Attention-Based Spatial-Temporal Graph Convolutional Recurrent Networks for Traffic Forecasting

Haiyang Liu[1], Chunjiang Zhu[2], Detian Zhang[1](✉), and Qing Li[3]

[1] Institute of Artificial Intelligence, Department of Computer Science and Technology, Soochow University, Suzhou, China
20215227052@stu.suda.edu.cn, detian@suda.edu.cn

[2] Department of Computer Science, UNC Greensboro, Greensboro, NC, USA
chunjiang.zhu@uncg.edu

[3] Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
qing-prof.li@polyu.edu.hk

Abstract. Traffic forecasting is one of the most fundamental problems in transportation science and artificial intelligence. The key challenge is to effectively model complex spatial-temporal dependencies and correlations in modern traffic data. Existing methods, however, cannot accurately model both long-term and short-term temporal correlations simultaneously, limiting their expressive power on complex spatial-temporal patterns. In this paper, we propose a novel spatial-temporal neural network framework: Attention-based Spatial-Temporal Graph Convolutional Recurrent Network (*ASTGCRN*), which consists of a graph convolutional recurrent module (*GCRN*) and a global attention module. In particular, *GCRN* integrates gated recurrent units and adaptive graph convolutional networks for dynamically learning graph structures and capturing spatial dependencies and local temporal relationships. To effectively extract global temporal dependencies, we design a temporal attention layer and implement it as three independent modules based on multi-head self-attention, transformer, and informer respectively. Extensive experiments on five real traffic datasets have demonstrated the excellent predictive performance of all our three models with all their average *MAE*, *RMSE* and *MAPE* across the test datasets lower than the baseline methods.

Keywords: Traffic forecasting · Graph convolutional networks · Attention mechanism

1 Introduction

With the development of urbanization, the diversification of transportation modes and the increasing number of transportation vehicles (cabs, electric vehicles, shared bicycles, *etc.*) have put tremendous pressure on urban transportation systems, which has led to large-scale traffic congestions that have become

X. Yang et al. (Eds.): ADMA 2023, LNAI 14176, pp. 630–645, 2023.
https://doi.org/10.1007/978-3-031-46661-8_42

a common phenomenon. Traffic congestion has brought serious economic and environmental impacts to cities in various countries, and early intervention in traffic systems based on traffic forecasting is one of the effective ways to alleviate traffic congestion. By accurately predicting future traffic conditions, it provides a reference basis for urban traffic managers to make proper decisions in advance and improve traffic efficiency.

Traffic forecasting is challenging since traffic data are complex, highly dynamic, and correlated in both spatial and temporal dimensions. Recently, deep learning has dominated the field of traffic forecasting due to their capability to model complex non-linear patterns in traffic data. Many works used different deep learning networks to model dynamic local and global spatial-temporal dependencies and achieve promising prediction performance. On the one hand, they often used Recurrent Neural Networks (*RNN*) and the variants such as Long Short-Term Memory (*LSTM*) [14] and Gated Recurrent Units (*GRU*) [6] for *temporal* dependency modeling [1,2,5,22]. Some other studies used Convolutional Neural Networks (*CNN*) [15,28,30,31] or attention mechanisms [12,32] to efficiently extract temporal features in traffic data. On the other hand, Graph Convolutional Networks (*GCNs*) [21–23,29] are widely used to capture complex *spatial* features and dependencies in traffic road network data.

However, *RNN*/*LSTM*/*GRU*-based models can only indirectly model sequential temporal dependencies, and their internal cyclic operations make them difficult to capture long-term global dependencies [20]. To capture global information, *CNN*-based models [29,30] stack multiple layers of spatial-temporal modules but they may lose local information. The attention mechanism, though effective in capturing global dependencies, is not good at making short-term predictions [32]. Most of the previous attention-based methods [12,13] also have complex structures and thus high computational complexity. For instance in [13], the prediction of architectures built by multi-layer encoder and decoder, though excellent, is much slower than most prediction models by 1 or 2 orders of magnitude. Furthermore, most of the current *GCN*-based methods [11] need to pre-define a static graph based on inter-node distances or similarity to capture spatial information. However, the constructed graph needs to satisfy the static assumption of the road network, and cannot effectively capture complex dynamic spatial dependencies. Moreover, it is difficult to adapt graph-structure-based spatial modeling in various spatial-temporal prediction domains without prior knowledge (*e.g.*, inter-node distances). Therefore, effectively capturing dynamic spatial-temporal correlations and fully considering long-term and short-term dependencies are crucial to further improve the prediction performance.

To fully capture local and global spatial-temporal dependencies from traffic data, in this paper we propose a novel spatial-temporal networks framework: Attention-based Spatial-temporal Graph Convolutional Recurrent Network (*ASTGCRN*). It consists of a graph convolution recurrent module (*GCRN*) and a global attention module. Our main contributions are summarized as follows:

- We develop a novel spatial-temporal neural network framework, called *ASTGCRN*, that can effectively model dynamic local and global spatial-temporal dependencies in a traffic road network.
- In *ASTGCRN*, we devise an adaptive graph convolutional network with signals at different depths convoluted and then incorporate it into the *GRU*. The obtained *GRU* with adaptive graph convolution can well capture the *dynamic graph structures, spatial features, and local temporal* dependencies.
- We propose a general attention layer that accepts inputs from *GCRN* at different time points and captures the *global temporal* dependencies. We implement the layer using multi-head self-attention, transformer, and informer to generate three respective models.
- Extensive experiments have been performed on five real-world traffic datasets to demonstrate the superior performance of all our three models compared with the current state of the art. In particular, our model with transformer improves the average *MAE*, *RMSE*, and *MAPE* (across the tested datasets) by 0.21, 0.37, and 0.28, respectively. We carry out an additional experiment to show the generalizability of our proposed models to other spatial-temporal learning tasks.

