Wiley
Journal of Advanced Transportation
Volume 2025, Article ID 5195875, 21 pages
https://doi.org/10.1155/atr/5195875

WILEY

Research Article

Optimal Graph Information Fused Graph Attention Network for Traffic Flow Forecasting

Xing Xu,[1] Luchen Fei,[2] Yun Zhao ,[1] and Xiaoshu Lü[3,4]

[1]*School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China*
[2]*School of Mechanical and Energy Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China*
[3]*Department of Civil Engineering, Aalto University, Espoo 02130, Finland*
[4]*Department of Electrical Engineering and Energy Technology, University of Vaasa, Vaasa 65380, Finland*

Correspondence should be addressed to Yun Zhao; zy_super0201@163.com

Received 26 June 2024; Accepted 13 March 2025

Academic Editor: Yajie Zou

To manage and make decisions about intelligent transportation systems more efficiently, accurate traffic flow forecasting is necessary. Traffic flow forecasting has complex spatial correlation and time dependence. Most current research models are based on a predefined graph structure with a priori knowledge for prediction, which cannot well extract the hidden spatial relationships in traffic data. In this paper, we propose the Optimal Graph Information Fused Graph Attention Network (OGIF-GAT). Specifically, we learn the actual connections between nodes and the hidden spatial relationships through the multigraph feature fusion structure. Next, we design a new graph attention network (GAT), which improves the problem of ignoring edge features in the graph structure in the traditional GAT model and considers their edge features when estimating the correlation of each neighboring node pair: the effect that the distance factor between neighboring nodes has on the spatial correlation. In addition, we use the temporal hybrid transformer (THT) to learn temporal dependencies. Extensive experiments on four public transportation datasets (PeMS04, PeMS08, PeMS-BAY, and METR-LA) demonstrate that our model achieves the optimal level of traffic flow prediction accuracy on all of them and is shown to have strong generalization ability. Compared to STSGCN, the mean absolute error (MAE) decreases by 7.9%, 10.3%, 33.2%, and 19.6%, respectively.

Keywords: deep learning; graph attention; spatiotemporal forecast; traffic forecast

1. Introduction

Intelligent transportation systems (ITS) [1, 2] have received great attention as people have higher requirements for travel safety and travel efficiency. Traffic flow forecasting can accurately predict the future traffic flow through the historical traffic flow information in real time. This enables urban traffic management authorities to rationally guide and plan traffic flow so that urban congestion can be effectively alleviated.

But now, there are still many challenges for accurate and efficient traffic flow forecasting. First, due to the complexity of the road network, it becomes very difficult to explicitly describe the influence of different road spaces on each other at a given moment in time. Second, in the time dimension, because of the complex temporal periodicity, both short-period and long-period historical information can have an impact on the prediction of future moments. Therefore, accurately mining spatial and temporal correlations within traffic patterns is the key to achieving accurate and efficient traffic flow forecasting.

In the temporal dimension, the researcher utilized recurrent neural networks (RNNs) [3, 4] and its variants such as LSTM [5, 6] and GRU [7] to capture temporal correlation.

There were also researchers who used a temporal convolutional network (TCN) [8] instead of RNN to solve the problem of time series forecasting with the ability to compute the output in parallel. Transformer was first proposed in the field of NLP; it can support parallel operations, is good at modeling long sequences, and has flexibility and interpretability, so there were also researchers who used the attention mechanism to model historical information in traffic information, such as ASTTGN [9], which used transformer to capture multiple time-step temporal correlations separately. However, these methods were only based on temporal dependence, but we still need to model spatial correlation using different spatial relationships to deal with the effects of road topology.

Recently, graph neural networks (GNNs) have achieved excellent results in the task of dealing with road topology. For example, STGCN [10] applied complete convolutional structures to solve timing prediction problems in the domain of graph structures; T-GCN [11] used graph convolutional networks (GCNs) and gated recursive units (GRUs) to learn complex topologies and temporal correlations, respectively. It could be said that most of the existing GNN models have been studied by constructing graphs that have been determined by predefined measurements, such as adjacency matrices based on road adjacencies and Euclidean structure matrices based on distances to urban roads. However, this may lead to insufficient traffic spatial feature extraction, e.g., two sensors that are far away or not physically connected yet exhibit similar traffic flows between them, or two sensors that are close or physically connected yet do not exhibit similar traffic flows between them. STGAFormer [12] introduced adaptive adjacency matrices to capture these hidden spatial correlations, but this learns graph structures that will remain unchanged from period to period, ignoring the dynamics of the graph structure. AdpSTGCN [13] used an attention mechanism to flexibly generate multiview feature maps, obtaining local dynamic spatial features through a digitally driven approach, yet ignoring the overall static spatial hidden features such as urban POIs. COGCN [14] used OAG to extract global spatial features of the road and ODPG to extract local spatial features of the road and combined them using a comparative learning algorithm. In addition, Peng et al. [15] used ANOVA and conditional regression models to explore the effects of two types of inclement weather on traffic parameters, which can help the transportation department formulate effective countermeasures for intelligent traffic control under inclement weather conditions.

In a word, all current research efforts have the following shortcomings, thus leading to unsatisfactory results.

1.1. Fixed Static Spatial Correlation. Current studies are using distance measurements of predefined road network structures to capture spatial correlations. On the one hand, this approach requires a great deal of domain knowledge and the fact that these road structures do not necessarily represent real road structure information. Circumstances such as weather, festival periods, accidents, congestion, and road construction can affect the spatial pattern of the road network. On the other hand, in most cases, the predefined graphs are mostly determined by the Euclidean distances between regions and the connectivity between road nodes, whereas the factors affecting the relationships between regions cannot be determined only by the distance and connectivity factors but are also influenced by many other hidden factors, as shown in Figure 1, although there is no direct road connection between the home and the supermarket. However, there is a hidden spatial correlation between homes and supermarkets because people need to shop.

