Adaptive Joint Spatio-Temporal Graph Learning Network for Traffic Data Forecasting

TIANYI WANG and SHU-CHING CHEN, University of Missouri–Kansas City, USA

Traffic data forecasting has become an integral part of the intelligent traffic system. Great efforts are spent developing tools and techniques to estimate traffic flow patterns. Many existing approaches lack the ability to model the complex and dynamic spatio-temporal relations in the traffic data, which are crucial in capturing the traffic dynamic. In this work, we propose AJSTGL, a novel adaptive joint spatio-temporal graph learning network for traffic data forecasting. The proposed model utilizes static and adaptive graph learning modules to capture the static and dynamic spatial traffic patterns and optimize the graph learning process. A sequence-to-sequence fusion model is proposed to learn the temporal correlation and combine the output of multiple parallelized encoders. We also develop a spatio-temporal graph transformer module to complement the sequence-to-sequence fusion module by dynamically capturing the time-evolving node relations in long-term intervals. Experiments on three large-scale traffic flow datasets demonstrate that our model could outperform other state-of-the-art baseline methods.

CCS Concepts: • **Information systems** → **Spatial-temporal systems**; • **Computing methodologies** → **Neural networks**; • **Applied computing** → *Transportation;*

Additional Key Words and Phrases: Traffic forecasting, graph convolutional network, spatio-temporal data, transformer

ACM Reference format:
Tianyi Wang and Shu-Ching Chen. 2024. Adaptive Joint Spatio-Temporal Graph Learning Network for Traffic Data Forecasting. *ACM Trans. Spatial Algorithms Syst.* 10, 3, Article 26 (October 2024), 20 pages.
https://doi.org/10.1145/3634913

1 INTRODUCTION

In recent years, spatio-temporal data modeling has received increasing attention due to its prevalent appearance in many real-world applications, such as traffic data forecasting and natural disaster prediction [3, 7, 12]. In this article, we study the problem of traffic data forecasting by given historical traffic conditions. Traffic data forecasting has always been an essential part of the intelligent traffic system (ITS). Traffic networks can be formulated as graphs, where each node represents a point of interest, such as road intersections and airports. Then, the edges can be represented as the connectivity among the points of interest, such as flight routes and road segments. Conventional spatio-temporal data processing approaches utilize **Convolutional Neural**

This work was partially supported by the National Science Foundation (NSF), grant CNS2301552.
Authors' address: T. Wang and S.-C. Chen, Division of Computing, Analytics and Mathematics, University of Missouri–Kansas City, 5100 Rockhill Road, Kansas City, MO 64110; e-mails: t.wang@mail.umkc.edu, s.chen@umkc.edu.

2374-0353/2024/10-ART26 $15.00
https://doi.org/10.1145/3634913

Networks (CNNs) to extract the spatial features and use **Recurrent Neural Networks (RNNs)** to learn the temporal correlation from the input sequence [4, 26, 37]. However, the "sliding grid" style of convolution used by CNN cannot produce functional features on graph-structured data. The grid segments measured using Euclidean distance are not viable candidates to represent the inherent network structure and node dependencies in a traffic network.

Recently, **Graph Convolutional Networks (GCNs)** have seen extensive applications on graph-structured data [30, 33, 39]. GCNs apply convolution on neighboring nodes based on the adjacency/correlation matrix that formulates the network topology. However, there are still significant challenges in applying GCNs to traffic data forecasting problems. Most existing GCN-based models only use pre-defined node relationships to construct the adjacency matrix [8, 11, 42]. The pre-defined graph network is often inferred from the physical route connection and specific distance measurements. Due to the complex nature of traffic forecasting problems, such an intuitive graph structure cannot capture diverse traffic patterns. Moreover, many external factors affecting the traffic patterns, such as weather and special events, are yet fully exploited and integrated into the model. Some studies [2, 6, 40] attempted to learn the adjacency matrix using data-driven methods to circumvent the disadvantage of a pre-defined graph. However, as the prediction horizon increases, most adjacency matrices are prone to noise and fine-scale roughness, which increases the difficulty of learning complex spatio-temporal correlations.

This article addresses the preceding challenges by proposing AJSTGL, a novel adaptive joint spatio-temporal graph learning network. AJSTGL is composed of several graph learning modules. In the static graph learning module, shifted graph Laplacian is applied to expand the sensitivity of the pre-defined graph. The transformed graph Laplacian can capture more granular hidden patterns and still leverage the original graph structure's knowledge. Furthermore, we adopt node-specific dependency modeling to replace the conventional graph convolutional layer so that the model can learn node-specific patterns. In the adaptive graph learning module, the adaptive graph generation and unidirectional graph convolution methods are applied to learn the network topology in a data-driven manner. A sequence-to-sequence fusion module is developed to encode multiple graph signals in parallel and hierarchically combine them to learn the short-term temporal node dependencies. We also develop the spatio-temporal graph transformer module to complement the sequence-to-sequence fusion module by dynamically capturing the time-evolving spatial node relations in long-term time intervals.

The main contribution of this work can be summarized as follows:

— A novel spatio-temporal graph learning network for traffic data forecasting is proposed. It integrates multiple graph learning modules, including static, adaptive, and dynamic spatial convolutional graphs, to capture the static and evolving spatial correlations.
— A sequence-to-sequence fusion module and a spatio-temporal graph transformer module are designed for jointly learning the short- and long-term temporal correlations.
— Several graph learning techniques, such as unidirectional graph convolution and graph regularization, are developed to help the model capture more complex hidden patterns and improve the graph quality.
— The proposed model is evaluated on three large-scale traffic datasets and compared with several baseline methods. The experimental results demonstrate the excellent performance of our model compared to other state-of-the-art techniques.

The rest of the article is organized as follows. Related work is given in Section 2. Section 3 discusses the proposed AJSTGL and its main components. Section 4 presents the experimental results and compares our model with other baseline methods. Finally, Section 5 summarizes the article and suggests potential future work.

2 RELATED WORK

Spatio-temporal data processing has attracted lots of attention due to its broad applications in many real-world problem domains [18, 19, 25]. CNN has been used to capture the spatial dependencies on grid-structured data. Recurrent models such as RNN and **Long Short-Term Memory (LSTM)** are widely adopted to model temporal relations. Yuan et al. [35] applied CNN and LSTM to create an auxiliary model that learns the latent space embedding of historical traffic flow. Environmental data and traffic flow records are combined to predict travel time in the road network. However, each of these methods has its limitations when handling spatio-temporal data. For instance, CNNs are restricted in processing grid-structured data since they can only be applied to data measured in Euclidean distance [13]. The performance of the recurrent-based model degrades when making long-term predictions since it does not have enough "memory" for long input sequences [36].

GCN has been successfully applied in spatio-temporal data modeling where the correlation between objects can be constructed as a graph. Existing applications of GCN-based networks (e.g., [9, 14, 23]) only model the stationary spatial dependencies in the GCN layers. Although there have been efforts in incorporating temporal node dependency into the graph convolutional layers, like embedding the signals of immediate adjacent time steps in the same GCN layer [40], such approaches still fall short in learning the long-term temporal relations. Recent advances in transformer-based models [5, 10, 27] enable the network to learn temporal information from a more extended period. Instead of memorizing the hidden states of neighboring tokens, the transformer utilizes the self-attention mechanism to generate the contextual vectors based on the entire input sequence.

Traffic forecasting is one domain that heavily involves spatio-temporal data. DCRNN [20] makes multi-step traffic prediction using an encoder-decoder network structure that integrates graph convolution with diffusion operation and RNNs. However, it only uses the traditional Euclidean distance measurement to construct the graph adjacency matrix, which ignores many complex node relations in a real traffic network. STGCN [34] applies GCN to model spatial correlations and temporal convolution networks to learn the temporal correlations from the input data. However, it does not take into account the dynamic aspect of the spatio-temporal dependencies. Some works utilize a data-driven method to adaptively construct the graph adjacency matrix. AGCRN [2] learns the adjacency matrix from input signals to model the spatial relations and adopts a gated RNN to model the temporal relations. Zheng et al. [41] apply dilated causal spatio-temporal graph convolution layers to capture the spatio-temporal dependencies in multiple time intervals and develop multi-range attention to help the model focus on different time ranges. In AdapGL [38], the model trains two GCNs back to back. The pre-defined adjacency matrix is used to optimize the learned adjacency matrix through each training iteration. However, the RNN-based layers these models adopt limit their ability to capture long-term temporal correlations. Attention mechanism has also been leveraged to formulate the dynamic spatio-temporal correlation in traffic forecasting problems. Li et al. [21] use static and dynamic graphs to learn short and long-term data patterns. A multi-head attention unit is leveraged to capture the correlations among multi-variables. Lan et al. [17] apply a dynamic attention map that extracts the probability distribution among nodes to derive the adjacency matrix. However, these methods overlook the node dependency in the pre-defined adjacency matrix and require specific manipulation of the network topology.