2 Related Work

2.1 Traffic Forecasting

Traffic forecasting originated from univariate time series forecasting. Early statistical methods include Historical Average (*HA*), Vector Auto-Regressive (*VAR*) [34] and Auto-Regressive Integrated Moving Average (*ARIMA*) [19,27], with the *ARIMA* family of models the most popular. However, most of these methods are linear, need to satisfy stationary assumptions, and cannot handle complex non-linear spatial-temporal data.

With the rise of deep learning, it has gradually dominated the field of traffic forecasting by virtue of the ability to capture complex non-linear patterns in spatial-temporal data. RNN-based and CNN-based deep learning methods are the two mainstream directions for modeling temporal dependence. Early RNN-based methods such as *DCRNN* [22] used an encoder-decoder architecture with pre-sampling to capture temporal dependencies, but the autoregressive computation is difficult to focus on long-term correlations effectively. Later, the attention mechanism has been used to improve predictive performance [12,13,26,32]. In CNN-based approaches, the combination of 1-D temporal convolution *TCN* and graph convolution [29,30] are commonly used. But CNN-based models require stacking multiple layers to expand the perceptual field. The emergence of *GCNs* has enabled deep learning models to handle non-Euclidean data and capture implicit spatial dependencies, and they have been widely used for spatial data modeling [15,23]. The static graphs pre-defined according to the distance or similarity between nodes cannot fully reflect the road network information, and cannot make dynamic adjustments during the training process to effectively capture complex spatial dependencies. Current research overcame the limitations of convolutional networks based on static graphs or single graphs, and more adaptive graph or dynamic graph building strategies [2,18,28] were proposed.

In addition to the above methods, differential equations have also been applied to improve traffic forecasting. [10] capture spatial-temporal dynamics through a tensor-based Ordinary Differential Equation (*ODE*) alternative to graph convolutional networks. [7] introduced Neural Control Differential Equations (*NCDEs*) into traffic prediction, which designed two *NCDEs* for temporal processing and spatial processing respectively. Although there are dense methods for spatial-temporal modeling, most of them lack the capability to focus on both long-term and short-term temporal correlations, which results in the limitations of capturing temporal dependencies and road network dynamics.

2.2 Graph Convolutional Networks

Graph convolution networks can be separated into spectral domain graph convolution and spatial domain graph convolution. In the field of traffic prediction, spectral domain graph convolution has been widely used to capture the spatial correlation between traffic series. [3] for the first time proposed spectral domain graph convolution based on spectral graph theory. The spatial domain signal is converted to the spectral domain by Fourier transform, and then the convolution result is inverted to the spatial domain after completing the convolution operation. The specific formula is defined as follows:

$$g_\theta \star_G x = g(L)x = U g_\theta(\Lambda) U^{\mathbf{T}} x, \tag{1}$$

In the equation, $\star_G$ denotes the graph convolution operation between the convolution kernel g_θ and the input signal x and $L = D^{-\frac{1}{2}} \mathbf{L} D^{-\frac{1}{2}} = U \Lambda U^{\mathbf{T}} \in \mathbb{R}^{N \times N}$ is the symmetric normalized graph Laplacian matrix, where $\mathbf{L} = D - A$ is the graph Laplacian matrix and $D = diag(\sum_{j=1}^{N} A_{1j}, \cdots, \sum_{j=1}^{N} A_{Nj}) \in \mathbb{R}^{N \times N}$ is the diagonal degree matrix. U is the Fourier basis of G and Λ is the diagonal matrix of $\mathbf{L}$ eigenvalues. However, the eigenvalue decomposition of the Laplacian matrix in Eq. (1) requires expensive computations. For this reason, [8] uses the Chebyshev polynomial to replace the convolution kernel g_θ in the spectral domain:

$$g_\theta \star_G x = g(L)x = \sum_{k=0}^{K-1} \beta_k T_k(\hat{L}) x, \tag{2}$$

where $[\beta_0, \beta_1, \ldots, \beta_{K-1}]$ are the learnable parameters, and $K \geq 1$ is the number of convolution kernels. ChebNet does not require eigenvalue decomposition of Laplacian matrices, but uses Chebyshev polynomials $T_0(\hat{L}) = I_n, T_1(\hat{L}) = \hat{L}$, and $T_{n+1}(\hat{L}) = 2\hat{L}T_n(\hat{L}) - T_{n-1}(\hat{L})$. Here $\hat{L} = \frac{2}{\lambda_{max}} L - I_n$ is the scaled Laplacian matrix, where λ_{max} is the largest eigenvalue and I_n is the identity matrix. When $K = 2$, ChebNet is simplified to GCN [17].