In addition, the predefined road network structure should be dynamic, although the static spatial relationships between roads can be adequately extracted. As shown in Figure 2, a solid red line indicates a high correlation and green is a low correlation. Although two neighboring sensors (e.g., Sensors 1 and 2) exhibit a high degree of correlation between them, this correlation is affected by time. This forecasting of future traffic conditions based on the spatial correlation between sensors dynamically over time is quite challenging.

1.2. Attentional Mechanisms Do Not Perform Well Enough in Graph Representation Learning. With the proposal of transformer, attention mechanisms are employed to identify intricate temporal patterns and spatial relationships in traffic data. Traffic road spatial information is often represented as a graph domain; today, the most effective way to utilize attention in graph representation is to replace some key modules in the classical GNN variants, such as feature aggregation, with SoftMax attention. However, the attention mechanism focuses only on the semantic similarity between Node *i* and other nodes but ignores the node's position in the graph structure, connectivity, and relationships between pairs of nodes. In a real-world scenario of traffic flow forecasting, the location of each node road, its connectivity with other node roads, and the distance between the nodes are important factors used to show spatial correlation.

To solve the above difficulties and challenges, we propose a model for forecasting traffic flow: Optimal Graph Information Fused Graph Attention Network (OGIF-GAT). In the spatial dimension, we design a multigraph feature fusion structure. The structure not only contains a priori road information representing connected relationships between nodes but also learns hidden dynamic spatial relationships based on node attributes through a corrected cosine similarity algorithm. In addition, the graph attention network (GAT) can only extract information between nodes and cannot extract information on the edges (e.g., the distance between neighboring nodes), we improve the traditional GAT model and capture the spatial features of the data.

In the time dimension, we use temporal gated convolution (TGC) and transformer models to capture the nonlinear correlations in short-term and long-term time series, respectively, and obtain the fusion model temporal hybrid transformer (THT). Finally, a gating mechanism is employed to combine the two models and extract both spatial and temporal characteristics.

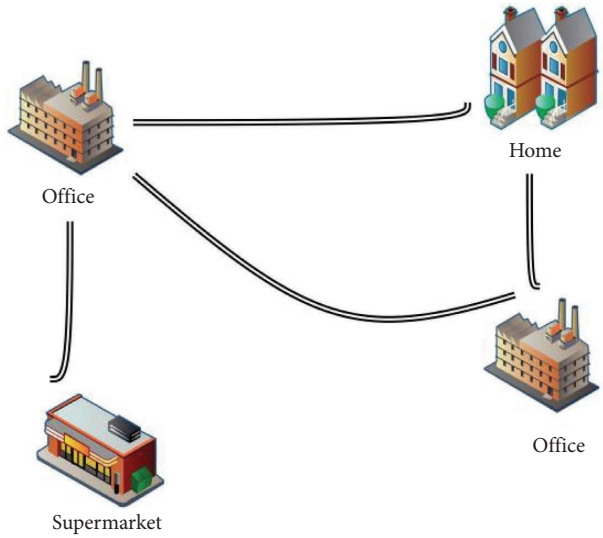

Figure 1: Hidden spatial correlation between home and supermarkets.

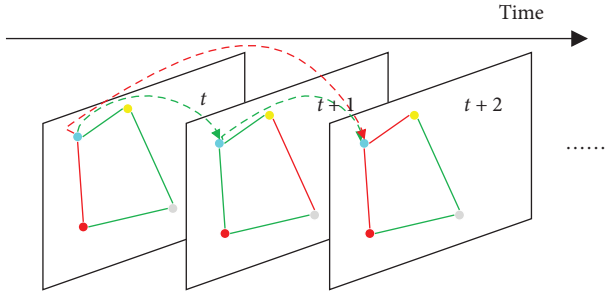

Figure 2: Dynamic spatial correlation.

We summarize the contributions made in this paper as follows:

1. We propose a multigraph feature fusion structure that can learn an optimal graph structure from a combination of dynamic and static aspects, respectively. On the dynamic side, we design a corrected cosine similarity algorithm to learn the hidden dynamic spatial correlations from node feature attributes. On the static side, we extract hidden static spatial relationships and real road connectivity relationships using an adaptive stabilization graph structure and a road connectivity graph structure, respectively.
2. We improve the traditional GAT by encoding edge features between neighboring nodes into the attention layer, allowing the newly improved GAT to consider edges connecting node pairs when computing their correlations. In addition, we construct a network model that can learn multigraph feature fusion structures: the adaptive graph attention network (AGAT), which we use to capture both static and dynamic features of the space.
3. We combine the AGAT and THT models as OGIF-GAT and evaluate the model on two real-world traffic datasets. Experiments demonstrate that our model outperforms state-of-the-art baseline methods.

The remainder of the paper is organized as follows: Section 1 reviews what is relevant to the article, as well as existing research. Section 3 presents the problem to be solved and describes the model we designed. In Section 4, we experiment the model with real-world traffic datasets and compare it with some of the current state-of-the-art methods. Finally, in Section 5, we summarize the research in this paper and have an outlook for future work.

2. Related Work

2.1. Traffic Flow Forecasting. Traffic flow is an important element of traffic data, and traffic flow refers to the number of traffic entities (vehicles) passing through a certain sensor or roadway cross section per unit of time. Traffic flow usually reflects the traffic pressure on the road. Early traditional researchers used statistical theoretical approaches to build mathematical and physical analytical models for traffic flow prediction studies, mainly including historical averaging (HA), traditional time series (e.g., ARIMA [16]), and Kalman filtering models [17, 18]. These methods are based on linear dependencies, which are usually assumed to be linear in the variation of data in traffic flow forecasting, and perform poorly for complex nonlinearly varying traffic flow forecasting. Today, more and more researchers have shifted their research efforts to machine learning and deep learning. The k-nearest neighbor (KNN) algorithm [19] makes predictions based on the traffic flow of neighboring points by calculating the distance between the point to be predicted and the historical data points, and support vector machines (SVMs) [20] makes traffic flow predictions by constructing a regression model close to the true value, which is capable of dealing with complex nonlinear relationships. Sequential models such as RNN and its variants LSTM [5] and GRU [7] can effectively mine the dynamics of nonlinear traffic data and perform well in traffic flow sequence data in terms of temporal correlation, but simple RNN models are unable to model the spatial information of traffic data. To solve this problem, some works apply the convolutional neural network (CNN) to model the spatial correlation of traffic flow forecasting. Zhang [21] et al. proposed to abstract the urban area into grid data of the same size and then used convolutional operations to mine the spatial information, but such an approach ignores the topology of the traffic network. GNN can effectively solve the problem of traffic network topology, so researchers have proposed to abstract the actual traffic road network into a topological graph structure and use the graph convolution model GCN [10, 22] to mine the spatial correlation in the topological graph data.

