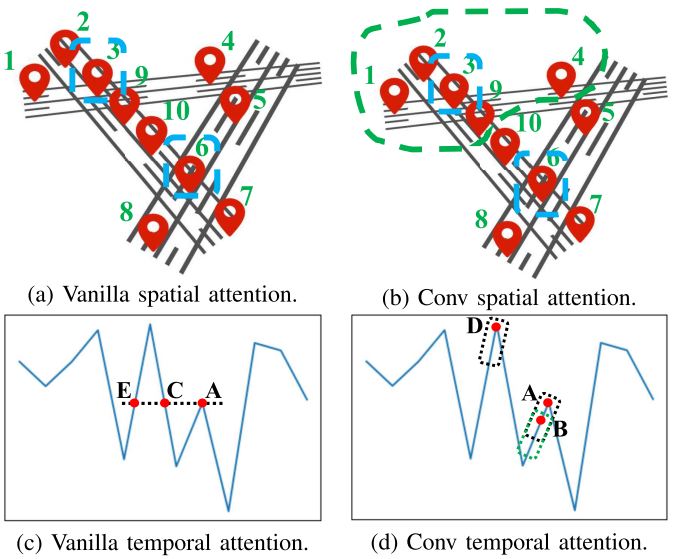


Fig. 3. The comparison between vanilla self-attention (left) and convolution (conv) self-attention (right) when they are applied to the spatial and temporal dimensions of the traffic data. The vanilla spatial self-attention will only match 6 with 3 and the vanilla temporal self-attention will wrongly match C, E with A since they have the same point-wise numerical value. In contrast, the convolution spatial self-attention can correctly match other relevant points (*i.e.*, neighbors in the green dotted box) by passing local spatial influence, and the convolution temporal self-attention can correctly match the most relevant points (*i.e.*, B and D) based on local temporal trends.

attention. Specifically, we set the last state $X^L_{en_{T_h}}$ from the encoder as the initial hidden state $H_{T_h-1} \in \mathbb{R}^{N\times d}$ and start token $X^0_{de_{T_h}} \in \mathbb{R}^{N\times d}$ in GRU to iteratively generate future sequences. Our transform GRU layer can be formulated as:

$$\begin{aligned} R_t &= \sigma([X^0_{de_t}, H_{t-1}]W^r + b^r) \\ U_t &= \sigma([X^0_{de_t}, H_{t-1}]W^u + b^u) \\ \tilde{H}_t &= tanh([X^0_{de_t}, (R_t \odot H_{t-1})]W^h + b^h) \\ H_t &= U_t \odot H_{t-1} + (1 - U_t) \odot \tilde{H}_t \\ X^0_{de_{t+1}} &= H_t W^o + b^o \end{aligned} \tag{8}$$

where σ and $\odot$ denote the sigmoid function and the hadamard product. $W^r, W^u, W^h, W^o \in \mathbb{R}^{d\times d}$ and $b^r, b^u, b^h, b^o \in \mathbb{R}^d$ are learnable parameters. Through the Transform GRU layer, we can derive the future sequence $\mathcal{X}^0_{de} \in \mathbb{R}^{T_p\times N\times d}$.

C. Decoder Stacks

Similar to the encoder, each layer in the decoder contains the STJA and feed-forward. The STJA in the decoder can capture the spatio-temporal dependencies in the future. Suppose there are L layers in the decoder. With the hidden states $\mathcal{X}^0_{de}$ from the Transform layer, the equation of l-th layer in the decoder can be summarized as follows:

$$\begin{aligned} \mathcal{D}^l_{de} &= LN(STJA([\mathcal{X}^{l-1}_{de}, STE_{de}]) + \mathcal{X}^{l-1}_{de}) \\ \mathcal{X}^l_{de} &= LN(FeedForward(\mathcal{D}^l_{de}) + \mathcal{D}^l_{de}), \end{aligned} \tag{9}$$

where $STE_{de} \in \mathbb{R}^{T_p\times N\times d}$ is the spatio-temporal embedding in the future. At the bottom of our model, we use a fully-connected layer to project the output $\mathcal{X}^L_{de} \in \mathbb{R}^{T_p\times N\times d}$ from high-dimensional space into the traffic $\hat{\mathcal{Y}} \in \mathbb{R}^{T_p\times N\times C}$.

D. Spatio-Temporal Joint Attention

To discover the intricate relationships in the traffic data, previous methods utilize the self-attention on the spatial and temporal dimensions of the traffic data to dynamically reveal the temporal correlations between sensors and their historical records and the spatial correlations between sensors and other sensors at the same time. However, attention-based works still fail to extract some implicit correlations hidden in the traffic data, *e.g.*, cross-sensor temporal correlations and cross-time spatial correlations. Therefore, different from separately using the spatial and temporal self-attention for traffic forecasting, we propose a novel spatio-temporal joint attention to capture all dynamic dependencies in the traffic data, which is the key technical contribution of our paper. Specifically, we flatten the two-dimensional spatio-temporal traffic data $\mathcal{X} \in \mathbb{R}^{T\times N\times d}$ to a one-dimensional sequence $\bar{X} \in \mathbb{R}^{TN\times d}$ and fed the sequence into a vanilla self-attention layer to make interactions between all sensors. Mathematically, the spatio-temporal joint attention can be formulated as:

$$\begin{aligned} STJA(\bar{X}) &= \Big\|_{i=1}^{TN} (Att(\bar{Q}, \bar{K}, \bar{V})_i), \; where \\ \bar{Q} &= W^Q \bar{X}, \; \bar{K} = W^K \bar{X}, \; \bar{V} = W^V \bar{X} \end{aligned} \tag{10}$$

where $\|(\cdot)$ represents the concatenate operation. To consist of the shape with the feed-forward layer, we reshape the output of $STJA(\cdot)$ to the two-dimensional spatio-temporal traffic data.

Even though our spatio-temporal joint attention can represent arbitrary relationships well, two problems limit its applicability for traffic prediction. First of all, it suffers from the local insensitivity problem, *i.e.*, the similarities between queries and keys in the vanilla self-attention are computed based on their point-wise values without fully leveraging the local context. As shown in Fig. 3a and 3c, which may ignore the essential local spatial dependence in the road network and wrongly capture the temporal dependence. Another problem comes from the quadratic complexity of the vanilla self-attention, *i.e.*, the time and space complexity of our spatio-temporal joint attention are $O((TN)^2)$, which brings unbearable computing demands. Therefore, we introduce the convolution and linear self-attention mechanism in our spatio-temporal joint attention to enhance its local representation capabilities and reduce its computation needs.