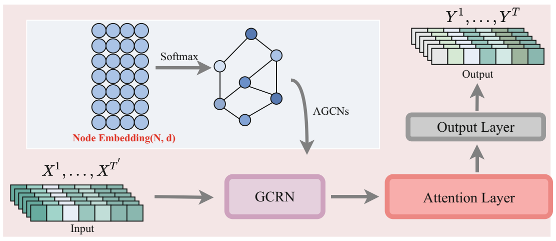


Fig. 1. Detailed framework of the *ASTGCRN* model.

3 Methodology

In this section, we first give the mathematical definition of the traffic prediction problem, and then describe in detail the implementation of the *ASTGCRN* framework (see Fig. 1): *GCRN* and the attention layer.

3.1 Problem Definition

The traffic prediction task can be formulated as a multi-step time series prediction problem that utilizes historical traffic data and prior knowledge of N locations (*e.g.*, traffic sensors) on a road network to predict future traffic conditions. Typically, prior knowledge refers to the road network represented as a graph $G = (V, E, \mathbf{A})$, where V is a set of $N = |V|$ nodes representing different locations on the road network, E is a set of edges, and $\mathbf{A} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$ is the weighted adjacency matrix representing the proximity between nodes (*e.g.*, the road network between nodes). We can formulate the traffic prediction problem as learning a function F to predict the graph signals $Y^{(t+1):(t+T)} \in \mathbb{R}^{T\times N\times C}$ of the next T steps based on the past T' steps graph signals $X^{(t-T'+1):t} \in \mathbb{R}^{T'\times N\times C}$ and G:

$$[X^{(t-T'+1):t}, G] \xrightarrow{F_\Theta} [X^{(t+1):(t+T)}], \tag{3}$$

where Θ denotes all the learnable parameters in the model.

3.2 Adaptive Graph Convolution

For traffic data in a road network, the dependencies between different nodes may change over time, and the pre-defined graph structure cannot contain complete spatial dependency information. Inspired by the adaptive adjacency matrix [2, 5, 29], we generate $T_1(\hat{L})$ in Eq. (2) by randomly initializing a learnable node embedding $E_\phi \in \mathbb{R}^{N\times D_e}$, where D_e denotes the size of the node embedding:

$$T_1(\hat{L}) = \hat{L} = softmax(E_\phi \cdot E_\phi^{\mathbf{T}}) \tag{4}$$

To explore the hidden spatial correlations between node domains at different depths, we generalize to high-dimensional graph signals $X \in \mathbb{R}^{N\times C_{in}}$ and concatenate $T_k(\hat{L})$ at different depths as a tensor $\tilde{T_\phi} = [I, T_1(\hat{L}), \ldots, T_{K-1}(\hat{L})]^{\mathbf{T}} \in \mathbb{R}^{K\times N\times N}$.

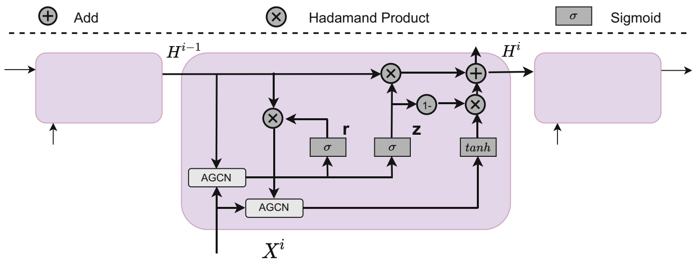


Fig. 2. The Graph Convolution Recurrent Module.

Let C_{in} and C_{out} represent the number of input and output channels, respectively. Then the graph convolution formula in Eq. (2) can be refined as:

$$g_\theta \star_G x = g(L)x = \tilde{T}_\phi X \Psi, \tag{5}$$

where the learnable parameters $\Psi \in \mathbb{R}^{K \times C_{in} \times C_{out}}$. However, the parameters shared by all nodes have limitations in capturing spatial dependencies [2]. Instead, we assign independent parameters to each node to get the parameters $\hat{\Psi} \in \mathbb{R}^{N \times K \times C_{in} \times C_{out}}$, which can more effectively capture the hidden information in different nodes. We further avoid overfitting and high spatial complexity problems by matrix factorization. That is to learn two smaller parameters to generate $\hat{\Psi} = E_\phi W$, where $E_\phi \in \mathbb{R}^{N \times D_e}$ is the node embedding dictionary and $W \in \mathbb{R}^{D_e \times K \times C_{in} \times C_{out}}$ are the learnable weights. Our adaptive graph convolution formula can be expressed as:

$$g_\theta \star_G x = g(L)x = \tilde{T}_\phi X E_\phi W \in \mathbb{R}^{N \times C_{out}} \tag{6}$$

3.3 GRU with Adaptive Graph Convolution

GRU is a simplified version of *LSTM* with multiple *GRUCell* modules and generally provides the same performance as *LSTM* but is significantly faster to compute. To further discover the spatial-temporal correlation between time series, we replace the *MLP* layers in *GRU* with adaptive graph convolution operation, named *GCRN*. The computation of *GCRN* is given as follows:

$$\begin{aligned}
z^t &= \sigma(\tilde{T}_\phi [X^t, h^{t-1}] E_\phi W_z + E_\phi b_z), \\
r^t &= \sigma(\tilde{T}_\phi [X^t, h^{t-1}] E_\phi W_r + E_\phi b_r), \\
\tilde{h}^t &= tanh(\tilde{T}_\phi [X^t, r^t \odot h^{t-1}] E_\phi W_{\tilde{h}} + E_\phi b_{\tilde{h}}), \\
h^t &= z^t \odot h^{t-1} + (1 - z^t) \odot \tilde{h}^t,
\end{aligned} \tag{7}$$

where $W_z, W_r, W_{\tilde{h}}, b_z, b_r$ and $b_{\tilde{h}}$ are learnable parameters, σ and $tanh$ are two activation functions, *i.e.*, the Sigmoid function and the Tanh function. The $[X^t, h^{t-1}]$ and h^t are the input and output at time step t, respectively. The network architecture of *GCRN* is plotted in Fig. 2.