To address the limitations in the aforementioned works, our proposed model utilizes both pre-defined and dynamically generated graph adjacency matrices. It enables our model to learn a more comprehensive representation of the spatial dependencies. Moreover, we incorporate recurrent- and transformer-based layers to facilitate the model capturing short- and long-term

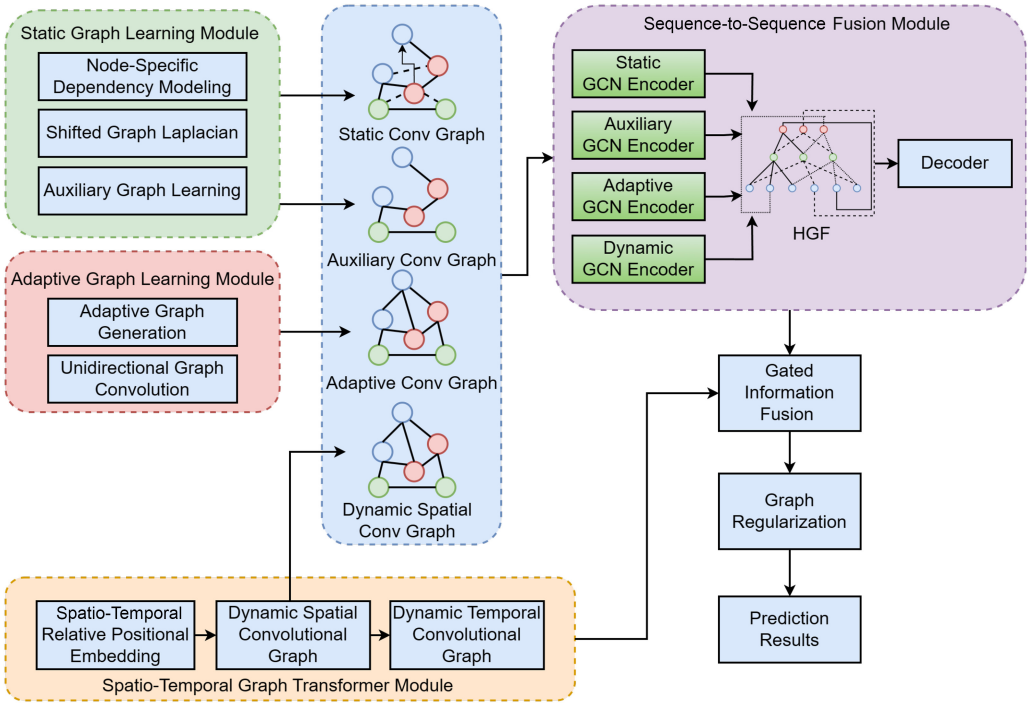


Fig. 1. Overall framework of the proposed model.

temporal dependency simultaneously. Additionally, the proposed graph learning approaches, such as the shifted graph Laplacian, node-specific dependency modeling, and the auxiliary graph, greatly expand the model's capability in extracting more complex and dynamic traffic patterns.

3 METHODOLOGY

In this section, we introduce the proposed AJSTGL and its main components. Figure 1 illustrates the overall framework of AJSTGL, which mainly consists of the static graph learning module, adaptive graph learning module, spatio-temporal graph transformer module, and sequence-to-sequence fusion module. We apply several techniques, such as unidirectional graph convolution, gated information fusion, and graph regularization, to enhance the model's ability to model spatio-temporal dependencies and produce higher-quality graphs.

3.1 Preliminaries

Given a graph $G = (V, E, X, A)$, where $|V| = N$ represents the set of nodes, E represents the edges, $X \in \mathbb{R}^{N \times Q}$ is the node feature with Q as the vector size, and $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix that defines the topology of the graph network. The normalized graph Laplacian with self-loop can be represented as follows:

$$L = I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}, \tag{1}$$

where $I_N \in \mathbb{R}^{N \times N}$ is the identity matrix and D represents the degree matrix of A. To reduce the computation complexity on a large graph, the first-order Chebyshev polynomial approximation [16] of the graph convolution can be described as follows:

$$X^{l+1} = \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) X^l \Theta + b, \tag{2}$$

where $X^l \in \mathbb{R}^{N\times Q}$ and $X^{l+1} \in \mathbb{R}^{N\times Z}$ are the input and output at layers l and $l+1$ with Q and Z as the corresponding vector sizes, and $\Theta \in \mathbb{R}^{Q\times Z}$ and $b \in \mathbb{R}^Z$ are the weight and bias terms, respectively.

3.2 Static Graph Learning Module

The static graph learning module combines node connectivity and geographical distance to model the spatial node dependencies.

3.2.1 Node-Specific Dependency Modeling. In a standard graph convolution layer, the weights are shared among all nodes when the convolution operation is applied. According to Equation (2), for a specific node $x_i \in \mathbb{R}^{1\times Q}$, the trainable weight parameter $\Theta \in \mathbb{R}^{Q\times Z}$ in a convolution operation transforms x_i into $x_i^{'} \in \mathbb{R}^{1\times Z}$. The shared weight Θ effectively learns the spatial relations among all nodes, which greatly reduces the number of parameters and saves computation time. However, in many real-world scenarios, such a weight-sharing strategy could become a constraint since heterogeneous spatial patterns could exist among closely associated nodes. For instance, airports with connected flight routes or are geographically closely located may exhibit diverse traffic patterns due to the influence of weather conditions and special events. Therefore, sharing weight parameters among all nodes limits the model's ability for accurate traffic data forecasting, which relies on learning the distinct spatial patterns. It is essential to incorporate additional parameter space to model node-specific properties for each node.

To address the aforementioned challenge, we conduct node-specific dependency modeling by adding an extra dimension to the weight parameter Θ, which results in a node-specific weight parameter $\hat{\Theta} \in \mathbb{R}^{N\times Q\times Z}$, where N indicates the node counts in the graph. However, due to the additional dimension in the new weight matrix, the number of parameters that need to be optimized expands exponentially. It significantly increases the computational cost and potential for overfitting, especially for graphs with large numbers of nodes. To solve this problem, we apply graph parameter decomposition to factorize $\hat{\Theta}$ into δ_s and W_s, where $\delta_s \in \mathbb{R}^{N\times d_s}$ with $d_s << N$ and $W_s \in \mathbb{R}^{d_s\times Q\times Z}$. As a result, instead of optimizing a huge three-dimensional matrix $\hat{\Theta}$, the model learns two smaller weight matrices δ_s and W_s. The updated graph convolution layer can be expressed as

$$X^{l+1} = \left(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)X^l\delta_s W_s + \delta_s b_s, \tag{3}$$

where $b_s \in \mathbb{R}^{d_s\times Z}$ is the updated bias term.

3.2.2 Shifted Graph Laplacian. The intuitive node relations are commonly used to model the spatial node dependency in a pre-defined network topology. Nevertheless, there exist hidden patterns that cannot be captured by the pre-defined graph schema. Thus, we develop a strategy to introduce a shifted graph Laplacian L_s on top of the intrinsic graph Laplacian L to learn the hidden spatial node dependencies.

The adjacency matrix in a shifted graph Laplacian uses the node-wise Gaussian smoothed similarity score as the values. In this work, the similarity score is measured by the cosine similarity between each pair of nodes. The shifted graph A_s adjacency matrix is represented as follows:

$$A_s^{ij} = \begin{cases} exp(-\frac{cos(x_i,x_j)^2}{2\sigma^2}), & i \neq j \ and \ cos(x_i,x_j) \geqslant \varepsilon_s \\ 0, & otherwise, \end{cases} \tag{4}$$

where $cos(x_i, x_j)$ is the cosine similarity score between nodes x_i and x_j, σ is the standard deviation, and ε_s is the threshold parameter that controls the matrix sparsity. We empirically set ε_s to 0.5

to avoid the matrix from becoming overly sparse. Then, the shifted graph Laplacian L_s can be expressed as follows:

$$L_s = I_N + D_s^{-\frac{1}{2}} A_s D_s^{-\frac{1}{2}}, \tag{5}$$

where D_s is the degree matrix of A_s. Finally, the graph Laplacian for the static convolutional graph can be calculated as

$$\widetilde{L} = L + \alpha L_s, \tag{6}$$

where α is a learnable parameter that controls the shifting scale on the original graph Laplacian. Based on Equation (6), Equation (3) can be transformed to

$$X^{l+1} = \widetilde{L} X^l \delta_s W_s + \delta_s b_s. \tag{7}$$

3.2.3 Auxiliary Convolutional Graph. An auxiliary convolutional graph is used to exploit contextual information such as wind, humidity, and temperature in a traffic network. The adjacency matrix of the auxiliary convolutional graph is generated based on the node's geographic proximity. To ensure the matrix symmetrical property, a Gaussian kernel function is applied to produce the edge weights. Similar to Equation (4), the adjacency matrix A_a can be expressed as follows:

$$A_a^{ij} = \begin{cases} exp(-\frac{dist(x_i, x_j)^2}{2\sigma^2}), & i \neq j \ and \ dist(x_i, x_j) \leqslant \varepsilon_c \\ 0, & otherwise, \end{cases} \tag{8}$$

where $dist(x_i, x_j)$ is the spherical distance between nodes x_i and x_j, σ is the standard deviation, and ε_c is the threshold parameter that controls the matrix sparsity. We empirically set ε_c to 0.4 for optimal model performance.