1) Convolution Self-Attention: We utilize the convolution self-attention mechanism to ease the local insensitive issue of vanilla self-attention. The architectural view of it is illustrated in Fig. 1(b). Rather than using a fully-connected layer to transform inputs on the feature dimension, we employ a one-layer causal convolution of kernel size k with stride 1 (with proper paddings) and a novel multi-hop diffusion convolution (proposed by our and is detailed next) to transform inputs into queries and keys. Mathematically, we can rewrite our spatio-temporal joint attention as follows:

$$\begin{aligned} STJA(\bar{X}) &= \Big\|_{i=1}^{TN} (Att(\tilde{Q}, \tilde{K}, \bar{V})_i), \; where \\ \tilde{Q} &= \bar{X}_{\star G}\Theta^Q + \bar{X} \star \Phi^Q \\ \tilde{K} &= \bar{X}_{\star G}\Theta^K + \bar{X} \star \Phi^K \end{aligned} \tag{11}$$

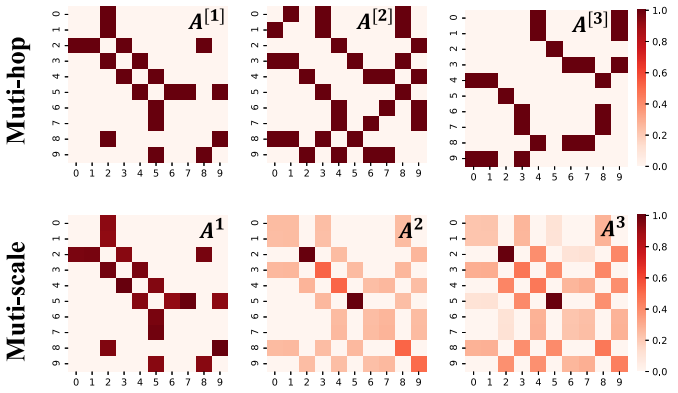


Fig. 4. An illustration of the multi-scale aggregation and our multi-hop aggregation. The adjacency matrix is constructed using the road network in Fig. 1. Through adjacency powering in multi-scale aggregation, the connection is biased toward A^1. Our novel multi-hop aggregation shows that all neighbors are strongly connected with a set of sub-graphs.

where $\star G$ and $\star$ denote the multi-hop diffusion convolution and causal convolution, respectively. $\Theta^Q, \Theta^K, \Phi^Q, \Phi^K$ are parameters of convolutions. By employing the causal convolution, generated queries and keys can be more aware of local temporal information, and hence, computing similarities in attention by sensors local temporal trends instead of point-wise values can be helpful for accurate forecasting. By employing the multi-hop diffusion convolution, generated queries and keys can receive information from the neighbor sensors depending on the local sub-structure of road networks, and thus the computed point-wise similarities are more sensitive to local spatial sensors. The reason for not taking multi-scale diffusion convolution is that there is a weight bias problem that occurs when powering the adjacency matrix for multi-scale aggregation operations, *i.e.*, the edge weight decreases exponentially when adjacency powering. As shown in Fig. 4, the weights in the multi-scale aggregation are only biased toward among connections in A^1. Moreover, connections in previous scales are repeatedly computed later, enhancing the weight bias problem. Therefore, we present a novel multi-hop diffusion convolution that utilizes multi-hop aggregation to gain local spatial information as shown in Fig. 4. To do this, we first define the s-hop adjacency matrix $\tilde{A}^{[s]} \in \mathbb{R}^{N\times N}$ according to the shortest distance between sensors. Algorithm 1 shows how to construct the s-hop adjacency matrix. Substituting the polynomial term A^s, A^{T^s} with s-hop adjacency matrix $\tilde{A}^{[s]}, \tilde{A}^{T^{[s]}}$, Eq. (3) can be reformulated as:

$$X_{\star G}\Theta = \sum_{s=0}^{S}(\theta_{s,1}(\tilde{D}_O^{[s]^{-1}}\tilde{A}^{[s]}) + \theta_{s,2}(\tilde{D}_I^{[s]^{-1}}\tilde{A}^{T^{[s]}}))X, \quad (12)$$

where $\tilde{D}_O^{[s]}$ and $\tilde{D}_I^{[s]}$ are the out/in degree matrix of $\tilde{A}^{[s]}$.

2) Linear Self-Attention: Among the current work on optimizing the self-attention mechanism, we consider leveraging the linear mechanism to optimize our spatio-temporal joint attention because it can ensure the acquisition of global information while reducing the quadratic space and time complexity to linear with regard to the input sequence length, thus exhibiting strong capacity in modeling long

Algorithm 1 Construct the s-Hop Adjacency Matrix

Data: Road network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$;
Initialize a s-hop matrix $\tilde{A}^{[s]} \in \mathbb{R}^{N\times N}$ with zero;
Initialize an unweighted matrix $\bar{A} \in \mathbb{R}^{N\times N}$ with zero;
Initialize a shortest path matrix $\mathcal{S} \in \mathbb{R}^{N\times N}$ with zero; **for** $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** N **do**
    $\bar{A}[i,j] = \begin{cases} 1 & , A[i,j] \neq 0 \\ 0 & , otherwise \end{cases}$
  end
end
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** N **do**
    If there is a path from sensor i to sensor j, compute the shortest distance d from i to j based on $\bar{A}$, and $\mathcal{S}[i,j] = d$;
  end
end
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** N **do**
    $\tilde{A}^{[s]}[i,j] = \begin{cases} 1 & , \mathcal{S}[i,j] = s \\ 0 & , otherwise \end{cases}$
  end
end
Result: s-hop adjacency matrix $\tilde{A}^{[s]}$

sequences. [46] points out that self-attention only needs to ensure the non-negativity of $sim(\cdot)$ in Eq. (6) to define the attention function. This includes all kernels $sim(\cdot,\cdot) : \mathbb{R}^{2\times d} \rightarrow \mathbb{R}_+$. Therefore, we can rewrite Eq. (6) in the form of kernel functions with the kernel function $\phi(\cdot)$ and sequence length TN as:

$$Att(Q,K,V)_i = \frac{\sum_{j=1}^{TN}\phi(Q_i)\phi(K_j^T)V_j}{\sum_{j=1}^{TN}\phi(Q_i)\phi(K_j^T)} \quad (13)$$

After that, the linear self-attention operation is achieved via the associative property of matrix product:

$$LAtt(Q,K,V)_i = \frac{\phi(Q_i)\sum_{j=1}^{TN}\phi(K_j^T)V_j}{\phi(Q_i)\sum_{j=1}^{TN}\phi(K_j^T)} \quad (14)$$

Specifically, in order to ensure full positive attention and avoid aggregating negatively-correlated information, we adopt the rectified linear unit as the kernel functions and therefore effectively eliminate negative values:

$$\phi(x) = ReLU(x) \quad (15)$$

The formulation of the proposed linear self-attention as:

$$LAtt(Q,K,V)_i = \frac{ReLU(Q_i)\sum_{j=1}^{TN}ReLU(K_j^T)V_j}{ReLU(Q_i)\sum_{j=1}^{TN}ReLU(K_j^T)} \quad (16)$$

Last, the final formulation of spatio-temporal joint attention used in our model is:

$$STJA(\bar{X}) = \Big\|_{i=1}^{TN}(LAtt(\tilde{Q},\tilde{K},\bar{V})_i) \quad (17)$$