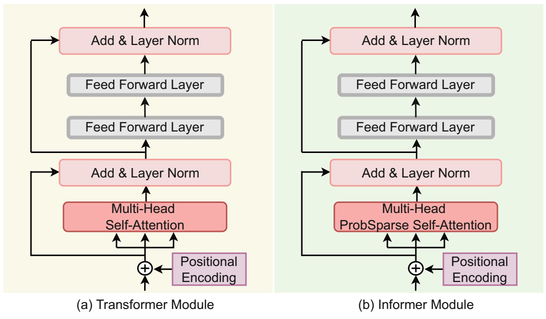


Fig. 3. The Attention Layer.

3.4 The Attention Layer

GCRN can effectively capture sequential dependencies, but its structural characteristics limit its ability to capture long-distance temporal information. For the traffic prediction task, the global temporal dependence clearly has a significant impact on the learning performance. Self-attention directly connects two time steps through dot product calculation, which greatly shortens the distance between long-distance dependent features, and improves the parallelization of computation, making it easier to capture long-term dependencies in traffic data. Therefore, we propose three independent modules for the self-attention mechanism, namely, the multi-headed self-attention module, the transformer module, and the informer module, in order to directly capture global temporal dependencies. In the following three subsections, we will explain these three modules in detail.

Multi-Head Self-attention Module. Multi-head attention is to learn the dependencies of different patterns in parallel with multiple sets of queries, keys and values (where each set is regarded as an attention head), and then concatenate the learned multiple relationships as the output. We use a self-attentive mechanism to construct the multi-headed attention module. Specifically, $Q_o = H_o W_q$, $K_o = H_o W_k$ and $V_o = H_o W_v$ are derived from the same matrix H_o by linear transformation. Here $H_o \in \mathbb{R}^{N \times T \times C_{out}}$ is the output result of the *GCRN* module, and $W_q \in \mathbb{R}^{C_{out} \times d_q}$, $W_k \in \mathbb{R}^{C_{out} \times d_k}$ and $W_v \in \mathbb{R}^{C_{out} \times d_v}$ are the learnable parameters of the linear projection. For multi-head self-attention mechanism, the formula can be stated as:

$$\begin{aligned} MHSelfAtt &= Concat(head_1, \ldots, head_h) W_o, \\ \text{where } head_i &= Att(Q_o, K_o, V_o) \\ &= softmax(\frac{Q_o {K_o}^{\mathbf{T}}}{\sqrt{d}}) V_o. \end{aligned} \tag{8}$$

Transformer Module. The Transformer module (see Fig. 3(a)) contains a multi-head self-attention layer and two feed-forward neural networks. For self-attention, each position of the input sequence is equally inner product, which results in the loss of sequential information. We use a fixed position encoding [25] to address this flaw:

$$\begin{aligned} PE_t(2c) &= sin(t/1000^{2c/C_{out}}), \\ PE_t(2c+1) &= cos(t/1000^{2c/C_{out}}), \end{aligned} \tag{9}$$

where t is the relative position of each sequence (time step) of the input and c represents the dimension. In order to better identify the relative positional relationship between sequences, H_o and position encoding are combined to generate $H_o^{'} \in \mathbb{R}^{N \times T \times C_{out}}$:

$$H_o^{'}[:, t, :] = H_o[:, t, :] + PE_t \tag{10}$$

After combining the location encoding, $H_o^{'}$ is fed as input into the multi-headed self-attentive layer for remote relationship capture, and then the output state is passed to the two fully connected layers. Layer normalization and residual connectivity are used in both sub-layers. Finally, Transformer module outputs the result $H_a \in \mathbb{R}^{N \times T \times C_{out}}$.

Informer Module. According to Eq. (8), the traditional self-attention mechanism requires two dot products and $O(T^2)$ space complexity. The sequence lengths of queries and keys are equal in self-attention computation, *i.e.*, $T_q = T_k = T$. After finding that most of the dot products have minimal attention and the main attention is focused on only a few dot products, [33] proposed ProbSparse self-attention. ProbSparse self-attention selects only the more important queries to reduce the computational complexity, *i.e.*, by measuring the dilution of the queries and then selecting only the top-u queries with $u = c \cdot lnT$ for constant c. The query dilution evaluation formula is as follows:

$$\bar{M}\left(q_i, K_o\right) = \max_j \left\{ \frac{q_i k_j^{\mathbf{T}}}{\sqrt{d}} \right\} - \frac{1}{T} \sum_{j=1}^{T} \frac{q_i k_j^{\mathbf{T}}}{\sqrt{d}} \tag{11}$$