2.2. Attention Mechanism. With the continuous research in the field of deep learning, the proposal of transformer [23] and BERT [24] gradually replaces the dominance of RNN in natural language and time series data problems, while the models proposed based on transformer, VIT [25], and Swin transformer [26] excel in image classification and match the performance of top convolutional networks, all while consuming fewer computational resources during training. So

researchers have also used the transformer model in traffic flow forecasting tasks, and the most important thing about the transformer model is that it uses the attention mechanism. Zou [27] proposed the Bayesian model averaging (BMA) method, which combines the traffic speed prediction results from three transformer models to identify stable long-term speed trends, and BMA demonstrated superior accuracy in short-term traffic speed prediction compared to the three single transformer models. The core of the attention mechanism lies in selecting the most relevant information for the task from the overall data. In the context of sequential data, it allows each element to interact with others throughout the sequence, not limited to adjacent elements, and dynamically captures long-range dependencies by evaluating the relative significance of each element with respect to the others. Thus, the attention mechanism is also good at globally capturing traffic data spatiotemporal properties in traffic data. Zong et al. [28] proposed PSTTransformer, which integrates spatiotemporal attention blocks to capture dynamic spatiotemporal dependencies and not only integrates self-attention mechanism to embed periodic information but also synergizes GCN and spatial self-attention mechanism through the gating mechanism. The attention mechanism, however, neglects the topological graph structure of transportation networks in the spatial dimension, and thus, GANs are proposed.

2.3. GANs. A GNN is a model designed to capture dependencies in a graph by propagating information across its nodes. Velikovi et al. [29] proposed a GAT, which dynamically assigns different weights to neighboring nodes in graph structured data, thus aggregating the information of neighboring nodes, and is able to solve the problem of sharing one and the same convolutional kernel parameter for all nodes in the neighborhood in GCN. However, the traditional GAT model can only focus on the layer of neighboring features by aggregating the node features, while ignoring edge features, such as the distance between neighboring sensors. Therefore, we improve the original GAT model and propose a new GAT model that can aggregate the features of neighboring nodes as well as the features of edges connected to two nodes.

2.4. Self-Adaptive Adjacency Matrix. Graph WaveNet [30] model literature proposes a self-adaptive adjacency matrix, while AGCRN [31] is also similar to Graph WaveNet by learning the embedding matrix. The self-adaptive adjacency matrix without needing any a priori information adaptively generates the adjacency matrix by performing stochastic gradient descent learning from end to end, randomly initializing two source and target nodes with learnable parameters and obtaining the spatial dependencies between the source and target nodes by using the adaptive adjacency matrix formula: $\mathrm{A}^{\mathrm{adp}} = \mathrm{SoftMax}(\mathrm{ReLU}(\mathrm{E}_1\mathrm{E}_2^{\mathrm{T}}))$. The adjacency matrix obtained by this method represents the hidden spatial relations. However, this direct design of the graph matrix generated from the learnable parameters, without considering the node attributes, reduces the accuracy of the adjacency matrix and makes the model difficult to optimize. Therefore, in our study, we will learn the dynamic optimal graph matrix from the node attributes and also still consider the predefined road network structure using the distance measure as a static graph structure.

3. Methodology

3.1. Problem Definition. The objective of traffic flow forecasting in this paper is to forecast the traffic conditions in the future period for each area based on the historical data that have been given for different node areas. We define a traffic network as a weighted directed graph $\mathrm{G} = (\mathrm{V}, \mathrm{E}, \mathrm{A})$, where $\mathrm{V} = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_{\mathrm{N}}\}$ is the set of $|\mathrm{V}| = \mathrm{N}$ nodes, N denotes the number of road sensors, E denotes the set of edges, $\mathrm{A} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ is a weighted adjacency matrix, and $A_{i,j}$ denotes the relationship between node v_i and node v_j. We denote the traffic flow data of sensor v_{n} at moment t as X_t^n.

Traffic flow forecasting is a typical time series forecasting task, where the traffic flow data of a given N nodes over a history of T time steps can be represented as $\mathcal{X} = \{X_{t-T+1}, X_{t-T+2}, X_{t-T+3}, \ldots, X_t\} \in \mathbb{R}^{T\times N\times H}$, where H denotes the traffic characteristics. We use the known values of traffic characteristics to forecast the traffic characteristics of all nodes for the future T' time step, denoted as $\widehat{y} = \{X_{t+1}, X_{t+2}, \ldots, X_{t+T'}\} \in \mathbb{R}^{T'\times N\times H}$, which we can define by the following equation:

$$\widehat{y} = F(\mathcal{X}, G), \tag{1}$$

where F is the function to be learned by our model.

3.2. Overview. Figure 2 depicts the overall architecture of the OGIF-GAT framework, which includes an input layer, four stacked spatiotemporal layers (STLs), and a spatiotemporal gating layer. Among them, an STL contains a THT module and an AGAT module. The THT module consists of a transformer module with a multihead attention mechanism and a gated temporal convolution module to extract the long-term temporal dependence and short-term temporal dependence in the traffic flow data, respectively. AGAT module, on the other hand, is a self-developed graph network model capable of learning the degree of correlation between the nodes of multigraph feature fusion structures, which allows the capture of data features in both static and dynamic space.