Algorithm 2 Training Procedure of Lastjomer

Data: Road network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$;
Time steps TS of train set;
Train data of all observing stations $\mathcal{TD} \in \mathbb{R}^{TS \times N \times C}$;
All hyperparameters;
Employ node2vec and onehot encoding to get the $STE \in \mathbb{R}^{TS \times N \times d}$;
for $t \leftarrow 1$ **to** $TS - 24$ **do**
  Append $\mathcal{TD}_{t:t+12}$ to input sample set $\mathcal{X}$;
  Append $\mathcal{TD}_{t+12:t+24}$ to label sample set $\mathcal{Y}$;
  Append $STE_{t:t+24}$ to embedding set $\mathcal{C}$;
end
Initialize all learnable parameters Θ in Lastjomer;
repeat
  Randomly select a batch of input sample $\mathcal{X}_{bs}$;
  Randomly select a batch of label sample $\mathcal{Y}_{bs}$;
  Randomly select a batch of embedding sample $\mathcal{C}_{bs}$;
  Optimize Θ by minimizing the objective function Eq. (19) with inputs of $\mathcal{X}_{bs}$, $\mathcal{Y}_{bs}$, and $\mathcal{C}_{bs}$;
until *met model stop criteria*;
Result: Learned Lastjomer model.

TABLE I
STATISTICS OF EXPERIMENTED DATASETS

Category	Datasets	#Nodes	#Time Steps	Mean	Std
Traffic flow	England	249	35040	427	382
Traffic speed	METR-LA	207	34272	58	13
	PEMS-BAY	325	52116	63	10

E. Other Components

1) Spatio-Temporal Embedding: To more effectively distinguish sensors at different positions and times, we concatenate the spatio-temporal embedding (STE) with input before each layer. The spatial embedding matrix $SE \in \mathbb{R}^{N \times d}$ is generated by node2vec [47], which randomly samples neighbors to train a vector for each node in a graph. The temporal embedding matrix $TE \in \mathbb{R}^{T \times d_{te}}$ is generated by one-hot encoding, where d_{te} denotes the time slots in a day and the value at the current time is 1. Then we fed the spatial and temporal embedding metrics into two fully-connected layers and derive the spatial embedding $SE \in \mathbb{R}^{N \times d}$ and temporal embedding $TE \in \mathbb{R}^{T \times d}$, respectively. To obtain the time-variant vertex representations, we fuse the aforementioned spatial embedding and temporal embedding as $STE \in \mathbb{R}^{T \times N \times d}$.

2) Feed-Forward Layer: The feed-forward layer in our model consists of two linear transformations:

$$FeedForward(\mathcal{X}) = ReLU(\mathcal{X}W^{F_1} + b^{F_1})W^{F_2} + b^{F_2} \tag{18}$$

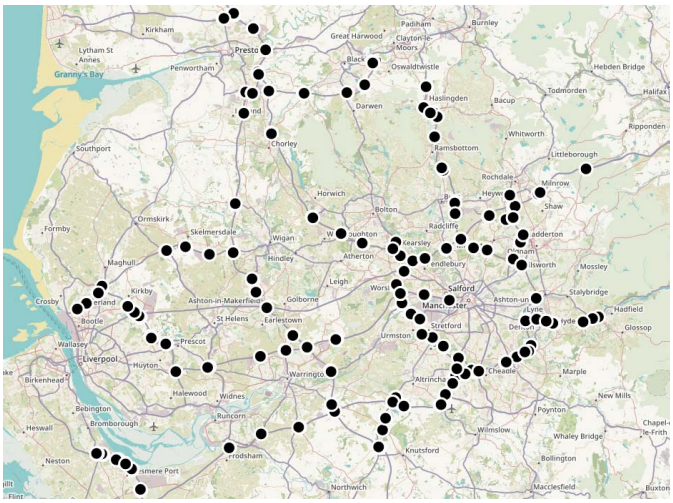
where W^{F_1}, $W^{F_2} \in \mathbb{R}^{d \times d}$ and b^{F_1}, $b^{F_2} \in \mathbb{R}^{d}$ are learnable parameters.

F. Loss Function

In this paper, we choose $L1$ loss as our training objective function and optimize the loss via back-propagation and the Adam optimizer [48]. The loss function can be formulated as:

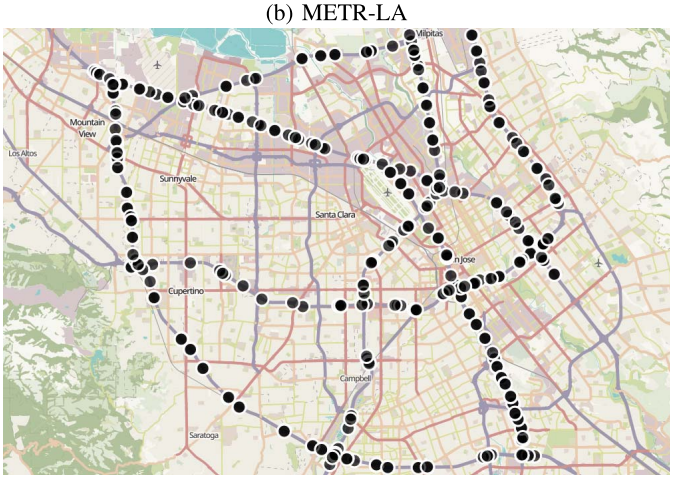
$$\mathcal{L}(\Theta) = \sum_{t=1}^{T_p} \sum_{n=1}^{N} |x_t^n - \hat{y}_t^n| \tag{19}$$

where Θ indicates all the learnable parameters in our model. x_t^n and $\hat{y}_t^n$ represent the ground truth and predicted value of



(a) England

(b) METR-LA



(c) PEMS-BAY

Fig. 5. The distribution of sensors on roads.

sensor n at time t, respectively. The training procedure of our Lastjomer is summarized in Algorithm 2.

V. Experiments

This section aims to answer following research questions:

- **RQ1**: How does our developed Lastjomer framework perform as compared to various state-of-the-art methods?
- **RQ2**: How do different components (*e.g.*, encoder and decoder) affect the results of Lastjomer?
- **RQ3**: How do the MHDCN, GRU, and STJA contribute to the performance?
- **RQ4**: How efficient is Lastjomer in traffic forecasting?
- **RQ5**: What is the influence of parameters in Lastjomer?
- **RQ6**: Can Lastjomer provide the model interpretability w.r.t spatial and temporal dimensions?

A. Experimental Settings

1) Datasets: We evaluate Lastjomer on three public traffic datasets, they are England [49], METR-LA [4], and PEMS-BAY [4]. The England dataset records traffic flow on the England highways from January 1st, 2014 to December 31st, 2014. METR-LA and PEMS-BAY record traffic speed on the Los Angeles County and Bay Area in America from March 1st, 2012 to June 30th, 2012, and January 1st, 2017 to May 31st, 2017, respectively. The England dataset aggregates traffic flow observations into 15 minutes windows. METR-LA and PEMS-BAY aggregate traffic speed observations into 5 minutes windows. Besides, we apply Z-Score normalization to normalize data before the model and re-transform the predicted values back to the actual values after the model. All the datasets are split in chronological order with 70% for training, 10% for validation, and 20% for testing. Table I and Fig. 5 show the detailed statistics and distribution of sensors of these datasets.