where q_i and k_i represent the i-th row in Q_o and K_o respectively. To compute $\bar{M}$, only $U = T \ln T$ dot product pairs are randomly selected, and the other pairs are filled with zeros. In this way, the time and space complexity are reduced to only $O(T \ln T)$. Therefore, we construct a new module called Informer module (see Fig. 3(b)) by using ProbSparse self-attention to replace the normal self-attention mechanism of Transformer module. It selects top-u query according to $\bar{M}$ to generate sparse matrix Q_o^{spa}. Then the Multi-head ProbSparse self-attention can be expressed as:

$$\begin{aligned} MHProbSelfAtt &= Concat(head_1, \ldots, head_h)W_o, \\ \text{where } head_i &= Att(Q_o^{spa}, K_o, V_o) \\ &= softmax(\frac{Q_o^{spa} {K_o}^{\mathbf{T}}}{\sqrt{d}})V_o. \end{aligned} \tag{12}$$

Table 1. Statistics of the tested datasets

Datasets	Nodes	Samples	Unit	Time Span
PEMSD3	358	$26,208$	5 mins	3 months
PEMSD4	307	$16,992$	5 mins	2 months
PEMSD7	883	$28,224$	5 mins	4 months
PEMSD8	170	$17,856$	5 mins	2 months
PEMSD7(M)	228	$12,672$	5 mins	2 months
DND-US	53	313	1 week	6 years

We choose the $L1$ loss to formulate the objective function and minimize the training error by back propagation. Specifically, the loss function is defined as follows.

$$Loss = \frac{1}{T}\sum_{t=0}^{T-1}(\hat{Y}^t - Y^t) \tag{13}$$

where $\hat{Y}$ is the real traffic data, Y is the predicted data, and T is the total predicted time steps.

4 Experimental Results

In this section, we present the results of the extensive experiments we have performed. We start by describing the experimental setups and then discuss the prediction results obtained in the baseline settings. Finally, the ablation study and the effects of hyperparameter tuning are provided.

Datasets. We evaluate the performance of the developed models on five widely used traffic prediction datasets collected by Caltrans Performance Measure System [4], namely *PEMSD3*, *PEMSD4*, *PEMSD7*, *PEMSD8*, and *PEMSD7(M)* [7,10]. The traffic data are aggregated into 5-minute time intervals, *i.e.*, 288 data points per day. In addition, we construct a new US natural death dataset *DND-US* to study the generalizability of our method to other spatial-temporal data. It contains weekly natural deaths for 53 (autonomous) states in the US for the six years from 2014 to 2020. Following existing works [2], the Z-score normalization method is adopted to normalize the input data to make the training process more stable. Detailed statistics for the tested datasets are summarized in Table 1.

Baseline Methods. We compare our models with the following baseline methods:

- Traditional time series forecasting methods, Historical Average (*HA*), *ARIMA* [27], *VAR* [34], and *SVR* [9];
- *RNN*-based models: *FC-LSTM* [24], *DCRNN* [22], *AGCRN* [2], and *Z-GCNETs* [5];
- *CNN*-based methods: *STGCN* [30], *Graph WaveNet* [29], *MSTGCN*, *LSGCN* [15], *STSGCN* [23], and *STFGNN* [21];

Table 2. Performance comparison of different models on the tested datasets. Underlined results represent the current best among existing methods. Our three models outperform *almost all* baseline methods, as shown in bold font.

Model	PEMSD3			PEMSD4			PEMSD7			PEMSD8			PEMSD7(M)		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HA	31.58	52.39	33.78%	38.03	59.24	27.88%	45.12	65.64	24.51%	34.86	59.24	27.88%	4.59	8.63	14.35%
ARIMA	35.41	47.59	33.78%	33.73	48.80	24.18%	38.17	59.27	19.46%	31.09	44.32	22.73%	7.27	13.20	15.38%
VAR	23.65	38.26	24.51%	24.54	38.61	17.24%	50.22	75.63	32.22%	19.19	29.81	13.10%	4.25	7.61	10.28%
SVR	20.73	34.97	20.63%	27.23	41.82	18.95%	32.49	44.54	19.20%	22.00	33.85	14.23%	3.33	6.63	8.53%
FC-LSTM	21.33	35.11	23.33%	26.77	40.65	18.23%	29.98	45.94	13.20%	23.09	35.17	14.99%	4.16	7.51	10.10%
DCRNN	17.99	30.31	18.34%	21.22	33.44	14.17%	25.22	38.61	11.82%	16.82	26.36	10.92%	3.83	7.18	9.81%
AGCRN	15.98	28.25	15.23%	19.83	32.26	12.97%	22.37	36.55	9.12%	15.95	25.22	10.09%	2.79	5.54	7.02%
Z-GCNETs	16.64	28.15	16.39%	19.50	31.61	12.78%	21.77	35.17	9.25%	15.76	25.11	10.01%	2.75	5.62	6.89%
STGCN	17.55	30.42	17.34%	21.16	34.89	13.83%	25.33	39.34	11.21%	17.50	27.09	11.29%	3.86	6.79	10.06%
Graph WaveNet	19.12	32.77	18.89%	24.89	39.66	17.29%	26.39	41.50	11.97%	18.28	30.05	12.15%	3.19	6.24	8.02%
MSTGCN	19.54	31.93	23.86%	23.96	37.21	14.33%	29.00	43.73	14.30%	19.00	29.15	12.38%	3.54	6.14	9.00%
LSGCN	17.94	29.85	16.98%	21.53	33.86	13.18%	27.31	41.46	11.98%	17.73	26.76	11.20%	3.05	5.98	7.62%
STSGCN	17.48	29.21	16.78%	21.19	33.65	13.90%	24.26	39.03	10.21%	17.13	26.80	10.96%	3.01	5.93	7.55%
STFGNN	16.77	28.34	16.30%	20.48	32.51	16.77%	23.46	36.60	9.21%	16.94	26.25	10.60%	2.90	5.79	7.23%
ASTGCN(r)	17.34	29.56	17.21%	22.93	35.22	16.56%	24.01	37.87	10.73%	18.25	28.06	11.64%	3.14	6.18	8.12%
DSTAGNN	15.57	27.21	14.68%	19.30	31.46	12.70%	21.42	34.51	9.01%	15.67	24.77	9.94%	2.75	5.53	6.93%
STGODE	16.50	27.84	16.69%	20.84	32.82	13.77%	22.59	37.54	10.14%	16.81	25.97	10.62%	2.97	5.66	7.36%
STG-NCDE	15.57	27.09	15.06%	19.21	31.09	12.76%	20.53	33.84	8.80%	15.45	24.81	9.92%	2.68	5.39	6.76%
A-ASTGCRN	**15.06**	**26.71**	**13.83%**	19.30	**30.92**	12.91%	**20.42**	**33.81**	**8.54%**	15.46	**24.54**	**9.89%**	**2.66**	**5.36**	**6.72%**
I-ASTGCRN	**15.06**	**26.40**	**13.91%**	**19.15**	**30.80**	12.89%	20.81	**33.83**	8.95%	**15.26**	**24.53**	**9.65%**	**2.63**	**5.30**	**6.60%**
T-ASTGCRN	**14.90**	**26.01**	**14.17%**	**19.21**	**31.05**	**12.67%**	**20.53**	**33.75**	**8.73%**	**15.14**	**24.24**	**9.63%**	**2.63**	**5.32**	**6.66%**