We express the traffic flow forecasting problem examined in this paper in mathematical form as follows:

$$G^* = g(\mathcal{X}, G), \tag{2}$$

$$\widehat{y} = F(\mathcal{X}, G^*), \tag{3}$$

where G^* is a multigraph feature structure learned from traffic flow data $\mathcal{X}$ and a predefined graph matrix G composed of road connection relationships, as shown in Figure 3. Then, the multigraph feature structure G^* and historical traffic flow data $\mathcal{X}$ are used to predict the future traffic flow $\widehat{y}$.

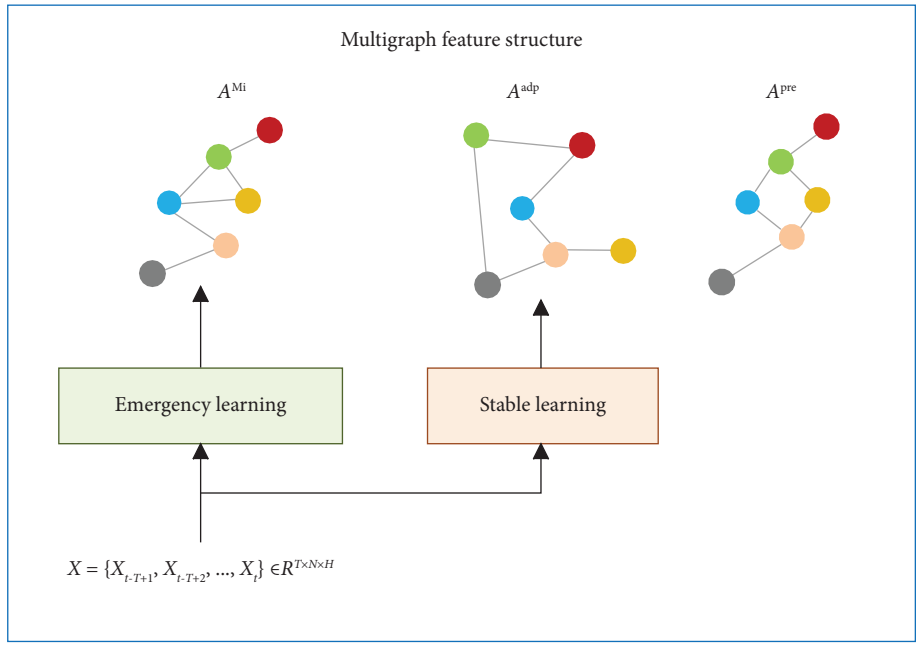

Figure 3: Multigraph feature structure learning.

Next, we introduce the multigraph feature fusion structure in detail, as well as the AGAT module, and then, we will introduce the THT module and finally the flow of the whole model.

3.3. AGATs

3.3.1. Multigraph Feature Structure Learning. In general, two interconnected roads have similar characteristics; however, in real life, two regions that are very far away from each other (e.g., two business districts that are very far away from each other, etc.) will also have implied characteristic correlations, and predefined adjacency matrices, which only have information about the connectivity of the roads and the distances, do not show these implied factors. Also, current research related to graph learning [30, 31] generates graph structures without considering the nodes' own attributes, making the model difficult to optimize. What we want to do is to learn the optimal graph structure from node properties.

The principle of this learning method is as follows: We map the node attributes (traffic flow, speed, and occupancy) in the input features into a high-dimensional node space to get a high-dimensional hidden attribute vector, which enables better mining of node attribute hidden features. Then, we design the corrected cosine similarity algorithm to calculate the similarity between different nodes to get the implied correlation between different nodes. Finally, the node features are cosine similarity computed two by two to get the graph adjacency matrix.

Existing similarity algorithms include inner product, cosine similarity, Euclidean distance, Pearson's correlation coefficient, and so on, while similarity algorithms such as Euclidean distance and Manhattan distance give more consideration to the spatial distance between the nodes, and for the similarity of hidden attributes in the nodes' attributes, the inner product or cosine similarity is more appropriate. However, cosine similarity has the property of being insensitive to the absolute magnitude of specific values, which can also cause some problems in our traffic flow prediction, for example, the flow features and speed features in Node *i* are small (e.g., [19, 20]), while the flow features and speed features in Node *j* are large (e.g., [49, 50]). Although the cosine similarity between the two nodes is extremely high, however, this does not match the real conditions, it is obvious that Section *j* is a more high-speed section than Section *i*. Perhaps Section *i* is very narrow and Section *j* is very wide. To address the problem that the cosine similarity algorithm is insensitive to the absolute magnitude of specific values, which leads to the inability to correctly determine whether two road segment nodes are similar or not, we designed the corrected cosine similarity algorithm to obtain the implicit correlation between different nodes. The principle of the corrected cosine similarity algorithm is as follows.

Because the cosine similarity algorithm focuses only on the direction of the vector and not on the magnitude of the specific values of the data in the vector, node vectors that are close to each other as long as the two directions are close to each other will have a strong cosine similarity. To distinguish between vectors with the same direction but too large and too small values, we first calculate the mean value on each dimension of the feature vector, and then, each node vector subtracts the mean value on the dimension to get a new node vector in each dimension, so that features with larger than the mean value on the dimension and those with smaller than the mean value can be more differentiated in the direction of the vector, thereby reducing the similarity between vectors with too large an absolute value and vectors with too small an absolute value. The algorithm effectively distinguishes between high-speed and low-speed road sections, or

wide and narrow road sections, making the calculation of the implied correlation between different nodes more accurate. In addition, the cosine similarity algorithm has to compute the modulus product of two node vectors in the denominator, which is not conducive to the computation of the final adjacency matrix. So we unitize the new node vectors to get the corrected node eigenvectors so that the product of modulus lengths is 1. By doing so, we can simplify the cosine similarity algorithm to the inner product similarity algorithm. Finally, the corrected node feature vectors are cosine similarity computed two by two (inner product computation) to obtain the adjacency matrix A^{Mi}. The specific implementation formula is given in "emergency graph structure learning."