2) Baselines: We compare our Lastjomer with widely used traffic forecasting models, including:

- VAR [18]. Vector auto-regression, which captures the pairwise relationships among multiple time series and we set order of 12 in experiments.
- SVR [2]. Support vector regression model. We use support vector with rbf kernel to iteratively perform multi-step predictions.
- LSTM [26]. Long short-term memory network, which is a variant of recurrent neural network. Compared with ordinary rnn, LSTM effectively prevents the gradient vanish and capture the long range temporal dependence.
- DCRNN [4]. Diffusion convolution recurrent neural network, which replaces the linear operation in the GRU with a diffusion graph convolution network to extract the spatial dependence. Besides, DCRNN uses a encoder-decoder architecture for multi-step forecasting and schedule sampling strategy to speed up convergence.
- STGCN [5]. Spatial-temporal graph convolution network uses graph convolution and gated linear unit (GLU) [50] to capture spatio-temporal dependencies, which greatly accelerates the training speed compared with RNN.
- Graph WaveNet [9]. A advanced model based on STGCN, which uses the self-adaptive adjacency matrix in graph convolution to capture the learned spatial dependence. It further utilizes Gated TCN instead of GLU to capture temporal dependence.
- MTGNN [10]. An improved model of Graph WaveNet, which no longer uses any prior knowledge for constructing graph and obtains multi-scale spatio-temporal information compared with Graph WaveNet.
- GMAN [11]. A graph multi-attention network, which adopts an encoder-decoder architecture, where both the encoder and decoder consist of multiple spatio-temporal attention blocks to capture spatio-temporal dependencies. In addition, GMAN design a novel transform attention to avoid dynamic decoding in vanilla Transformer.
- Traffic Transformer [12]. A variant of Transformer. It uses the global encoder and global-local decoder to extract and fuse the spatial patterns globally and locally and can learn the dynamic and hierarchical structure of traffic.

3) Metrics: We use the mean absolute error (MAE), root mean square error (RMSE), and mean absolute percentage error (MAPE) to evaluate the performance for all experiments on all datasets. These metrics are formulated as follows:

$$MAE = \sum_{t=1}^{T}\sum_{n=1}^{N}|x_t^n - \hat{y}_t^n| \tag{20}$$

$$RMSE = \sum_{t=1}^{T}\sum_{n=1}^{N}\sqrt{(x_t^n - \hat{y}_t^n)^2} \tag{21}$$

$$MAPE = \sum_{t=1}^{T}\sum_{n=1}^{N}|\frac{x_t^n - \hat{y}_t^n}{x_t^n}| \tag{22}$$

4) Parameter Settings: We set $T_h = 12$ and $T_p = 12$ in our paper, *i.e.*, we utilize the known traffic data at historical 12 time slots to forecast the traffic data at future 12 time slots, which is the standard benchmark-setting in this domain. All experiments in our paper are conducted using PyTorch with Tesla-A100. For consistency, we apply the same dimension size d for all weight matrices. Specifically, we set the d to 64, number of layers of Lastjomer to 3, number of layers of GRU to 1, hops of MHDCN to 4, kernel size of causal convolution to 3, heads of attention to 8, and dimensions of each head in attention to 8, respectively. All the trainable parameters in our model are optimized using Adam optimizer with the batch size of 16, learning rate of 0.001, epoch 40, and $L1$ loss function.

For baselines, we run their codes with optimal hyperparameters based on their recommended configurations if their accuracy is not known for a dataset. If known, we use their officially reported accuracy.

B. Performance Comparison (RQ1)

We summarise the evaluation results of all methods on traffic flow and speed forecasting tasks in Tables II and III. In general, we observe that our Lastjomer consistently and significantly outperforms all baseline techniques on most tasks. Besides, we have the following observations based on the experimental results.

TABLE II
TRAFFIC FLOW FORECASTING PERFORMANCE ON THE ENGLAND DATASET

Datasets	Methods	45 min			90 min			3 hour		
		MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
England	VAR	50.14	78.72	36.46	82.66	122.82	60.57	105.21	150.72	83.10
	SVR	54.42	85.88	53.23	97.02	152.44	70.12	139.35	223.89	74.75
	LSTM	58.09	97.87	28.85	97.56	159.68	48.61	131.22	208.38	70.55
	DCRNN	34.87	59.97	25.14	41.02	71.92	30.10	46.45	83.21	34.59
	STGCN	34.11	59.13	25.92	48.30	83.83	34.18	65.25	117.96	42.02
	Graph WaveNet	31.94	56.94	22.79	38.75	66.83	28.08	43.06	74.99	33.14
	MTGNN	32.47	56.45	25.11	38.22	65.71	31.79	42.69	72.88	36.23
	GMAN	33.85	58.69	29.28	37.87	65.77	32.73	41.20	70.94	36.15
	Traffic Transformer	33.30	57.72	28.81	37.68	65.28	31.38	44.30	79.49	36.16
	Lastjomer	**29.16**	**52.77**	**20.92**	**33.20**	**60.53**	**26.10**	**38.44**	**70.81**	**32.28**

TABLE III
TRAFFIC SPEED FORECASTING PERFORMANCE ON METR-LA AND PEMS-BAY DATASETS

Datasets	Methods	15 min			30 min			1 hour		
		MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
METR-LA	VAR	4.42	7.89	10.20	5.41	9.13	12.70	6.52	10.11	15.80
	SVR	3.99	8.45	9.30	5.05	10.87	12.10	6.72	13.76	16.70
	LSTM	3.44	6.30	9.60	3.77	7.23	10.90	4.37	8.69	13.20
	DCRNN	2.77	5.38	7.30	3.15	6.45	8.80	3.60	7.59	10.50
	STGCN	2.88	5.74	7.62	3.47	7.24	9.57	4.59	9.40	12.70
	Graph WaveNet	2.69	5.15	6.90	3.07	6.22	8.37	3.53	7.37	10.01
	MTGNN	2.69	5.18	6.86	3.05	6.17	8.19	3.49	7.23	9.87
	GMAN	2.77	5.48	7.25	3.07	6.34	8.35	3.40	7.21	9.72
	Traffic Transformer	2.66	**5.11**	6.75	3.00	6.06	**8.00**	3.39	7.04	**9.37**
	Lastjomer	**2.64**	**5.11**	**6.74**	**2.99**	**6.01**	8.13	**3.36**	**7.03**	9.67
PEMS-BAY	VAR	1.74	3.16	3.60	2.32	4.25	5.00	2.93	5.44	6.50
	SVR	1.85	3.59	3.80	2.48	5.18	5.50	3.28	7.08	8.00
	LSTM	2.05	4.19	4.80	2.20	4.55	5.20	2.37	4.96	5.70
	DCRNN	1.38	2.95	2.90	1.74	3.97	3.90	2.07	4.74	4.90
	STGCN	1.36	2.96	2.90	1.81	4.27	4.17	2.49	5.69	5.79
	Graph WaveNet	1.30	2.74	2.73	1.63	3.70	3.67	1.95	4.52	4.63
	MTGNN	1.32	2.79	2.77	1.65	3.74	3.69	1.94	4.49	4.53
	GMAN	1.36	2.93	2.88	1.64	3.78	3.71	1.90	4.40	4.45
	Traffic Transformer	1.35	2.82	2.84	1.66	3.72	3.75	1.95	4.49	4.65
	Lastjomer	**1.30**	**2.74**	**2.68**	**1.61**	**3.67**	**3.56**	**1.89**	**4.38**	**4.42**