– Attention-based models: *ASTGCN(r)* [12], and *DSTAGNN* [18];
– Other types of models: *STGODE* [10] and *STG-NCDE* [7].

Experimental Settings. All datasets are split into training set, validation set and test set in the ratio of 6:2:2. Our model and all baseline methods use the 12 historical continuous time steps as input to predict the data for the next 12 continuous time steps.

Our models are implemented based on the Pytorch framework, and all the experiments are performed on an NVIDIA GeForce GTX 1080 TI GPU with 11G memory. The following hyperparameters are configured based on the models' performance on the validation dataset: we train the model with 300 epochs at a learning rate of 0.003 using the Adam optimizer [16] and an early stop strategy with a patience number of 15. The code is available at https://github.com/Liuhy-666/ASTGCRN.git.

Three common prediction metrics, Mean Absolute Error (*MAE*), Root Mean Square Error (*RMSE*), and Mean Absolute Percentage Error (*MAPE*), are used to measure the traffic forecasting performance of the tested methods. In the discussions below, we refer to our specific *ASTGCRN* models based on the Multi-head self-attention module, Transformer module, and Informer module as *A-ASTGCRN*, *T-ASTGCRN*, and *I-ASTGCRN*, respectively.

4.1 Experimental Results

Table 2 shows the prediction performance of our three models together with the nineteen baseline methods on the five tested datasets. Remarkably, our three models outperform almost all the baseline methods in prediction on all the datasets. Table 3 lists the training time (s/epoch), inference time (s/epoch) and memory cost (MB) of our models, as well as several recent and best-performing baselines on the *PEMSD4* dataset.

Table 3. Computation time on *PEMSD4*.

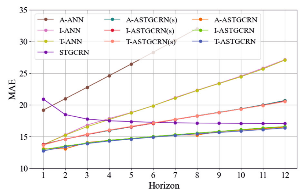
Model	Training	Inference	Memory
STGODE	111.77	12.19	8773
Z-GCNETs	63.34	7.40	8597
DSTAGNN	242.57	14.64	10347
STG-NCDE	1318.35	93.77	6091
A-ASTGCRN	45.12	5.18	7167
I-ASTGCRN	58.84	6.51	7527
T-ASTGCRN	54.80	5.62	7319

Table 4. Forecasting performance of several competitive methods on *DND-US*

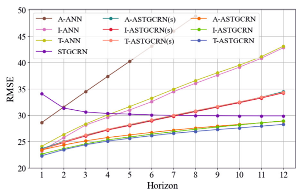
Dataset	Model	MAE	RMSE	MAPE
DND-US	AGCRN	105.97	325.09	7.49%
	DSTAGNN	47.49	73.37	7.47%
	STG-NCDE	47.70	77.30	6.13%
	A-ASTGCRN	**39.33**	**62.86***	**5.36%**
	I-ASTGCRN	**38.79***	**66.99**	**5.16%***
	T-ASTGCRN	**40.60**	**66.28**	**5.43%**

The overall prediction results of traditional statistical methods (including *HA*, *ARIMA*, *VR*, and *SVR*) are not satisfactory because of its limited ability to handle non-linear data. Their prediction performance is worse than deep learning methods by large margins. *RNN*-based methods such as *DCRNN*, *AGCRN*, and *Z-GCNRTs* suffer from the limitation of *RNNs* that cannot successfully capture long-term temporal dependence and produce worse results than our methods. *CNN*-based models such as *STGCN*, *Graph WaveNet*, *STSGCN*, *STFGCN*, and *STGODE*, have either worse or comparable performance compared to *RNN*-based methods in our empirical study. They get the 1-D *CNN* by temporal information, but the size of the convolutional kernel prevents them from capturing the complete long-term temporal correlation. Although both *ASTGCN* and *DSTAGNN* use temporal attention modules, they ignore local temporal information. *STG-NCDE* achieves currently best performance in multiple datasets. But their temporal *NCDE* using only the fully connected operation cannot pay full attention to the temporal information.