We take the adjacency matrix, which represents the hidden feature relations, together with the predefined adjacency matrix obtained from the road connectivity relations, to form a complete multigraph feature structure used to achieve information complementation, as shown in Figure 3. Specifically, the module consists of three parts: emergency graph structure, stabilization graph structure, and road connection graph structure. The emergency graph structure is designed to be able to detect drastic changes arising at a given moment and to generate a graph adjacency matrix for that moment to capture that fluctuation; the stabilization graph structure is designed to learn a relatively stable graph adjacency matrix over the entire time period by using the entire historical information; and the road connection graph structure is still very important because the actual connectivity between the roads and Euclidean distance between the nodes play a vital role in the traffic flow forecasting's spatial correlation. Below, we present the main elements of these three components.

3.3.1.1. Emergency Graph Structure Learning. The emergency graph structure learning is designed to learn the correlation between nodes when a drastic change occurs at a certain moment in time (during traffic accidents, traffic jams, or drastic weather changes), which requires that our graph learning structure be real time and dynamic. We describe the fluctuation of emergency events by mining the relevant information in the node attributes and then use the corrected cosine similarity algorithm proposed above to obtain the hidden correlations between different nodes to generate the emergency graph adjacency matrix at that moment. The method in detail is presented below.

First, we input the node attribute data $\mathcal{X} = \{X_{t-T+1}, X_{t-T+2}, X_{t-T+3}, \ldots, X_t\} \in \mathbb{R}^{T\times N\times H}$, since these data represent the H-dimensional traffic flow data of a given N nodes over a history of T time steps, and the emergency fluctuations mined through the node attributes must be real time. The generated graph adjacency matrix is located at $t - T + 1 \sim t$ and varies with t.

As shown in Figure 4, in order to map the raw data into the hidden layer feature space, we use a fully connected network to map the H-dimension (traffic flow condition) attribute features into the high-dimension node feature space. In this way, our data have the ability to be learned in the hidden space using the following equation:

$$H = FC(\mathcal{X}) \in \mathbb{R}^{T\times N\times D}. \tag{4}$$

Next, in order to integrate the spatial relationships of the nodes in the T time period, we use a convolution operation that sets the kernel size to a dimension T equal to the length of the historical sequence.

$$M = \mathrm{AGGREGATE}(H) \in \mathbb{R}^{N\times D'}. \tag{5}$$

In this way, we obtain a comprehensive attribute characterization M_i for each node in real time yet dynamically at moment t. To mine the correlation between different nodes, we need to design a metric learning method; metric learning [32] is used to learn the degree of similarity between the data features of two nodes, and the more similar the two nodes indicate closer proximity and more correlation. We use the corrected cosine similarity algorithm proposed above to learn the implicit relationships between nodes to generate the neighbor matrix for emergencies. The following is a specific implementation formula for the corrected cosine similarity algorithm:

First compute the mean $M_{\mathrm{mean}} \in \mathbb{R}^{D'}$ of the feature vector $M \in \mathbb{R}^{N\times D'}$ on each dimension d':

$$M_{\mathrm{mean}} = \frac{\sum_{i=1}^{N} M_i}{N}. \tag{6}$$

Then, each node vector M_i is subtracted from the mean in each dimension to get a new node vector $M_i' \in \mathbb{R}^{D'}$:

$$M_i' = M_i - M_{\mathrm{mean}}. \tag{7}$$

Next, the new node vector $M_i{}'$ is unitized and the corrected node feature vector $M_{\mathrm{adj,i}} \in \mathbb{R}^{D'}$ is obtained as follows:

$$M_{\mathrm{adj},i} = \frac{M_i'}{|M_i'|}. \tag{8}$$

Finally, the corrected node feature vectors are cosine similarity computed two by two:

$$\cos\left(M_{\mathrm{adj},i}, M_{\mathrm{adj},j}\right) = \frac{M_{\mathrm{adj},i} M_{\mathrm{adj},j}^T}{\left|M_{\mathrm{adj},i}\right|\left|M_{\mathrm{adj},j}\right|}. \tag{9}$$

Because the node feature vectors are treated by vector unitization, we can simplify equation (9) to an inner product calculation as follows:

$$A^{\mathrm{Mi}}[i, j] = M_{\mathrm{adj},i} M_{\mathrm{adj},j}^T. \tag{10}$$

Equation (10) represents the cosine similarity between Node i and Node j, where $M_{\mathrm{adj},i}$ represents the vector features of Node i after unitization and $M_{\mathrm{adj},j}$ represents the vector features of Node j after unitization.

Finally, we utilize the ReLU activation function to filter out weak connections.