3.3 Adaptive Graph Learning Module

3.3.1 Adaptive Graph Generation. Many existing studies only use pre-defined adjacency matrices to represent the node relations. This limits the model's ability to learn more complex spatio-temporal graph patterns. Another disadvantage of the conventional graph convolutional layer is that it assumes bidirectional node correlations. However, in many real-world problems, such as traffic data forecasting, the changes in traffic patterns may only transmit in a single direction.

To address the aforementioned limitations, we propose an adaptive graph generation approach. It automatically learns the hidden graph topology from the input data. Furthermore, the learned graph adjacency matrix considers the unidirectional correlations between each pair of nodes. First, we use an anti-symmetric matrix to construct the normalized graph Laplacian of the new graph L_a:

$$L_a = I_N + softmax\left(ReLU\left(M_p M_q^T - M_q M_p^T\right)\right), \tag{9}$$

where $M_p, M_q \in \mathbb{R}^{N \times d_a}$, $d_a << N$ are the learned node embedding that formulates the adjacency matrix through the training process. Being the product of the two anti-symmetric matrices, $M_p M_q^T - M_q M_p^T$ has zero values in all of its diagonal elements. The $ReLU()$ function further transforms another half of the negative matrix elements to zeroes. The $softmax()$ function is used to normalize the adjacency matrix. Additionally, we apply node-specific dependency modeling from Equation (3) to enable the model to learn node-specific dependencies. Finally, the GCN layer in the adaptive convolutional graph can be expressed as follows:

$$X^{l+1} = L_a X^l \delta_a W_a + \delta_a b_a. \tag{10}$$

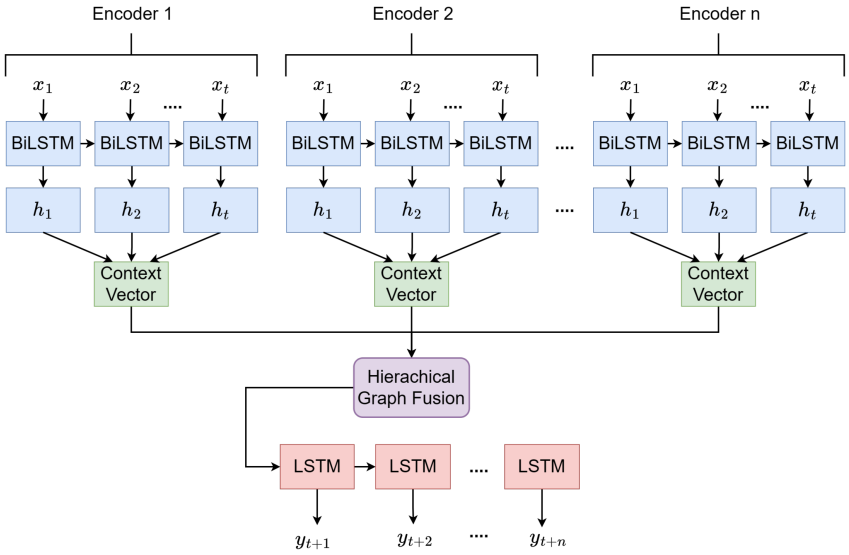


Fig. 2. Illustration of the overall structure of S2SFM.

3.3.2 Unidirectional Graph Convolution Layer. A unique graph convolution transformation approach is developed to further enhance the model's capability of learning the heterogeneous unidirectional data patterns. The proposed unidirectional graph convolution layer can be represented as follows:

$$X^{l+1} = \left(\hat{L}X^l\hat{\delta}\hat{W} + \hat{\delta}\hat{b}\right) \oplus \left(\hat{L}^T X^l\hat{\delta}\hat{W} + \hat{\delta}\hat{b}\right). \tag{11}$$

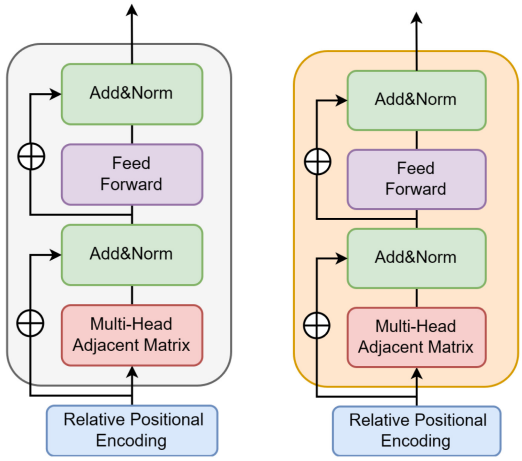
The transformed graph convolutional layer adds a second term $\hat{L}^T X^l\hat{\delta}\hat{W} + \hat{\delta}\hat{b}$ with transposed graph Laplacian $\hat{L}^T$ to model the reversed dataflow pattern.

3.4 Sequence-to-Sequence Fusion Module

We develop S2SFM, a sequence-to-sequence fusion module, to capture the short-term temporal node dependencies. Figure 2 demonstrates the overall structure of the proposed S2SFM. Unlike typical sequence-to-sequence models with a single encoder, S2SFM utilizes parallelized BiLSTM-based encoders to concurrently process the graph signals from all GCNs. The context vectors of all encoders are combined using the **Hierarchical Graph Fusion (HGF)** approach [31] to capture the n-modal cross-modality interactions among all graph signals. HGF utilizes a tree-based graph structure to join the input signals on different levels. It learns the unique joint-modality representations, with lower-level nodes representing basic interactions and higher-level nodes modeling more complex correlations. The final output of HGF can be represented as follows:

$$F_{combined} = F_{uni-modal} \oplus F_{bi-modal} \oplus F_{tri-modal} \cdots \oplus F_{n-modal}, \tag{12}$$

where $F_{combined}$ is the final combined context vector, $F_{uni-modal}$, $F_{bi-modal}$, $F_{tri-modal}$, and $F_{n-modal}$ are the representation vectors for each level, and $\oplus$ is the concatenation operation. In this study, we use HGF to combine the output signals of four GCNs. Therefore, the first level contains the uni-modal interactions, the second level learns the bi-modal interactions, the third level models tri-modal interactions, and the fourth level captures the quad-modal interactions. The combined context vector is then passed into the decoder to produce the output sequence.



(a) Dynamic Spatial Convolutional Graph (b) Dynamic Temporal Convolutional Graph

Fig. 3. Illustration of the architecture of STGTM, the proposed spatio-temporal graph transformer module. The relative positional encoding learns the node's spatial and temporal dependency and generates the spatio-temporal aware embedding vectors. The dynamic spatial and temporal convolutional graphs are stacked together to model the node relations jointly.

3.5 Spatio-Temporal Graph Transformer Module

To complement S2SFM in long-term prediction, we develop STGTM (the spatio-temporal graph transformer module) to jointly model the dynamic spatio-temporal dependencies. Its overall structure is demonstrated in Figure 3. STGTM consists of spatio-temporal relative positional encoding, which generates the time-evolving spatial embedding of the graph signals, and the dynamic spatial and temporal convolutional graphs that model the spatial and temporal node dependencies, respectively.

3.5.1 Spatio-Temporal Relative Positional Encoding. The transformer-based model applies positional encoding to embed the spatial relations of each token in the input sequence. Static positional encoding approaches, such as the widely adopted sinusoidal wavelength method [29], rely on fixed-length input sequences and do not consider their relative positional relations. In STGTM, we apply a relative positional encoding layer with trainable parameters to learn the dynamic spatial and temporal dependencies. More specifically, matrices $PE_S \in \mathbb{R}^{N\times N}$ and $PE_T \in \mathbb{R}^{C\times C}$ in the encoding layer hold the learned spatial and temporal relative positional embedding, where C represents the number of timesteps in the input sequence. The embedded input feature vector can be represented as

$$X_E = conv\left(\hat{X} \oplus PE_S^{'} \oplus PE_T^{'}\right), \tag{13}$$

where $\hat{X} \in \mathbb{R}^{C\times N\times Q}$ is a three-dimensional vector that contains the features of all nodes across the entire historical time sequence, and $PE_S^{'} \in \mathbb{R}^{C\times N\times N}$ and $PE_T^{'} \in \mathbb{R}^{C\times N\times C}$ are the expanded matrices of PE_S and PE_T along the spatial and temporal dimensions, respectively. In addition, $conv()$ is a 1×1 convolutional layer that converts the original input vector $\hat{X}$ into its spatio-temporal embedded form $X_E \in \mathbb{R}^{C\times N\times Q}$.