Firstly, the excess performance of DCRNN and STGCN compared with VAR, SVR, and LSTM shows the powerful representation ability of deep learning on the non-linear traffic data, the robust generalization performance of deep learning, and the importance of spatial correlations for traffic forecasting. The performance of subsequent works further proves the effectiveness of spatial correlation and deep learning.

Secondly, the adaptive GCN-based (*i.e.*, the adjacency matrix is learned through back-propagation) models (Graph WaveNet and MTGNN) and self-attention-based models (GMAN and Traffic Transformer) achieve better performance than pre-defined GCN-based (*i.e.*, the adjacency matrix is constructed through the road network) models (DCRNN and STGCN). What the adaptive GCN and self-attention-based models have in common is that they have a global spatial receptive field, which is helpful to gain more information from remote but similar sensors.

Thirdly, compared to previous adaptive GCN and self-attention-based models, Traffic Transformer consistently yields significant improvements on most tasks. Because previous adaptive GCN and self-attention-based models remove all inductive bias of the road network and thus are insensitive to local spatial information. The results of our Lastjomer further show the effectiveness of considering the local spatio-temporal dependencies under the global receptive field.

Lastly, the overall performance of all models on METR-LA and PEMS-BAY is slightly better than that on England. The reason is that sensors in England are more sparse than METR-LA and PEMS-BAY, traffic flow is more fluctuating than the traffic speed (*i.e.*, the standard deviation of England is larger than METR-LA and PEMS-BAY), resulting in more complex spatio-temporal dependencies. The superiority of our proposed Lastjomer is more obvious in England, showing that Lastjomer can learn more from sparse and fluctuating traffic flow data compared with other alternatives. This is because Lastjomer can capture the cross-time spatial and cross-sensor temporal patterns and incorporate the local knowledge in both spatial and temporal dimensions.

TABLE IV
ABLATION STUDIES OF SUB-MODULES IN LASTJOMER ON THE METR-LA DATASET

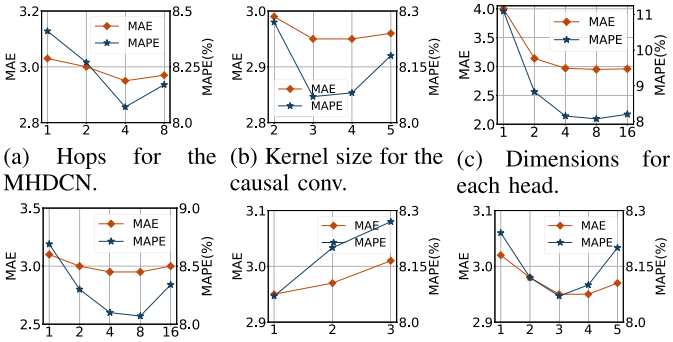
Methods	15 min			30 min			1 hour		
	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
w/o GRU	2.69	5.26	6.88	3.03	6.17	8.39	3.42	7.20	9.82
w/o Encoder	2.76	5.37	7.27	3.09	6.33	8.67	3.44	7.27	10.20
w/o Decoder	2.72	5.30	7.17	3.07	6.30	8.63	3.45	7.32	10.31
w/o MHDCN	2.73	5.35	7.11	3.06	6.31	8.50	3.40	7.21	9.94
w/o CausalConv	2.74	5.36	7.12	3.07	6.32	8.46	3.42	7.23	9.87
Lastjomer	**2.64**	**5.11**	**6.74**	**2.99**	**6.01**	**8.13**	**3.36**	**7.03**	**9.67**

TABLE V
THE PERFORMANCE OF LASTJOMER BEFORE AND AFTER REPLACING ITS DIFFERENT MODULES ON THE METR-LA DATASET

	Methods	15 min			30 min			1 hour		
		MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
Graph Convolution	JKNet	2.69	5.21	7.02	3.04	6.26	8.52	3.39	7.21	10.03
	Mixhop	2.68	5.21	6.98	3.03	6.22	8.43	3.38	7.17	9.93
	MHGCN	2.68	5.21	6.94	3.02	6.22	8.31	3.38	7.18	9.82
Transform layer	RNN	2.69	5.21	6.97	3.03	6.20	8.35	3.38	7.15	9.79
	LSTM	2.65	5.14	6.85	3.00	6.06	8.21	3.37	7.06	9.73
	MLP	2.66	5.16	6.90	3.03	6.23	8.36	3.39	7.23	9.94
	Attention	2.70	5.27	7.00	3.04	6.27	8.47	3.40	7.21	9.91
Attention type	Spatial	2.72	5.28	7.08	3.06	6.27	8.48	3.42	7.27	10.04
	Temporal	2.79	5.56	7.50	3.16	6.64	9.16	3.54	7.62	10.80
	Stacked	2.65	5.13	6.82	3.02	6.17	8.26	3.40	7.26	10.02
	Parallel	2.66	5.15	6.83	3.03	6.22	8.28	3.43	7.30	9.97
	Lastjomer-Full2	2.67	5.17	6.85	3.02	6.20	8.28	3.39	7.19	9.82
Ours	Lastjomer	**2.64**	**5.11**	**6.74**	**2.99**	**6.01**	**8.13**	**3.36**	**7.03**	**9.67**

TABLE VI
COMPUTATION NEEDS COMPARISON ON METR-LA

Methods	Traning (s/epoch)	Inference (s)	Memory (MiB)
DCRNN	245.32	33.79	6655
Graph WaveNet	82.81	4.24	3851
GMAN	323.75	18.97	13060
Traffic Transformer	118.69	4.56	5298
Lastjomer	111.52	4.30	4723
Lastjomer-Full2	252.47	14.83	20855



(a) Hops for the MHDCN. (b) Kernel size for the causal conv. (c) Dimensions for each head.

(d) Heads for the attention. (e) Layers for the GRU. (f) Layers for the Lastjomer.

Fig. 6. Experimental results on the METR-LA dataset under different hyperparameter settings.