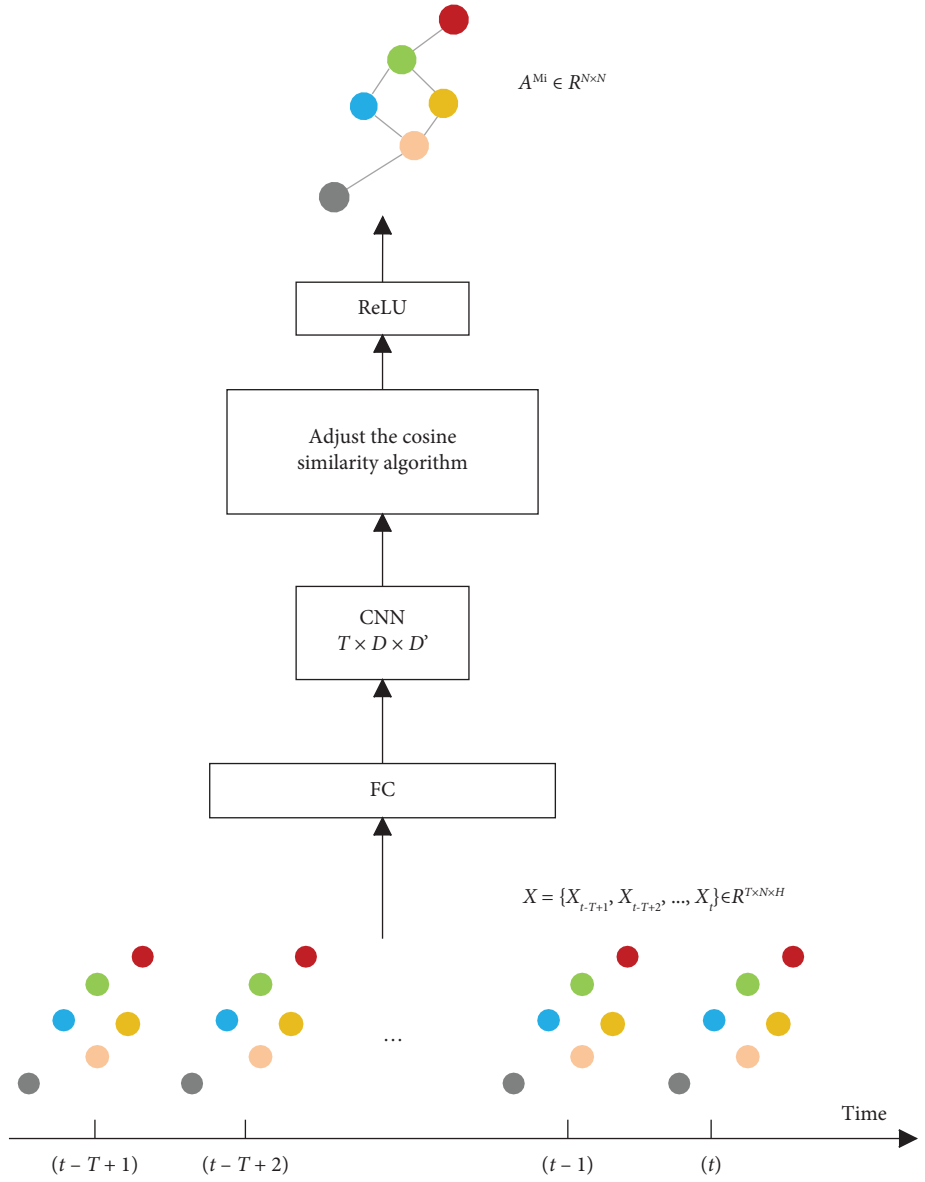

Figure 4: Emergency graph structure learning.

$$A^{\mathrm{Mi}} = \mathrm{ReLU}\left(M_{\mathrm{adj}} M_{\mathrm{adj}}^{T}\right) \in \mathbb{R}^{N\times N}. \tag{11}$$

We use the learned A^{Mi} for the emergency graph adjacency matrix we need.

3.3.1.2. Stabilization Graph Structure Learning. Stabilization graph structure learning is learning graph structures that are stable over time based on information from the entire history. The actual connectivity between roads and the Euclidean distance between nodes is the graph structure information that belongs to the long-term stability, but some implicit correlations between nodes are missing, which are not sudden, but exist stably over an extended period of time, (e.g., POI distributions and regional functions). Our stabilization graph structure learning, on the other hand, focuses on learning these long-term stable but implicit correlations between nodes and generating an adjacency matrix to perform the representation of that relationship.

For this part, we use the learning method of Graph WaveNet [30] with an adaptive adjacency matrix. First, randomly initialize the source node embedding dictionary E_1 and the target node embedding dictionary E_2. Multiply E_1 and E_2 to get the spatial dependency weights between the source and target nodes, the ReLU activation function is used to eliminate the weak connections, and the SoftMax function is used to normalize them. The specific formula is as follows:

$$A^{\mathrm{adp}} = \mathrm{SoftMax}\left(\mathrm{ReLU}\left(E_1 E_2^{T}\right)\right) \in \mathbb{R}^{N\times N}. \tag{12}$$

We use the learned A^{adp} as the adjacency matrix of the implicit relationships between nodes in the stabilization case.

3.3.1.3. Road Connection Graph Structure. We do not give up the predefined adjacency matrix obtained from the a priori information, which represents the connectivity of roads in a real road network. Its adjacency matrix $A^{\text{pre}} \in \mathbb{R}^{N\times N}$ is of the form

$$A_{\text{ij}}^{\text{pre}} = \begin{cases} 1, & \left(v_i, v_j \in e_{ij}\right), \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The adjacency matrix $A_{\text{ij}}^{\text{pre}} = 1$ when two nodes are connected and 0 when two nodes are not adjacent.

3.3.1.4. Multigraph Feature Structure. In order to combine emergency graph structure, stabilization graph structure, and road connection graph structure, we propose a multigraph fusion method in the form of multigraph: $G^* = \{G_1, G_2, G_3\}$. $G_1 = (V, E^{\text{Mi}}, A^{\text{Mi}})$ represents emergency graph structure, $G_2 = (V, E^{\text{adp}}, A^{\text{adp}})$ is stabilization graph structure, and $G_3 = (V, E^{\text{pre}}, A^{\text{pre}})$ represents by road connection graph structure. Multigraph feature structure learning is to better capture the data features in both static and dynamic space.

3.3.2. Spatial Relationship Learning

3.3.2.1. A GAT That Expresses Edge Characteristics. Traditional GATs focus on nodes that are only adjacent to the target node through the adjacency matrix and assign different weights to different nodes, while ignoring the effect of edge characteristics between two neighboring nodes in the graph structure on the correlation of that neighboring node. Our research is to design a GAT model that can express edge features and add the distance feature between neighboring sensors, as an important factor for GAT target nodes to pay attention to neighboring nodes, into the bias term of the self-attention mechanism calculation formula, which solves the above problem.