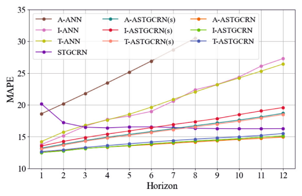
To test the generalizability of our proposed models to other spatial-temporal learning tasks, we perform an additional experiment on the *DND-US* dataset to predict the number of natural deaths in each US state. As shown in Table 4, all our three models again outperform several competitive baseline methods with significant margins.



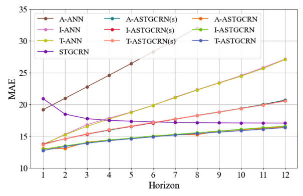
(a) *MAE* on *PEMSD3*



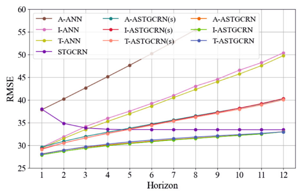
(b) *RMSE* on *PEMSD3*



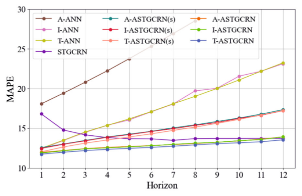
(c) *MAPE* on *PEMSD3*



(d) *MAE* on *PEMSD4*



(e) *RMSE* on *PEMSD4*

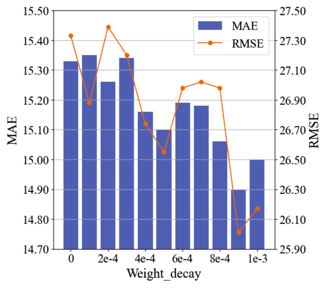


(f) *MAPE* on *PEMSD4*

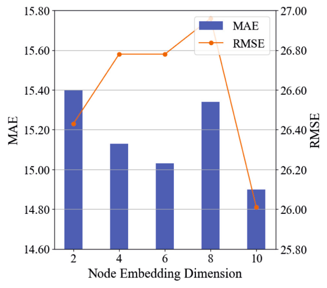
Fig. 4. Prediction performance at each horizon

4.2 Ablation and Parameter Study

Ablation Study. We refer to the model without an attention layer as *STGCRN*, and the *A-ASTGCRN*, *I-ASTGCRN* and *T-ASTGCRN* without the *GCRN* layer as *A-ANN*, *I-ANN* and *T-ANN*, respectively. Also, *A-ASTGCRN(s)*, *I-ASTGCRN(s)* and *T-ASTGCRN(s)* are variant models that use static graphs for graph convolution. we plot the detailed values of different horizons for our methods on the *PEMSD3* and *PEMSD4* datasets in Fig. 4. It shows that the prediction performance of *STGCRN* become closer to the three models as the predicted horizon increases, and the spatial modeling ability of static graphs is much lower than that of adaptive adjacency matrices. The autoregressive feature of the *GRU* model allows more spatial-temporal information to be pooled in the later time horizons, so that long-term prediction appears to be better than short-term prediction. But the performance of *STGCRN* lags behind the three attention-based models at all time horizons. The attention module is crucial for capturing long-term temporal dependencies in traffic data, further enhancing the modeling of spatio-temporal dependencies. However, only using attention modules to focus on long-term temporal dependencies and removing *GRU* using adaptive graph convolutions hurts prediction performance.



(a) Effects of weight decay



(b) Effects of (D_e)

Fig. 5. Effects of hyperparameter tuning on *T-ASTGCRN* in *PEMSD3*

Table 5. Effect of convolution kernel number K on *T-ASTGCRN*.

Dataset	K	MAE	RMSE	MAPE	Training	Inference	Memory
PEMSD3	1	15.24	26.46	15.10%	**88.94**	**9.95**	**6497**
	2	**14.90**	**26.01**	**14.17%**	95.80	10.22	7555
	3	15.33	27.04	13.92%	121.40	13.39	8535
PEMSD4	1	19.40	31.19	13.00%	**48.72**	**5.24**	**6355**
	2	**19.21**	**31.05**	**12.67%**	54.80	5.62	7319
	3	19.22	31.07	12.84%	66.43	7.12	8137

Parameter Study. To investigate the effects of hyperparameters on the prediction results, we conduct a series of experiments on the main hyperparameters. Figure 5 shows the *MAE* and *RMSE* values of *T-ASTGCRN* in the *PEMSD3* dataset when varying the weight decay and node embedding dimension D_e. It can be seen that fine-tuning the weight decay and node embedding dimension can improve the prediction performance of the model. Meanwhile, Table 5 shows the prediction performance and training cost for varying the number of convolution kernels K. From the experimental results, we can see that with $K = 1$, the graph convolution is simplified to a unit matrix-based implementation, which does not enable effective information transfer between nodes. A larger convolution depth does not improve the prediction performance, but instead incurs longer training time and memory cost. Therefore, for our model and dataset, we set K to 2.