C. Ablation Study (RQ2)

We design the following variants of Lastjomer to verify the effectiveness of each part proposed in our model:

- w/o GRU: Lastjomer is no longer equipped with GRU, *i.e.*, we remove the Transform layer in our model and use two encoders with a fully-connected layer on the feature dimension to predict the traffic flow/speed.
- w/o Encoder: This model removes the encoder in our Lastjomer.
- w/o Decoder: This model removes the decoder in our Lastjomer.
- w/o MHDCN: This model only utilizes the causal convolution to transform inputs into queries and keys.
- w/o CausalConv: This model only utilizes the MHDCN to transform inputs into queries and keys.

All variants have the same settings as Lastjomer, except the differences mentioned above. Table IV shows the performance on the METR-LA dataset. First, the worse performance of w/o GRU indicates that there is a huge gap between the learned historical information and the ground truth of the future, thus the Transform layer is important in our architecture. The decrease of performance in w/o Encoder and w/o Decoder represents the essential of simultaneously extracting spatio-temporal relationships in the history and future. When the MHDCN and causal convolution are removed, w/o MHDCN and w/o CausalConv have a clear performance drop proving that enhancing the spatio-temporal locality of our attention is effective for correctly matching sensors.

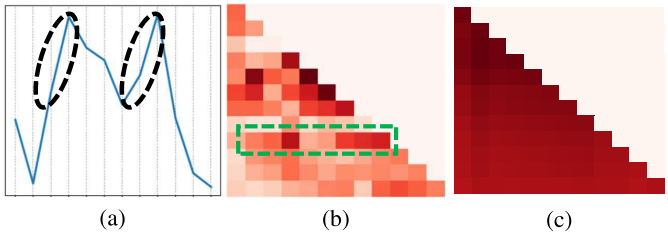


Fig. 7. (a) Time series of sensor 73 in METR-LA from 8:00 to 9:00 AM. (b) Attention weights of Lastjomer-Full2 between sensor 73 at 9:00 AM and its historical records to 8:00 AM. (c) Attention weights of Lastjomer-Full2 w/o Conv between sensor 73 at 9:00 AM and its historical records to 8:00 AM.

D. Module Replacement Analysis (RQ3)

In this subsection, we explore whether our framework design is separate and replaceable using traffic speed forecasting on the METR-LA dataset as an example. Several variants for components such as the MHDCN module, transform layer, and attention type are constructed for experiments.

More specifically, for the graph convolution, we attempt to replace our MHDCN with other common methods such as JKNet [51], Mixhop [52], and our multi-hop aggregation based-GCN (MHGCN). In terms of the Transform layer, we attempt to replace the GRU with the vanilla RNN, LSTM, MLP on the temporal dimension, and attention on the temporal dimension (assume spatio-temporal embedding is the traffic speed/flow). As for the attention type, we design five variants abbreviated to Spatial, Temporal, Stacked, Parallel, and Lastjomer-Full2. Spatial and Temporal denote only the spatial and temporal self-attention are used in our architecture. Stacked and Parallel indicate that spatial and temporal self-attention are stacked or parallel processed in our architecture. The Lastjomer-Full2 drops the linear attention mechanism in our spatio-temporal joint attention and can only stacked two layers limited by computing resources. Experimental results for these variants are shown in Table V. Specific analyses of these variants are as follows.

For the graph convolution, we can observe that JKNet performs worst among all variants, which implies only using adjacency matrix after powering, such as A^3 in Fig. 4, is a bias toward the identity and insufficiently to extract the local spatial correlation. One potential reason for the lower performance of Mixhop and MHGCN is that they ignore the direction of traffic, *i.e.*, they consider the inflow and outflow under the same transformation. Our MHDCN transforms the inflow and outflow under two different projections.

For the Transform layer, RNN and LSTM yield lower performance than GRU because of the under/over fitting. Besides, the attention is inferior to MLP under most evaluation metrics, indicating the spatio-temporal embedding can not be seen as the real-world traffic data. The results in Table V indicate that GRU is more suitable to transform historical data into the future compared with other common deep models.

For the attention type, Spatial and Temporal perform worse than other methods mainly because of their insufficient ability to capture temporal and spatial dependencies. Our Lastjomer makes significant improvement compared with Stacked and Parallel. This shows that Lastjomer can capture more subtle cross-sensor temporal and cross-time spatial dynamics to make a better traffic prediction. Lastjomer-Full2 receives comparable results as Lastjomer with only two layers. It proves the capability of our spatio-temporal joint attention in unveiling the implicit correlations in traffic data.

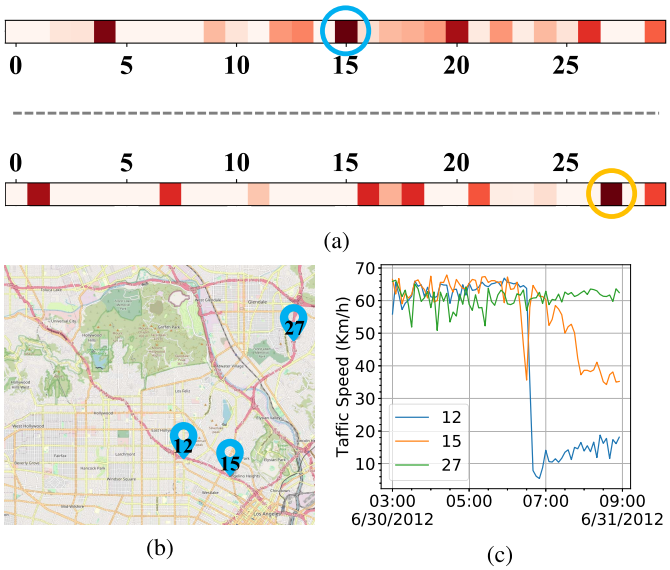


Fig. 8. (a) The top one is the attention weights of Lastjomer-Full2 between sensor 12 and the first 30 sensors at 6:00 AM; the bottom one is the attention weights of Lastjomer-Full2 w/o Conv between sensor 12 and the first 30 sensors at 6:00 AM. (b) Location of sensors 12, 15, and 27. (c) Time series of sensors 12, 15, and 27.

E. Computation Needs (RQ4)

We compare the computation needs of DCRNN, Graph WaveNet, GMAN, and Traffic Transformer with our Lastjomer-Full2 and Lastjomer on the METR-LA dataset. The training time, inference time, and memory usage are shown in Table VI. From the results in Table VI, we observe Lastjomer-Full2 is not suitable for large-scale traffic datasets due to the quadratic space and time complexity. With the use of a linear attention mechanism, our Lastjomer achieves comparable time and memory consumption with the state-of-the-art Graph WaveNet and Traffic Transformer. Moreover, our spatio-temporal joint attention with a linear time and space complexity about the number of time slots and sensors. DCRNN is less efficient in inference as it needs iterative computation to generate the 12 prediction results. GMAN is less efficient in training and memory as it stacks many quadratic temporal and spatial self-attention modules.