Edge features representing the distance between neighboring nodes are important for graph representation, and encoding them into the network along with node features is essential. In the previous work, there are two approaches to encode edge features. In the first approach, edge features are added to the associated node features [33, 34]; in the second approach, for each node, its associated edge features will be used together with the node features in the aggregation [22, 35, 36]. However, this method of edge feature encoding, which only propagates edge feature information to its associated nodes, may not be an efficient way to utilize edge information in the entire graph representation.

In order to better encode edge features into the attention layer, we propose a new way of encoding edge features. When the attention mechanism in traditional GAT calculates the correlation of neighboring node pairs (i, j), the feature values i and j in the two nodes are linearly transformed by the learnable weight matrix W, and then, the inner product is calculated to obtain the attention coefficients. However, we believe that the edge features between the two nodes should be considered in the calculation of the attention coefficient, so the edge features between the two nodes are added to the self-attention formula as a bias term when the inner product of the node features after the linear transformation is calculated. The longer the distance between two nodes, the weaker the correlation, so when encoding the distance feature, we subject its inverse to a learnable linear transformation to obtain this bias term c_{ij}. However, in the data preprocessing stage, we use Z score to normalize the input node feature data X. Therefore, when we do the node inner product calculation, the inner product value will be close to 1. If the bias term c_{ij} is either large or small, it will make the model difficult to optimize; therefore, we divide the average value of the distance X_{mean} by the distance X_{distance} between the two neighboring nodes as the distance feature, so that the value is both close to 1 and inversely proportional to the distance. This distance feature is then subjected to a learnable linear transformation to obtain bias term c_{ij}. The improved attention formula contains two contents: (a) the dot product attention value e_{ij} of the node pair (v_i, v_j) and (b) the bias value c_{ij} of the edge features.

The following is a specific implementation of the new GAT we have designed:

As is shown in Figure 5, we input the feature $X = \{x_1, x_2, \ldots, x_N\} \in \mathbb{R}^{N\times D}$ of a node at a given point in time and then transform that feature through a learnable linear transformation that converts it to get more advanced features. $W \in \mathbb{R}^{D'\times D}$ is a trainable random parameter matrix, and then, the self-attention mechanism is applied on the node—a shared attentional mechanism a (): $\mathbb{R}^{D'} \times \mathbb{R}^{D'} \longrightarrow \mathbb{R}$, as shown in equation:

$$e_{ij} = a\left(\text{Wx}_i, \text{Wx}_j\right), \quad j \in \mathcal{N}_{\text{i}}, \tag{14}$$

where e_{ij} denotes the dot product attention value of node pair (v_i, v_j). In our study, only the neighborhood Node j of Node i is of interest, so we compute only node $j \in \mathcal{N}_i$ by performing masked attention; $\mathcal{N}_{\text{i}}$ denotes the neighborhood of node i.

We believe that the edge features between them should be considered in this attention coefficient calculation, so we add the edge features as a bias term to this attention calculation as shown in equation as follows:

$$A_{ij} = e_{ij} + c_{ij} = a\left(\text{Wx}_i, \text{Wx}_j\right) + c_{ij}, \text{ where } c_{ij} = \frac{1}{\text{N}} \sum_{n=1}^{N} W_{eij} \cdot \left(\frac{X_{\text{mean}}}{X_{\text{distance}}}\right), \tag{15}$$

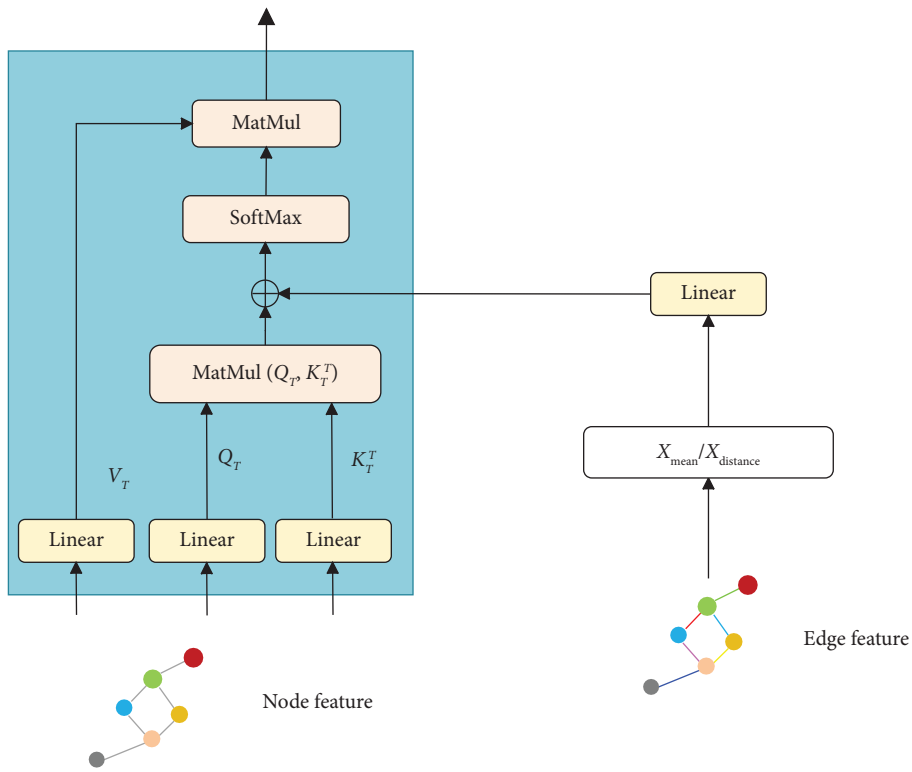

Figure 5: A GAT that expresses edge characteristics.

where $X_{distance}$ is the distance of an edge between two neighboring nodes and X_{mean} is the average of the distances. We make the distance feature undergo a learnable linear transformation, $W_{eij} \in \mathbb{R}^{E\times 1}$ is the learnable distance weight matrix, and E denotes the dimension of the edge attribute. In the formula, N denotes the shortest path $SP_{ij} = (e_1, e_2, \ldots, e_N)$ between Node i to Node j. In the generalized attention formula, the correlation between Nodes i and j decreases as the number of paths increases. But in our graph attention layer, the value of N is taken as 1 because only the first-order domain nodes of the target node are concerned.