3.5.2 Dynamic Spatial Convolutional Graph. The DSCG (dynamic spatial convolutional graph) learns the adjacency matrix from the time-evolving hidden patterns in the positional embedded

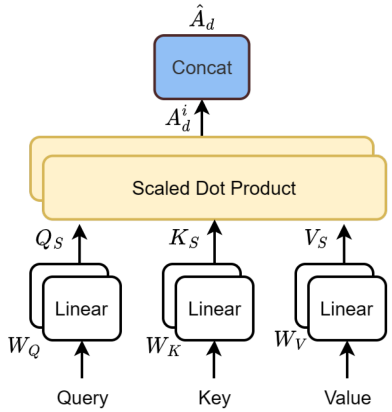


Fig. 4. The multi-head adjacency matrix structure.

input sequence. We apply a multi-head self-attention mechanism to capture the pattern representations from different subspaces and form the adjacency matrix. Figure 4 demonstrates the structure of the multi-head adjacency matrix that applies the self-attention mechanism. The subspaces used in DSCG contains the query $Q_S \in \mathbb{R}^{N\times d_k}$, key $K_S \in \mathbb{R}^{N\times d_k}$, and value $V_S \in \mathbb{R}^{N\times d_v}$ spaces, where d_k is the query size and key vector size, and d_v is the value vector size. Each matrix is calculated as the product of the input feature vector and its corresponding weight parameter:

$$Q_S = X_E^{'} W_Q,\ K_S = X_E^{'} W_K,\ V_S = X_E^{'} W_V, \tag{14}$$

where $X_E^{'} \in \mathbb{R}^{N\times Q}$ represents a single timestep in X_E, and $W_Q \in \mathbb{R}^{Q\times d_k}$, $W_K \in \mathbb{R}^{Q\times d_k}$, and $W_V \in \mathbb{R}^{Q\times Q}$ are the learnable weight parameters for Q_S, K_S, and V_S, respectively.

The adjacency matrix of DSCG for each attention head unit can be derived from the scaled dot product among the query, key, and value matrices:

$$A_d^i = softmax\left(\frac{Q_S^i K_S^{iT}}{\sqrt{d_k}}\right) V_S^i, \tag{15}$$

where $softmax()$ is used to obtain the normalized spatio-temporal node-wise dependency in the ith attention head A_d^i, and $\sqrt{d_k}$ serves as the scaling factor to stabilize the gradients during the training. The multi-head attention unit is generated by concatenating each attention head unit:

$$\hat{A_d} = concat\left(A_d^1, A_d^2, \ldots, A_d^i\right) W_O, \tag{16}$$

where $A_d^1, \ldots, A_d^i$ are the single-head attention units, and W_O is the learnable weight parameter. The output of the node input feature through the multi-head attention unit is then calculated as follows:

$$\hat{X_E} = \hat{A_d} X_E. \tag{17}$$

We add residual connection and layer normalization to help improve the model stability and generalization performance:

$$\hat{X_E^{'}} = LN\left(X_E + \hat{X_E}\right), \tag{18}$$

where $LN()$ represents layer normalization [1]. Then, the output is fed into two feed-forward layers with ReLU activation function:

$$X_{ST} = ReLU\left(ReLU\left(\hat{X_E^{'}} W_a + b_a\right) W_b + b_b\right), \tag{19}$$

where W_a, W_b, b_a, and b_b are the weight parameters. In the last step, we apply residual connection and layer normalization again on the output of the two feed-forward layers to generate the final feature vector $\hat{X_{ST}} \in \mathbb{R}^{C\times N\times Q}$ of STGTM:

$$\hat{X_{ST}} = LN\left(X_{ST} + \hat{X_E^{'}}\right). \tag{20}$$

3.5.3 Dynamic Temporal Convolutional Graph. The DTCG (dynamic temporal convolutional graph) shares a network structure similar to that of DSCG. We first apply residual connection between the original input vector X and DSCG output $\hat{X_{ST}}$ to get $\tilde{X} \in \mathbb{R}^{C\times N\times Q}$. The relative positional encoding layer is applied to produce embedding results using the temporal encoding PE_T. $\tilde{X}$ and PE_T are concatenated and passed through a 1×1 convolutional layer to generate the temporal encoded vector $\tilde{X}_E$. Similar to DSCG, the multi-head self-attention mechanism is applied to capture the pattern representations from $\tilde{Q}_S$, $\tilde{K}_S$, and $\tilde{V}_S$ subspaces:

$$\tilde{Q_S} = \tilde{X}_E^{'}\tilde{W_Q},\ \tilde{K_S} = \tilde{X}_E^{'}\tilde{W_K},\ \tilde{V_S} = \tilde{X}_E^{'}\tilde{W_V}, \tag{21}$$

where $\tilde{X}_E^{'} \in \mathbb{R}^{C\times Q}$ represents an arbitrary node in $\tilde{X}_E$, and $\tilde{W_Q} \in \mathbb{R}^{Q\times \tilde{d_k}}$, $\tilde{W_K} \in \mathbb{R}^{Q\times \tilde{d_k}}$, and $\tilde{W_V} \in \mathbb{R}^{Q\times Q}$ are the trainable weight parameters for $\tilde{Q}_S$, $\tilde{K}_S$, and $\tilde{V}_S$. Then, the attention-aware adjacency matrix based on a single attention unit can be expressed as

$$\tilde{A}_d^i = softmax\left(\frac{\tilde{Q_S^i}\tilde{K_S^{iT}}}{\sqrt{d_k}}\right)\tilde{V_S^i}, \tag{22}$$

Similar to DSCG, the final multi-head attention context vector is generated by concatenating each single-head unit:

$$\tilde{A}_d = concat\left(\tilde{A}_d^1, \tilde{A}_d^2, \ldots, \tilde{A}_d^i\right)\tilde{W}_O, \tag{23}$$

Next, the intermediate output $\tilde{X}_E$ is multiplied with the multi-head unit to produce the attention-weighted vector:

$$\bar{X}_E = \tilde{A}_d\tilde{X}_E, \tag{24}$$

The attention-weighted vector is fed into two feed-forward layers with residual connection and layer normalization:

$$\bar{X}_E^{'} = LN\left(\tilde{X}_E + \bar{X}_E\right), \tag{25}$$

$$\bar{X}_{ST} = ReLU\left(ReLU(\bar{X}_E^{'}\bar{W}_a + \bar{b}_a)\bar{W}_b + \bar{b}_b\right), \tag{26}$$

$$\tilde{X}_{ST} = LN\left(\bar{X}_{ST} + \bar{X_E^{'}}\right), \tag{27}$$

where $\tilde{X}_{ST} \in \mathbb{R}^{C\times N\times Q}$ is the final output of STGTM, which will be combined with the output of S2SFM using gated information fusion.

3.5.4 Gated Information Fusion. We apply a gating mechanism to adaptively combine the output features of S2SFM and STGTM. More specifically, a set of learnable parameters are used as a gate to control the relative weighting between the two input vectors:

$$\mathbb{G} = sigmoid\left(\left(\gamma\left(\tilde{X}_{ST}\right) \oplus \gamma\left(X_{S2SFM}\right)\right)W_g + b_g\right), \tag{28}$$

where the sigmoid activation function is used to limit the value of gate $\mathbb{G}$ within range $[0, 1]$, $\gamma()$ is a linear projection function that transforms the feature vectors from both modules into a one-dimensional vector, and W_g and b_g are the weight parameters. As a result, we can use $\mathbb{G}$ to fuse the two feature vectors as

$$X_f = \mathbb{G}\odot\tilde{X}_{ST} + (1-\mathbb{G})\odot X_{S2SFM}, \tag{29}$$

where $\odot$ is element-wise multiplication. $X_f \in \mathbb{R}^{C\times N\times Q}$ represents the joint spatio-temporal dependencies that will be used for the final prediction.

3.6 Prediction Layer and Loss Function

To generate the multi-step prediction, the output of the gated information fusion module X_f is passed into a double-layered convolutional network:

$$Y = conv(conv(X_f)). \tag{30}$$

X_f is transformed into a two-dimensional vector $Y \in \mathbb{R}^{N\times\tau}$, where τ is the number of timesteps in the output sequence.