F. Effect of Hyperparameters (RQ5)

Fig. 6 shows the average prediction of Lastjomer under different hyperparameter settings over the next hour on the METR-LA dataset. When we tune one of the hyperparameters, the other hyperparameters remain the default optimal values. As shown in Fig. 6b and 6a, the best performance on small kernel size and number of hop denotes the long-range and

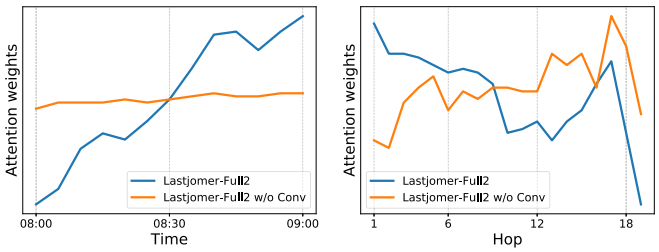


Fig. 9. (a) The average of attention weights that between all sensors at 9:00 AM and their historical records to 8:00 AM. (b) The average of attention weights that between all sensors and their neighbors with a different hop.

long-term spatial and temporal information will mislead the matching in the attention. Fig. 6c and 6d show that utilizing 8 heads and 8 dimensions for each head in Lastjomer can achieve the best performance. Fig. 6e and 6f show that stacking 1 layer of GRU and 3 layers of Lastjomer can obtain the best performance. We think this is because stacking too many GRU layers will increase the accumulative errors.

G. Learned Correlation Analysis (RQ6)

Experimental results indicate that our model can better dynamically extract intricate correlations in traffic data, especially for the local correlations compared with previous Transformer-based methods and the cross-sensor/time correlations compared with previous methods. In this section, we offer several examples to give an intuitive impression of our model explainability. Specifically, Fig. 7, 8, and 9 visualize the attention weights from multi-aspects and Fig. 10 visualizes some of the most influential sensors of Lastjomer-Full2 and Lastjomer-Full2 w/o Conv on the METR-LA dataset because the linear attention fails to generate attention weights. Lastjomer-Full2 w/o Conv replaces the convolution operation in spaio-temporal joint attention with the fully connected layers. From the results in Fig. 7, 8, 9, and 10, we have the following observations:

1) Correctness: For the temporal dimension, as shown in Fig. 7, the attention weights of Lastjomer-Full2 w/o Conv are hard to distinguish because the numerical value of normalized traffic speed is always similar. The weights in the green rectangle of its corresponding time series in black circles learned through our Lastjomer-Full2 show that our Lastjomer-Full2 can match sensors according to the local temporal trends. For the spatial dimension, as shown in Fig. 8, we can find that the most important sensors of sensor 12 learned from Lastjomer-Full2 and Lastjomer-Full2 w/o Conv are different. This is because Lastjomer-Full2 w/o Conv calculates attention weights only through numerical values and ignores the essential spatial structure in the traffic forecasting, *i.e.*, Lastjomer-Full2 w/o Conv fails to extract the information that spatially adjacent sensors 12 and 15 will be influenced by the same congestion resulting in varying degrees of speed reduction. Therefore, our Lastjomer can correctly match sensors with the assistant of the locality spatio-temporal information.

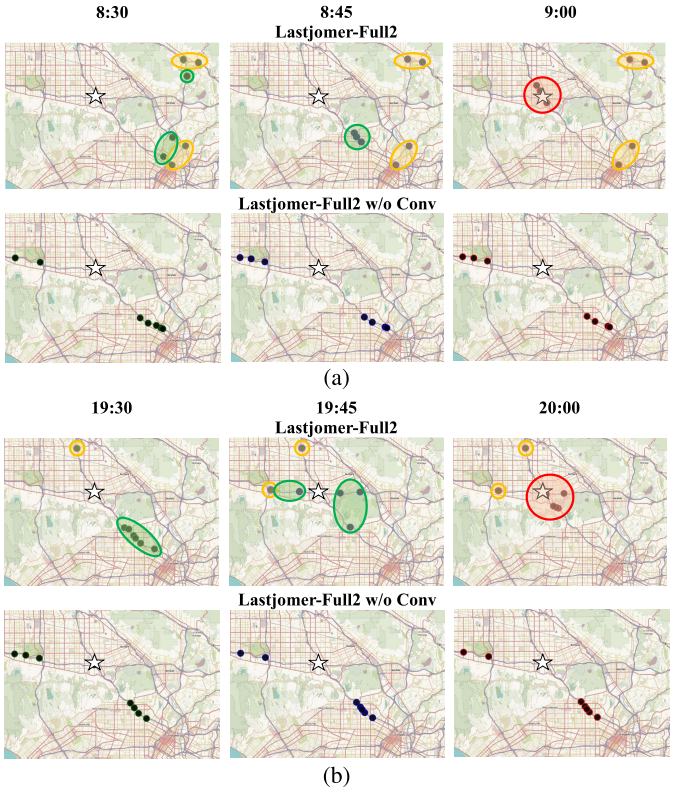


Fig. 10. (a) Influential sensors (black dots) at 9:00, 8:45, and 8:30 AM for the sensor 73 (white star) in METR-LA at 9:00 AM. (b) Influential sensors (black dots) at 8:00, 7:45, and 7:30 PM for the sensor 73 (white star) at 8:00 PM.

2) Locality: To show our Lastjomer can reveal more local information compared with vanilla self-attention, we show the learned average attention weights of all sensors in the Fig. 9. We can observe that curves of Lastjomer-Full2 show obvious locality-aware ability, that is, attention weights will become smaller and smaller as time goes backward and the distance increases, while Lastjomer-Full2 w/o Conv fails to effectively obtain local information. This is because we inject spatial and temporal local information into our STJA through the causal convolution and MHDCN.

3) Cross-Sensor Temporal Correlations: As shown in Fig. 10, the observed correlations between sensor 73 and the same sensors in yellow circles at different time slots indicate that we endow our model with the capability of capturing cross-sensor temporal dependencies, *i.e.*, can learn temporal trends from sensors that have similar functions.

4) Cross-Time Spatial Correlations: As shown in Fig. 10, the observed correlations between sensor 73 and different sensors in green circles at different time slots indicate that we endow our model with the capability of capturing cross-time spatial dependencies, *i.e.*, can gain spatial information from sensors that need time to pass their influence.

VI. Conclusion

In this paper, we propose a novel locality-aware spaio-temporal Transformer network for traffic forecasting. We adopt the elaborately designed spatio-temporal joint attention in

the encoder-decoder Transformer architecture to extract all dynamic correlations among traffic data, especially the cross-time spatial dependence and the cross-sensor temporal dependence. To alleviate the local insensitivity of the vanilla self-attention, we replace the fully-connected layer with the causal convolution and multi-hop diffusion convolution to produce queries and keys in the self-attention. Moreover, the linear attention mechanism is also used in our model to afford memory usage. In the future, we would like to extend our model for other spatio-temporal forecasting tasks, such as crime and weather forecasting.