5 Conclusion

In this paper, we design an attention-based spatial-temporal graph convolutional recurrent network framework for traffic prediction. We instantiate the framework with three attention modules based on Multi-head self-attention, Transformer and Informer, all of which, in particular the Transformer-based module, can well capture long-term temporal dependence and incorporate with the spatial and short-term temporal features by the *GCRN* module. Extensive experiments confirm the effectiveness of all our three models in improving the prediction performance. We believe that the design ideas of Transformer and Informer can bring new research thrusts in the field of traffic forecasting.

Acknowledgements. Chunjiang Zhu is supported by UNCG Start-up Funds and Faculty First Award. Detian Zhang is partially supported by the Collaborative Innovation Center of Novel Software Technology and Industrialization, the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

1. Bai, L., Yao, L., Kanhere, S.S., Yang, Z., Chu, J., Wang, X.: Passenger demand forecasting with multi-task convolutional recurrent neural networks. In: Yang, Q., Zhou, Z.-H., Gong, Z., Zhang, M.-L., Huang, S.-J. (eds.) PAKDD 2019. LNCS (LNAI), vol. 11440, pp. 29–42. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-16145-3_3
2. Bai, L., Yao, L., Li, C., Wang, X., Wang, C.: Adaptive graph convolutional recurrent network for traffic forecasting. Adv. Neural. Inf. Process. Syst. **33**, 17804–17815 (2020)
3. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
4. Chen, C., Petty, K., Skabardonis, A., Varaiya, P., Jia, Z.: Freeway performance measurement system: mining loop detector data. Transp. Res. Rec. **1748**(1), 96–102 (2001)
5. Chen, Y., Segovia, I., Gel, Y.R.: Z-gcnets: time zigzags at graph convolutional networks for time series forecasting. In: International Conference on Machine Learning, pp. 1684–1694. PMLR (2021)
6. Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014)
7. Choi, J., Choi, H., Hwang, J., Park, N.: Graph neural controlled differential equations for traffic forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 6367–6374 (2022)
8. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems 29 (2016)
9. Drucker, H., Burges, C.J., Kaufman, L., Smola, A., Vapnik, V.: Support vector regression machines. In: Advances in Neural Information Processing Systems 9 (1996)
10. Fang, Z., Long, Q., Song, G., Xie, K.: Spatial-temporal graph ode networks for traffic flow forecasting. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 364–373 (2021)
11. Geng, X., Li, Y., Wang, L., Zhang, L., Yang, Q., Ye, J., Liu, Y.: Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 3656–3663 (2019)
12. Guo, S., Lin, Y., Feng, N., Song, C., Wan, H.: Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 922–929 (2019)

13. Guo, S., Lin, Y., Wan, H., Li, X., Cong, G.: Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. IEEE Trans. Knowl. Data Eng. (2021)
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
15. Huang, R., Huang, C., Liu, Y., Dai, G., Kong, W.: Lsgcn: Long short-term traffic prediction with graph convolutional networks. In: IJCAI, pp. 2355–2361 (2020)
16. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
18. Lan, S., Ma, Y., Huang, W., Wang, W., Yang, H., Li, P.: Dstagnn: dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In: International Conference on Machine Learning, pp. 11906–11917. PMLR (2022)
19. Lee, S., Fambro, D.B.: Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting. Transp. Res. Rec. **1678**(1), 179–188 (1999)
20. Li, F., et al.: Dynamic graph convolutional recurrent network for traffic prediction: benchmark and solution. ACM Trans. Knowl. Dis. Data (TKDD) (2021)
21. Li, M., Zhu, Z.: Spatial-temporal fusion graph neural networks for traffic flow forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4189–4196 (2021)
22. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: data-driven traffic forecasting. In: International Conference on Learning Representations (ICLR 2018) (2018)
23. Song, C., Lin, Y., Guo, S., Wan, H.: Spatial-temporal synchronous graph convolutional networks: a new framework for spatial-temporal network data forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 914–921 (2020)
24. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems 27 (2014)
25. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems 30 (2017)
26. Wang, X., et al.: Traffic flow prediction via spatial temporal graph neural network. In: Proceedings of The Web Conference 2020, pp. 1082–1092 (2020)
27. Williams, B.M., Hoel, L.A.: Modeling and forecasting vehicular traffic flow as a seasonal arima process: theoretical basis and empirical results. J. Transp. Eng. **129**(6), 664–672 (2003)
28. Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., Zhang, C.: Connecting the dots: Multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 753–763 (2020)
29. Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C.: Graph wavenet for deep spatial-temporal graph modeling. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 1907–1913 (2019)
30. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 3634–3640 (2018)
31. Zhang, Q., Chang, J., Meng, G., Xiang, S., Pan, C.: Spatio-temporal graph structure learning for traffic forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 1177–1185 (2020)

32. Zheng, C., Fan, X., Wang, C., Qi, J.: Gman: a graph multi-attention network for traffic prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 1234–1241 (2020)
33. Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 11106–11115 (2021)
34. Zivot, E., Wang, J.: Vector autoregressive models for multivariate time series. Modeling financial time series with S-PLUS®, pp. 385–429 (2006)