To improve the quality of the convolutional graphs, we optimize the learned graph adjacency matrix by applying certain constraints in the loss function. The model is forced to minimize a graph regularization term during the training process. The graph regularization term is expressed as follows:

$$J_{gr} = \sum_{i=1}^{N}\sum_{j=1}^{N} \left\|x_i - x_j\right\|_2^2 A_a^{ij} + \beta\left(A_a^{ij}\right)^2, \tag{31}$$

where A_a^{ij} is the corresponding adjacency matrix element for nodes x_i and x_j, and β is a scaling factor controlling the matrix sparsity. Term $\left\|x_i - x_j\right\|_2^2$ enforces the graph proximity property by encouraging larger A_a^{ij} values when x_i and x_j are close and smaller A_a^{ij} values when the two nodes move away in the latent space.

The loss function minimizes the **Mean Absolute Error (MAE)** between the ground truth value and the predicted value. The graph regularization term in Equation (31) is added to the MAE loss function. The final loss function is expressed as follows:

$$L_{loss} = \left\|Y - \hat{Y}\right\|_1 + \lambda J_{gr}, \tag{32}$$

where $\hat{Y}$ is the ground truth value and λ is a scaling factor that controls the degree of regularization. We empirically set λ to 0.4 to achieve the best model performance.

4 EXPERIMENTS AND ANALYSES

4.1 Datasets

We evaluate AJSTGL on several large-scale real-world datasets: **Reporting Carrier On-Time Performance (RCOTP)**, PeMSD4, and PeMSD8:

- *RCOTP*: Released by the U.S. Bureau of Transportation Statistics (BTS), this dataset contains flight operation information for all reporting airlines in the U.S. domestic market. Records from January 2017 to December 2021 were collected, which include 433 airports and 30,940,455 records. Among them, 8,102 origin and destination (OD) pairs are retrieved. Figure 5 visualizes major airports based on the volumes of their connection flights. In the experiment, we predict the multi-step flight arrival delays at the airport level.
- *PeMSD4*: This dataset includes historical traffic condition data in the San Francisco Bay area published by the Caltrans Group using the Performance Measurements System (PeMS). The data was collected in 5-minute intervals using 307 sensors on seven major roads. The period covered by PeMSD4 ranges from January 2018 to February 2018. Three types of measurements are used, which include average speed, average occupancy, and traffic flow. In this study, we focus on traffic flow and traffic speed predictions.
- *PeMSD8*: Also published by Caltrans Group, this dataset contains the traffic information in the San Bernardino area from July 2016 to August 2016. The 170 sensors used 5-minute

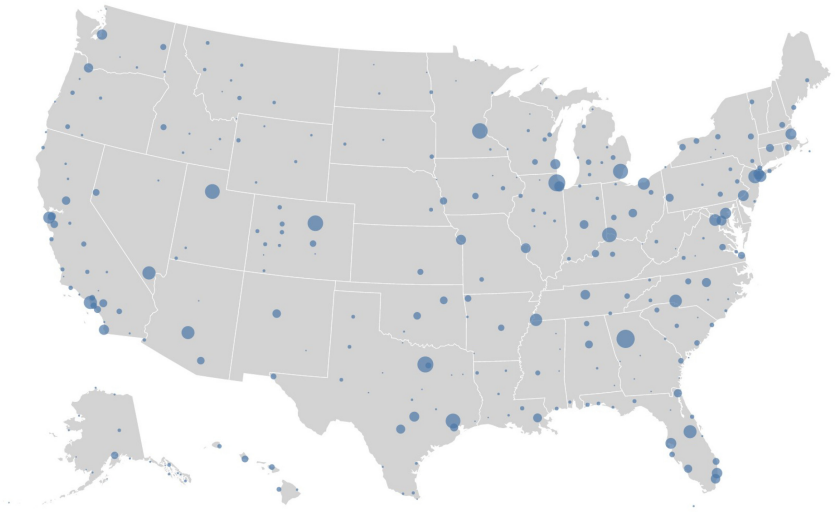


Fig. 5. Visualization of the RCOTP dataset showing major U.S. airports based on the number of connection flights. The blue dot size indicates the relative connection flight volume an airport receives compared to other airports.

intervals on eight roads to collect the average speed, average occupancy, and traffic flow information. Like PeMSD4, we focus on traffic flow and speed predictions in this study.

We also utilize real-time weather forecasting records from the National Climate Data Center (NCDC). Weather conditions collected by nearby weather stations at each traffic network node in the same period are used as the input for the auxiliary convolutional graph network.

4.2 Experimental Setup

The RCOTP data is aggregated at the airport level to produce the average hourly flight delay. For PeMSD4 and PeMSD8, we aggregate the traffic speed at 5-minute intervals. Data preprocessing is done on all datasets: (1) missing values are interpolated as the mean value of the previous and later timesteps, (2) categorical and discrete values are one-hot encoded, and (3) continuous values are normalized using min-max normalization. For RCOTP, we conduct arrival delay prediction in the next 10-hour horizons. For PeMSD4 and PeMSD8, we perform traffic flow prediction and traffic speed forecast in the next 15-, 30-, 45-, and 60-minute horizons.

All datasets are split into 60% for training, 20% for validation, and 20% for testing. Hyperparameters including the threshold parameter ε_s in the shifted graph Laplacian adjacency matrix, ε_c in the auxiliary GCN adjacency matrix, node embedding size d_s, β in the graph regularization, and λ in the final loss function are tuned on the validation set. The numbers of layers in the static, adaptive, and auxiliary convolutional graphs are set to 3, 4, and 3, respectively. In the dynamic spatial and temporal convolutional graph, the hidden dimension of self-attention, the dimension of all sub-layers, and the number of attention heads are set to 64, 512, and 8, respectively.

The model is trained using the Adam optimizer [15] with a training rate of 0.001 and batch size of 128. Early stopping is applied to prevent the model from overfitting, and the model that performed best on the validation set is saved. All experiments are repeated five times, and the dataset is shuffled before the split to ensure randomness in the training, validation, and testing datasets.

Table 1. Overall Performance Comparison of AJSTGL and Baselines on Three Datasets: RCOTP (Average 10-Hour Arrival Delay), PeMSD4 (Traffic Flow), and PeMSD8 (Traffic Flow)

Model	Dataset	RCOTP		PeMSD4		PeMSD8	
	Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE
ARIMA		13.36	24.04	40.05	66.08	36.11	59.85
DCRNN		9.93	16.39	21.25	33.49	16.81	26.34
STGCN		9.86	16.86	21.2	35.01	17.52	27.14
ASTGCN		9.66	16.22	22.94	35.41	18.28	28.44
Graph WaveNet		10.52	17.88	20.01	31.11	15.61	24.44
AGCRN		9.02	15.24	19.85	32.28	15.97	25.55
STGMN		9.36	16.15	20.83	32.98	16.04	24.18
iDCGCN		9.10	15.57	19.98	32.11	15.56	25.02
DSTAGNN		9.08	15.33	19.76	31.97	15.67	25.37
AJSTGL		**7.60**	**12.84**	**18.07**	**29.37**	**13.79**	**22.48**

4.3 Evaluation Metrics and Baseline Methods

To evaluate the proposed AJSTGL, we adopt MAE and **Root Mean-Squared Error (RMSE)**. Several baselines are utilized to compare with our proposed method:

— *ARIMA (Autoregressive Integrated Moving Average)* [28]: A statistical analysis model that uses previous timestep values to predict future values.
— *DCRNN* [20]: A deep neural network with an encoder-decoder structure that combines graph convolution with diffusion operation and RNNs for multi-step prediction.
— *STGCN* [34]: A network utilizing GCN to model spatial correlations and a temporal convolution network to capture temporal dependencies.
— *ASTGCN* [8]: A spatio-temporal convolutional graph network leveraging an attention mechanism to model spatial and temporal dependencies.
— *Graph WaveNet* [32]: A spatio-temporal convolutional graph network that applies diffusion convolution to capture spatial dependencies and leverages dilated convolution to model the temporal correlations.
— *AGCRN* [2]: A convolutional graph network learning the correlation matrix from data to capture the spatial dependencies and utilizing an RNN to learn the temporal correlation from the input data.
— *STGMN* [24]: A spatial-temporal graph convolutional model utilizing gated multi-graph convolution to model the spatial dependency and leveraging the multi-scale receptive field of the CNN to capture the temporal dependency.
— *iDCGCN* [22]: A GCN using dynamic Chebyshev polynomial mechanism to capture long-term temporal dependency by extracting the features from various time periods.
— *DSTAGNN* [17]: A graph neural network that extracts the probability distribution among nodes from the historical data to derive the node dependencies that replace the pre-defined static graph.

4.4 Experimental Results

4.4.1 Overall Comparison. Table 1 illustrates the model performance of AJSTGL on the three datasets against all nine baselines. As shown in the table, we report the average arrival delay for RCOTP and traffic flow prediction for PeMSD4 and PeMSD8. The traffic speed prediction results on PeMSD4 and PeMSD8 are presented later in Tables 3 and 4, respectively.