REFERENCES

[1] B. Williams and L. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results," *J. Transp. Eng.*, vol. 129, no. 6, pp. 664–672, Nov./Dec. 2003.

[2] C.-H. Wu, J.-M. Ho, and D. T. Lee, "Travel-time prediction with support vector regression," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 276–281, Dec. 2004.

[3] Z. Zhao *et al.*, "LSTM network: A deep learning approach for short-term traffic forecast," *IET Intell. Transp. Syst.*, vol. 11, no. 2, pp. 68–75, 2017.

[4] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–16.

[5] 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)*, Stockholm, Sweden, Jul. 2018, pp. 3634–3640.

[6] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*.

[7] L. Wu, M. Haynes, A. Smith, T. Chen, and X. Li, "Generating life course trajectory sequences with recurrent neural networks and application to early detection of social disadvantage," in *Proc. 13th Int. Conf. Adv. Data Mining Appl. (ADMA)* (Lecture Notes in Computer Science), vol. 10604. Singapore: Springer, Nov. 2017, pp. 225–242.

[8] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015, *arXiv:1511.08458*.

[9] 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. (IJCAI)*, Macao, China, Aug. 2019, pp. 1907–1913.

[10] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 753–763.

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

[12] H. Yan, X. Ma, and Z. Pu, "Learning dynamic and hierarchical traffic spatiotemporal features with transformer," *IEEE Trans. Intell. Transp. Syst.*, early access, Aug. 20, 2021, doi: 10.1109/TITS.2021.3102983.

[13] Y. Qin, F. Zhao, Y. Fang, H. Luo, and C. Wang, "Memory attention enhanced graph convolution long short-term memory network for traffic forecasting," *Int. J. Intell. Syst.*, vol. 37, no. 9, pp. 6555–6576, 2022.

[14] C. Park *et al.*, "ST-GRAT: A novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manag.*, Oct. 2020, pp. 1215–1224.

[15] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.

[16] S. Li *et al.*, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

[17] K. Choromanski *et al.*, "Rethinking attention with performers," 2020, *arXiv:2009.14794*.

[18] Z. Lu, C. Zhou, J. Wu, H. Jiang, and S. Cui, "Integrating Granger causality and vector auto-regression for traffic prediction of large-scale WLANs," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 1, pp. 136–151, 2016.

[19] H. Van Lint and C. Van Hinsbergen, "Short-term traffic and travel time prediction models," *Artif. Intell. Appl. Crit. Transp.*, vol. 22, pp. 22–41, Nov. 2012.

[20] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. London, U.K.: Pearson, 2012.

[21] E. D. Liddy, "Natural language processing," in *Encyclopedia of Library and Information Science*, D. Marcel, Ed., 2nd ed. New York, NY, USA: Marcel Decker, 2001.

[22] F. J. Owens, *Signal Processing of Speech*. New York, NY, USA: Macmillan International Higher Education, 1993.

[23] D. J. Hand and N. M. Adams, "Data mining," in *Wiley StatsRef: Statistics Reference Online*. Chichester, U.K.: Wiley, 2014, pp. 1–7.

[24] Z. Lv, J. Xu, K. Zheng, H. Yin, P. Zhao, and X. Zhou, "LC-RNN: A deep learning model for traffic speed prediction," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, p. 27.

[25] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *Proc. 31st Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, Nov. 2016, pp. 324–328.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1724–1734.

[28] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "DNN-based prediction model for spatio-temporal data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Oct. 2016, pp. 1–4.

[29] Y. Liu, H. Zheng, X. Feng, and Z. Chen, "Short-term traffic flow prediction with Conv-LSTM," in *Proc. 9th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2017, pp. 1–6.

[30] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, vol. 31, no. 1, pp. 1–7.

[31] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.

[32] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1993–2001.

[33] C. Cheng, C. Zhang, Y. Wei, and Y.-G. Jiang, "Sparse temporal causal convolution for efficient action modeling," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 592–600.

[34] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," in *Proc. Adv. Neural Inf. Process. Syst., 33rd Annu. Conf. Neural Inf. Process. Syst.*, Dec. 2020, pp. 1–12.

[35] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.

[36] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Lisbon, Portugal, 2015, pp. 1412–1421.

[37] P. F. Brown *et al.*, "A statistical approach to machine translation," *Comput. Linguistics*, vol. 16, no. 2, pp. 79–85, 1990.

[38] D. Ravichandran and E. Hovy, "Learning surface text patterns for a question answering system," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2001, pp. 41–47.

[39] A. D. Baxevanis, G. D. Bader, and D. S. Wishart, *Bioinformatics*. Hoboken, NJ, USA: Wiley, 2020.

[40] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," 2019, *arXiv:1904.10509*.

[41] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," 2020, *arXiv:2004.05150*.

[42] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–12.

[43] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, "Efficient content-based sparse attention with routing transformers," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 53–68, Feb. 2021.

[44] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," 2020, *arXiv:2006.04768*.

[45] S.-H. Teng, "Scalable algorithms for data and network analysis," *Found. Trends Theor. Comput. Sci.*, vol. 12, nos. 1–2, pp. 1–274, 2016.

[46] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are RNNs: Fast autoregressive transformers with linear attention," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5156–5165.

[47] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–10.
[49] S. Zhang, Y. Guo, P. Zhao, C. Zheng, and X. Chen, "A graph-based temporal attention framework for multi-sensor traffic flow forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 7743–7758, Jul. 2022.
[50] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 933–941.
[51] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-I. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5453–5462.
[52] S. Abu-El-Haija *et al.*, "MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 21–29.

Yuchen Fang received the B.S. degree from the School of Information Science and Technology, Beijing Forestry University, Beijing, China. He is currently pursuing the M.S. degree with the School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing. His current interests include traffic forecasting and graph neural networks.

Fang Zhao received the B.S. degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 1990, and the M.S. and Ph.D. degrees in computer science and technology from the Beijing University of Posts and Telecommunication, Beijing, China, in 2004 and 2009, respectively. She is currently a Professor with the School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications. Her research interests include mobile computing, location-based services, and computer networks.

Yanjun Qin is currently pursuing the Ph.D. degree with the School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing, China. Her main current interests include location-based services, pervasive computing, convolution neural networks, and machine learning. She is mainly involved in traffic pattern recognition related project research and implementation.

Haiyong Luo (Member, IEEE) received the B.S. degree from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, China, in 1989, the M.S. degree from the School of Information and Communication Engineering, Beijing University of Posts and Telecommunication China, Beijing, China, in 2002, and the Ph.D. degree in computer science from the University of Chinese Academy of Sciences, Beijing, in 2008. He is currently an Associate Professor with the Institute of Computer Technology, Chinese Academy of Science, Beijing. His main research interests are location-based services, pervasive computing, mobile computing, and the Internet of Things.

Chenxing Wang (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with the School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing, China. His main current interests include spatial-temporal data mining, travel time estimation, traffic flow prediction, and transportation mode detection using deep learning techniques.