Table 2. Overall Performance Comparison of AJSTGL and Baselines on Average Hourly Flight Arrival Delay Prediction in 10-Hour Horizons Using the RCOTP Dataset

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
ARIMA	7.33	8.02	9.26	11.83	13.10	15.77	15.93	16.28	17.73	18.32
DCRNN	6.38	6.78	7.36	8.10	9.39	11.02	11.86	12.17	12.42	12.94
STGCN	6.24	6.83	7.88	8.07	9.14	11.73	11.89	12.19	12.60	12.02
ASTGCN	5.98	5.54	6.85	8.15	9.28	10.85	11.99	12.27	13.65	12.03
GW	5.19	5.68	6.46	8.38	10.98	12.07	13.09	13.43	14.83	15.07
AGCRN	5.15	5.63	5.50	7.31	9.58	10.31	11.24	11.42	11.84	12.19
STGMN	5.44	5.75	5.97	7.63	9.87	10.6	11.55	11.97	12.33	12.51
iDCGCN	5.2	5.58	5.71	7.49	9.64	10.43	11.36	11.55	11.96	12.23
DSTAGNN	5.09	5.42	5.57	7.02	9.5	10.39	11.63	11.72	12.01	12.28
AJSTGL	**4.02**	**4.26**	**5.21**	**6.99**	**8.44**	**9.32**	**9.59**	**9.73**	**9.95**	**10.21**

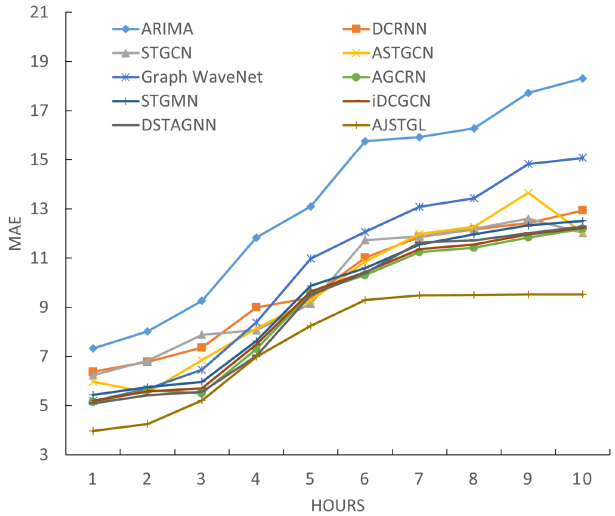


Fig. 6. Visualization of MAE for the RCOTP dataset obtained by AJSTGL and other baselines—arrival delay in the next 10 horizons.

RCOTP. It can be seen from Table 1 that our model achieves the best performance in terms of all evaluation metrics. ARIMA, the traditional time series approach, performs the worst among all methods, as it only captures the temporal dependency in the data. Other spatio-temporal graph models demonstrate more robust performance since they model both the spatial and temporal correlations from the data. Our proposed AJSTGL outperforms all baseline methods and leads the second-best performer (AGCRN) by 15.74% in MAE. This implies that AJSTGL is very effective at long-term forecasting and that including auxiliary GCN facilitates the model to capture additional spatial node dependencies.

Table 2 and Figure 6 contain a more in-depth view of the changes in prediction results through all horizons. Overall, the prediction accuracy drops across all models as the prediction interval increases. As shown in the figure, ARIMA's performance decreases dramatically in long-term intervals compared to short-term ones. In comparison, the rest of the GCN-based models achieve better results in long-term intervals, as they consider both temporal and spatial correlations. For

Table 3. MAE and RMSE of Traffic Speed Prediction on PeMSD4 for the Next Four Horizons

Model	Dataset	PeMSD4 (15/30/45/60 min)	
	Metrics	MAE	RMSE
ARIMA		2.8	5.43
DCRNN		1.34/1.79/2.06/2.27	2.95/4.08/4.82/5.36
STGCN		1.48/1.95/2.23/2.61	3.01/4.22/5.03/5.66
ASTGCN		2.11/2.45/2.62/2.74	3.94/4.58/4.94/5.18
Graph WaveNet		1.45/1.92/2.15/2.55	2.97/4.15/4.89/5.52
AGCRN		2.04/2.39/2.58/2.66	3.86/4.47/4.80/5.12
STGMN		1.87/2.05/2.31/2.54	3.52/4.40/4.66/5.08
iDCGCN		1.63/1.78/1.98/2.25	3.32/4.02/4.35/4.85
DSTAGNN		1.57/1.72/1.90/2.16	3.08/3.94/4.05/4.67
AJSTGL		**1.26/1.62/1.78/1.91**	**2.65/3.40/3.74/3.82**

instance, Graph WaveNet outperforms ARIMA since it applies diffusion and dilated convolution to capture the spatial and temporal correlations. However, Graph WaveNet's prediction accuracy deteriorates too quickly compared to other GCN-based models in long-term intervals. DCRNN, STGCN, ASTGCN, AGCRN, STGMN, iDCGCN, and DSTAGNN have similar performance, with some models demonstrating an advantage in short-term intervals, such as DSTAGNN, and other models like STGCN and ASTGCN maintaining a more stable prediction accuracy.

Nevertheless, our proposed AJSTGL yields higher prediction accuracy values across all horizons, especially in mid- to long-term intervals. As discussed earlier, long-term prediction suffers from diminishing return effects while utilizing historical observations. Therefore, it is essential to leverage contextual information and construct the graph adjacency matrix based on dynamic long-term spatio-temporal dependencies. In addition, as a spatio-temporal model, AJSTGL incorporates the functional structure to extract temporal dependency information from the data. However, compared with other spatio-temporal baselines, AJSTGL leverages recurrent and transformer-based network structures simultaneously. Combined with HGF and gated information fusion, our model demonstrates a more stable performance through short- and long-term time intervals.

PeMSD4 and PeMSD8. According to Table 1, similar to the outcome from the RCOTP dataset, the traditional time series model ARIMA produces the lowest overall prediction accuracy and shows its weakness in long-term interval prediction. Other GCN-based models achieve better results by simultaneously learning spatial and temporal correlations. There are some subtle changes in the performance ranking. DSTAGNN produces the second-lowest prediction error on the PeMSD4 dataset, and iDCGCN has the second-best prediction accuracy on the PeMSD8 dataset. Once again, AJSTGL achieves the lowest MAE and RMSE scores in both datasets and leads the second-best baseline model DSTAGNN by 8.55% in MAE on PeMSD4 and iDCGCN by 11.38% in MAE on PeMSD8.

Tables 3 and 4 show the traffic speed prediction results in four consecutive intervals (15, 30, 45, and 60 minutes). As an autoregressive model, the result of ARIMA is only shown on the last time interval. In comparison, all GCN-based models outperform ARIMA, given that they learn the spatio-temporal correlations simultaneously. On average, DSTAGNN outperforms other baseline models for traffic speed prediction by a small margin. This may be due to the availability and consistency of the historical data pattern in the two PeMSD datasets. DSTAGNN's focus on constructing the probability distribution of historical data helps enhance the quality of the learned graph adjacency matrix. Again, our model produces the lowest MAE and RMSE scores among the baselines. On average, it beats DSTAGNN by 10.61% and 7.13% in MAE on PeMSD4 and PeMSD8, respectively.

Table 4. MAE and RMSE of Traffic Speed Prediction on PeMSD8 for the Next Four Horizons

Model	Dataset	PeMSD8 (15/30/45/60 min)	
	Metrics	MAE	RMSE
ARIMA		2.22	4.57
DCRNN		1.16/1.52/1.69/1.88	2.55/3.53/4.10/4.47
STGCN		1.21/1.62/1.79/1.85	3.21/3.74/3.94/4.19
ASTGCN		1.53/1.71/1.85/1.94	3.22/3.75/3.98/4.25
Graph WaveNet		1.19/1.58/1.81/1.97	3.17/3.69/3.92/4.13
AGCRN		1.51/1.66/1.82/1.88	3.21/3.68/3.86/4.31
STGMN		1.23/1.54/1.71/1.84	3.25/3.62/4.13/4.39
iDCGCN		1.22/1.45/1.67/1.80	3.15/3.48/4.03/4.37
DSTAGNN		1.19/1.42/1.63/1.79	3.10/3.42/3.97/4.35
AJSTGL		**1.13/1.32/1.51/1.64**	**2.37/2.77/3.17/3.28**

Table 5. Ablation Study MAE and RMSE for Average Arrival Delay on RCOTP and Traffic Flow on PeMSD4 and PeMSD8 Datasets

Method	Dataset	RCOTP		PeMSD4		PeMSD8	
	Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE
w/o SGL		7.99	15.83	19.01	30.88	14.50	23.64
w/o NSDM		8.27	16.37	19.66	31.94	15.00	24.45
w/o AGG		8.32	16.47	19.78	32.14	15.10	24.61
w/o GR		8.12	16.08	19.31	31.37	14.73	24.02
w/o UC		7.93	15.70	18.85	30.63	14.39	23.45
w/o AGCN		8.11	16.05	19.27	31.32	14.71	23.97
w/o S2SFM		7.84	15.52	18.64	30.29	14.23	23.19
w/o STGTM		8.01	15.86	19.04	30.95	14.53	23.69
AJSTGL		**7.60**	**12.84**	**18.07**	**29.37**	**13.79**	**22.48**

Table 6. Ablation Study MAE for Arrival Delay on RCOTP for the Next 10 Horizons

Method	MAE									
	1	2	3	4	5	6	7	8	9	10
w/o SGL	4.19	4.49	5.49	7.35	8.68	9.79	9.98	9.99	10.00	10.01
w/o NSDM	4.04	4.47	5.74	7.55	8.71	9.85	10.33	10.27	10.75	11.00
w/o AGG	4.63	4.90	5.53	8.04	8.67	9.33	9.86	10.23	10.85	11.18
w/o GR	4.23	4.47	5.47	7.41	8.90	9.51	9.90	10.11	10.45	10.79
w/o UC	4.19	4.44	5.43	7.29	8.80	9.56	9.76	9.87	10.09	10.21
w/o AGCN	4.30	4.85	5.57	6.84	7.59	9.32	10.23	10.38	10.77	11.25
w/o S2SFM	4.77	5.82	6.50	7.62	8.77	9.79	9.81	10.43	10.66	11.25
w/o STGTM	4.02	4.28	5.83	7.55	8.88	9.99	10.74	11.39	11.84	12.95
AJSTGL	**3.98**	**4.26**	**5.21**	**6.99**	**8.25**	**9.30**	**9.49**	**9.50**	**9.52**	**9.52**

4.4.2 Ablation Study. We conduct an ablation study to further investigate each main component's effectiveness in AJSTGL and their impacts on the model performance. Table 5 presents the component-wise impact on all three datasets. Table 6 and Figure 7 demonstrate the ablation study of arrival delay prediction results in the next 10 horizons on RCOTP.

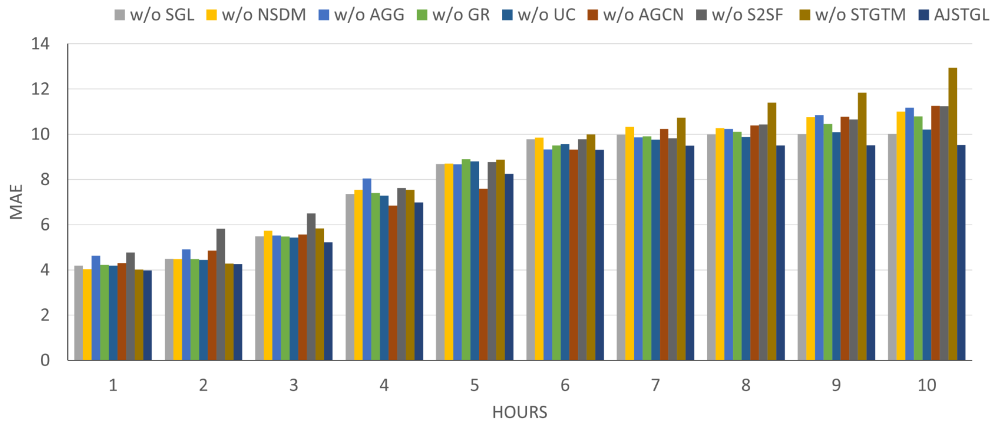


Fig. 7. Ablation study MAE for arrival delay on RCOTP for the next 10 horizons.

Effect of Shifted Graph Laplacian (w/o SGL). In this scenario, the static convolutional graph only utilizes the explicit network topology for the graph adjacency matrix. As demonstrated in Table 5, the MAE of w/o SGL variant increases by 5.18% on average across all datasets. This implies that the shifted graph Laplacian can effectively help the static GCN capture the hidden patterns in the graph signal.

Effect of Node-Specific Dependency Modeling (w/o NSDM). To investigate the effect of node-specific dependency modeling, we remove the node-specific embedding in the weight parameters of all GCN layers. Results in Table 5 show that the MAE of w/o NSDM increases by 8.78% on average across all datasets. In Table 6 and Figure 7, it can also be observed that the performance gain from NSDM is substantially higher in mid- to long-term intervals. We argue that NSDM helps the model learn node-related patterns that could compensate for the lack of historical information in long-term prediction.

Effect of Adaptive Graph Generation (w/o AGG). In the w/o AGG test, the GCN created by adaptive graph generation is removed. We want to study the impact of the hidden patterns captured by the adaptive GCN and observe whether it could complement the intrinsic pre-defined adjacency matrix. As illustrated in Table 5, the MAE of w/o AGG increases by 9.5% on average across all datasets. The substantial performance impact suggests that the pre-defined graph structure could benefit significantly from the granular node dependencies captured by AGG.

Effect of Graph Regularization (w/o GR). Graph regularization enforces the smoothness of the learned graph and further controls the matrix sparsity. The w/o GR test removes the regularization term (set λ to zero in the loss function) and uses only the MAE loss function to optimize the model parameters. As shown in Table 5, the MAE of w/o GR test increases by 6.84% compared to the baseline. Since AJSTGL heavily relies on learning the spatio-temporal dependencies, this indicates that graph regularization could improve the learned graph quality.

Effect of Unidirectional Graph Convolution (w/o UC). In AJSTGL, we transform the standard Chebyshev polynomials-based graph convolution layer by concatenating two convolutional layers with transposed graph Laplacian to capture the unidirectional dataflow patterns. To evaluate its effectiveness, we use the standard graph convolution layer in all GCNs. Table 5 shows that the MAE of w/o UC increases by 4.31% on average. This implies that modeling the unidirectional dataflow on complex real-world data enables the model to capture the in-flow and out-flow patterns.

Effect of Auxiliary GCN (w/o AGCN). We remove the auxiliary GCN, which adopts the spherical distance between nodes as the adjacency matrix to model the contextual weather conditions.

As shown in Table 5, w/o AGCN produces an increase of 6.65% in average MAE score, substantially impacting the overall prediction accuracy. As a result, we observe a substantial correlation between the contextual weather condition and traffic patterns.

Effect of the Sequence-to-Sequence Fusion Model (w/o S2SFM). We further validate that the proposed S2SFM is capable of learning short-term temporal dependencies. In this test, all GCNs are combined with the gated fusion mechanism and passed into DTCG. We remove S2SFM so that AJSTGL solely relies on DTCG to model the temporal node dependency. From Table 5, we can observe a 12.37% increase in average MAE score, which produces the second largest hit on the model performance. Table 6 and Figure 7 further illustrate the impact of S2SFM on short-term prediction. Compared to STGTM, removing S2SFM creates a much greater penalty in short-term prediction performance (first to fourth horizons).

Effect of the Spatio-Temporal Graph Transformer Module (w/o STGTM). The main goal of STGTM is to complement S2SFM on long-term prediction. To evaluate the effectiveness of STGTM, we completely remove it from AJSTGL. Table 5 demonstrates a significant 15.13% increase in average MAE from all datasets. Table 6 and Figure 7 provide more insight from the 10 horizons arrival delay prediction results. It can be observed that STGTM excels at mid- to long-term predictions (5th to 10th horizons) when compared to S2SFM. This outcome can be explained from several aspects. First, instead of relying on the absolute position, the relative positional encoding generates the spatial and temporal embedding of the input sequence in a data-aware manner. This helps the model capture the dynamic hidden patterns from the graph signals. Second, the adjacency matrix constructed by the multi-head attention mechanism learns the spatial node dependencies from high-dimensional latent subspaces, which extends the model's capacity in modeling the hidden spatio-temporal relations. Third, AJSTGL balances short-term and long-term predictions by adopting both S2SFM and STGTM. The gated information fusion module ensures the model learns the optimal weighting when combining the features from the two modules.

5 CONCLUSION AND FUTURE WORK

This article proposed a novel graph learning network for traffic data prediction. We used static and adaptive graph learning modules to improve the pre-defined graph and adaptively learn new graphs to capture complex and dynamic traffic patterns. An auxiliary convolutional graph was adopted to leverage contextual information. We further transformed the standard graph convolutional layer to allow the model to learn unidirectional traffic flow patterns. The sequence-to-sequence fusion module hierarchically combines the parallelized encoders and learns the short-term temporal node dependencies. We also developed the spatio-temporal graph transformer module to complement S2SFM by dynamically capturing the spatio-temporal dependencies in long-term prediction. Experimental results on three real-world datasets demonstrated the excellent performance of our approach compared to other state-of-the-art baselines. Due to the flexibility of the proposed model, we see the potential of extending it to other spatio-temporal data.

REFERENCES

[1] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR abs/1607.06450* (2016). http://arxiv.org/abs/1607.06450

[2] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems* 33 (2020), 17804–17815.

[3] Thomas Bohnstingl, Stanisław Woźniak, Angeliki Pantazi, and Evangelos Eleftheriou. 2023. Online spatio-temporal learning in deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 34, 11 (2023), 8894–8908.

[4] Miaomiao Cao, Victor O. K. Li, and Vincent W. S. Chan. 2020. A CNN-LSTM model for traffic speed prediction. In *Proceedings of the 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring'20)*. IEEE, Los Alamitos, CA, 1–5.

[5] Changlu Chen, Yanbin Liu, Ling Chen, and Chengqi Zhang. 2023. Bidirectional spatial-temporal adaptive transformer for Urban traffic flow forecasting. *IEEE Transactions on Neural Networks and Learning Systems* 34, 10 (2023), 6913–6925.
[6] Zhirong Duan, Yan Yang, and Wei Zhou. 2021. Multi-view spatial-temporal adaptive graph convolutional networks for traffic forecasting. In *Proceedings of the 2021 16th International Conference on Intelligent Systems and Knowledge Engineering (ISKE'21)*. IEEE, Los Alamitos, CA, 35–41.
[7] Hao Fan, Chuanfeng Zhao, and Yikun Yang. 2020. A comprehensive analysis of the spatio-temporal variation of urban air pollution in China during 2014–2018. *Atmospheric Environment* 220 (2020), 117066.
[8] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 922–929.
[9] Fan Hou, Yue Zhang, Xinli Fu, Lele Jiao, and Wen Zheng. 2021. The prediction of multistep traffic flow based on AST-GCN-LSTM. *Journal of Advanced Transportation* 2021 (2021), 1–10.
[10] Oleksii Hrinchuk, Mariya Popova, and Boris Ginsburg. 2020. Correction of automatic speech recognition with transformer sequence-to-sequence model. In *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'20)*. IEEE, Los Alamitos, CA, 7074–7078.
[11] Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. 2019. DSANet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2129–2132.
[12] Jayson S. Jia, Xin Lu, Yun Yuan, Ge Xu, Jianmin Jia, and Nicholas A. Christakis. 2020. Population flow drives spatio-temporal distribution of COVID-19 in China. *Nature* 582, 7812 (2020), 389–394.
[13] Honghua Jiang, Chuanyin Zhang, Yongliang Qiao, Zhao Zhang, Wenjing Zhang, and Changqing Song. 2020. CNN feature based graph convolutional network for weed and crop recognition in smart farming. *Computers and Electronics in Agriculture* 174 (2020), 105450.
[14] Manrui Jiang, Wei Chen, and Xiang Li. 2021. S-GCN-GRU-NN: A novel hybrid model by combining a spatiotemporal graph convolutional network and a gated recurrent units neural network for short-term traffic speed forecasting. *Journal of Data, Information and Management* 3 (2021), 1–20.
[15] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.
[16] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17): Conference Track*. https://openreview.net/forum?id=SJU4ayYgl
[17] Shiyong Lan, Yitong Ma, Weikang Huang, Wenwu Wang, Hongyu Yang, and Pyang Li. 2022. DSTAGNN: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *Proceedings of the International Conference on Machine Learning*. 11906–11917.
[18] Sheng Li, Fengxiang He, Bo Du, Lefei Zhang, Yonghao Xu, and Dacheng Tao. 2019. Fast spatio-temporal residual network for video super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10522–10531.
[19] Yiming Li, Changhong Fu, Fangqiang Ding, Ziyuan Huang, and Geng Lu. 2020. AutoTrack: Towards high-performance visual tracking for UAV with automatic spatio-temporal regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11923–11932.
[20] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18): Conference Track*.
[21] Zhuoling Li, Gaowei Zhang, Lingyu Xu, and Jie Yu. 2021. Dynamic graph learning-neural network for multivariate time series modeling. *CoRR abs/2112.03273* (2021). https://arxiv.org/abs/2112.03273
[22] Lyuchao Liao, Zhiyuan Hu, Yuxin Zheng, Shuoben Bi, Fumin Zou, Huai Qiu, and Maolin Zhang. 2022. An improved dynamic Chebyshev graph convolution network for traffic flow prediction with spatial-temporal attention. *Applied Intelligence* 52, 14 (2022), 16104–16116.
[23] Daoguang Liu, Shen Hui, Li Li, Zhigui Liu, and Zhiming Zhang. 2020. A method for short-term traffic flow forecasting based on GCN-LSTM. In *Proceedings of the 2020 International Conference on Computer Vision, Image, and Deep Learning (CVIDL'20)*. IEEE, Los Alamitos, CA, 364–368.
[24] Qingjian Ni and Meng Zhang. 2022. STGMN: A gated multi-graph convolutional network framework for traffic flow prediction. *Applied Intelligence* 52, 13 (2022), 15026–15039.
[25] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Xinmei Tian, and Tao Mei. 2019. Learning spatio-temporal representation with local and global diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12056–12065.

[26] Navin Ranjan, Sovit Bhandari, Hong Ping Zhao, Hoon Kim, and Pervez Khan. 2020. City-wide traffic congestion prediction based on CNN, LSTM and transpose CNN. *IEEE Access* 8 (2020), 81606–81620.
[27] Selim Reza, Marta Campos Ferreira, J. J. M. Machado, and João Manuel R. S. Tavares. 2022. A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks. *Expert Systems with Applications* 202 (2022), 117275.
[28] Robert H. Shumway and David S. Stoffer. 2017. ARIMA models. In *Time Series Analysis and Its Applications*. Springer, 75–163.
[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017), 1–11.
[30] Shui-Hua Wang, Vishnu Varthanan Govindaraj, Juan Manuel Górriz, Xin Zhang, and Yu-Dong Zhang. 2021. Covid-19 classification by FGCNet with deep feature fusion from graph convolutional network and convolutional neural network. *Information Fusion* 67 (2021), 208–229.
[31] Tianyi Wang and Shu-Ching Chen. 2021. Hierarchical multimodal fusion network with dynamic multi-task learning. In *Proceedings of the IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI'21)*. IEEE, Los Alamitos, CA, 208–214.
[32] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*. 1907–1913. https://doi.org/10.24963/ijcai.2019/264
[33] Yongsang Yoon, Jongmin Yu, and Moongu Jeon. 2022. Predictively encoded graph convolutional network for noise-robust skeleton-based action recognition. *Applied Intelligence* 52, 3 (2022), 2317–2331.
[34] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 3634–3640. https://doi.org/10.24963/ijcai.2018/505
[35] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2020. Effective travel time estimation: When historical trajectories over road networks matter. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2135–2149.
[36] Albert Zeyer, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney. 2019. A comparison of transformer and LSTM encoder decoder models for ASR. In *Proceedings of the 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU'19)*. IEEE, Los Alamitos, CA, 8–15.
[37] Weibin Zhang, Yinghao Yu, Yong Qi, Feng Shu, and Yinhai Wang. 2019. Short-term traffic flow prediction based on spatio-temporal analysis and CNN deep learning. *Transportmetrica A: Transport Science* 15, 2 (2019), 1688–1711.
[38] Wei Zhang, Fenghua Zhu, Yisheng Lv, Chang Tan, Wen Liu, Xin Zhang, and Fei-Yue Wang. 2022. AdapGL: An adaptive graph learning algorithm for traffic prediction based on spatiotemporal neural networks. *Transportation Research Part C: Emerging Technologies* 139 (2022), 103659.
[39] Yu-Dong Zhang, Suresh Chandra Satapathy, David S. Guttery, Juan Manuel Górriz, and Shui-Hua Wang. 2021. Improved breast cancer classification through combining graph convolutional network and convolutional neural network. *Information Processing & Management* 58, 2 (2021), 102439.
[40] Chuanpan Zheng, Xiaoliang Fan, Shirui Pan, Zonghan Wu, Cheng Wang, and Philip S. Yu. 2021. Spatio-temporal joint graph convolutional networks for traffic forecasting. *arXiv preprint arXiv:2111.13684* (2021).
[41] Chuanpan Zheng, Xiaoliang Fan, Shirui Pan, Zonghan Wu, Cheng Wang, and Philip S. Yu. 2021. Spatio-temporal joint graph convolutional networks for traffic forecasting. *CoRR abs/2111.13684* (2021). https://arxiv.org/abs/2111.13684
[42] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. GMAN: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1234–1241.

Received 4 March 2023; revised 15 September 2023; accepted 23 November 2023