Normalization using the SoftMax function is as follows:

$$\alpha_{ij} = \mathrm{SoftMax}\left(A_{ij}\right) = \frac{\exp\left(A_{ij}\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(A_{ij}\right)}. \tag{16}$$

Finally, the updated result can be obtained by passing a linear combination of the adjacency matrix features through a nonlinear activation function:

$$x_i' = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}Wxj\right), \tag{17}$$

where σ is the nonlinear activation function ReLU and $x_i' \in \mathbb{R}^D$ denotes the final output of Node i.

3.3.2.2. Graph Attention Learning for Multigraph Feature Structures. This is because the GAT model we designed to represent edge features only considers the distance between nodes that are connected by real roads, while the emergency graph structure and the stabilization graph structure in the multigraph feature structure are both used to represent hidden correlations between nodes that are not connected in the real world. Therefore, we use different GAT models to extract different graph structures in the multigraph feature structure: The new GAT model is used to extract the road connection graph structure, and the traditional GAT model is used to extract the spatial correlation in the emergency graph structure and the stabilization graph structure, respectively. Finally, we design a gated fusion mechanism to aggregate the spatial features captured by two different GATs. The new GAT model captures the actual spatial feature ACT(X), and the traditional GAT model captures the hidden spatial feature HID(X); we multiply ACT(X) and HID(X), respectively, by a learnable weight matrix and obtain the tensor between 0 and 1 as the gating value z after the sigmoid function processing, which can be used as the weights of actual spatial feature ACT(X) and hidden spatial feature HID(X) in the computation of final spatial feature formula. The specific formula is shown below.

The gating mechanism we designed is specifically shown in Figure 6. The graphs $G_1 = (V, E^{Mi}, A^{Mi}), G_2 = (V, E^{adp}, A^{adp}), G_3 = (V, E^{pre}, A^{pre})$ represent the emergency graph structure, stabilization graph structure, and road connection graph structure, respectively. This contains node features (traffic flow data), edge features (distance features), and an adjacency matrix. Then, the G_1, G_2 data are added to the traditional GAT model and the G_3 data are added to the new GAT model that we have improved to extract the spatial correlations in different feature graph structures, respectively. It can be expressed by the formula as follows:

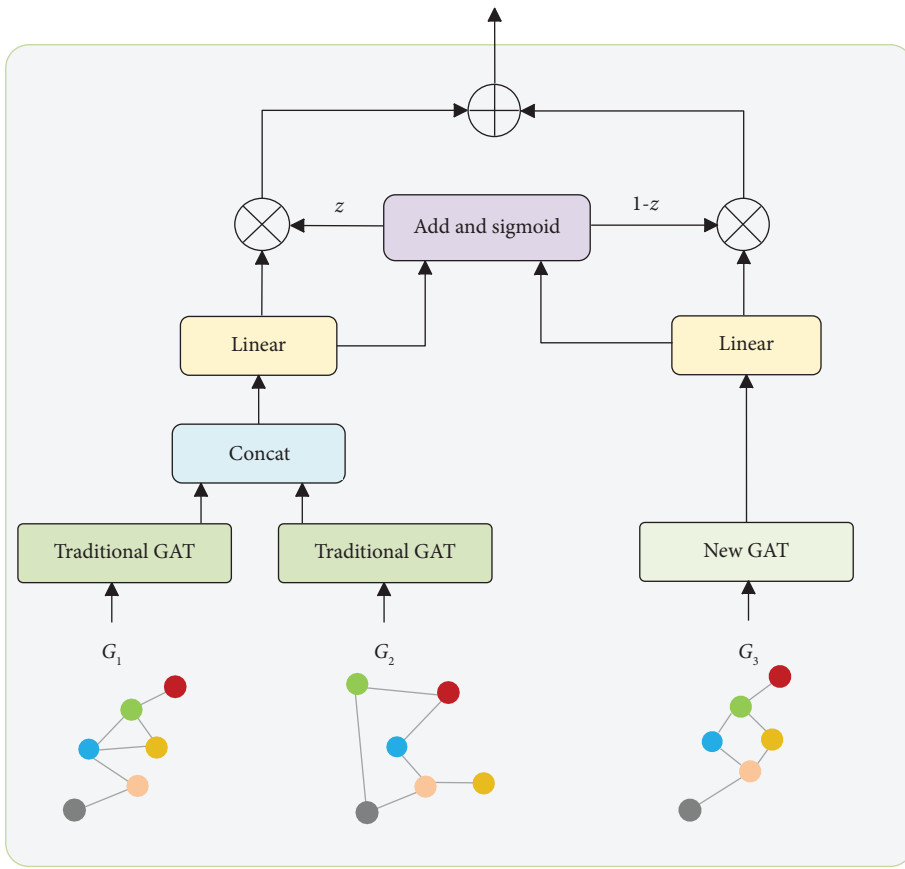

Figure 6: Gated fusion GAT.

$$\begin{aligned} \mathrm{HID}(X) &= \mathrm{Concat}\left(\mathrm{GAT}(G_1, X), \mathrm{GAT}(G_2, X)\right), \\ \mathrm{ACT}(X) &= \mathrm{NewGAT}(G_3, X), \end{aligned} \tag{18}$$

where $\mathrm{HID}(X) \in \mathbb{R}^{T\times N\times D}$ is the hidden space feature and $\mathrm{ACT}(X) \in \mathbb{R}^{T\times N\times D}$ is the actual space feature.

We design the gated fusion mechanism to consider both real and hidden space features and obtain the tensor located between 0 and 1 by the sigmoid function in this mechanism. Specifically, the following method is used to realize the gated fusion.

First, the gating value z is learned by the following equation:

$$\mathrm{z} = \mathrm{sigmoid}\left(\mathrm{ACT}(\mathrm{X})\mathrm{W}_{\mathrm{act}} + \mathrm{HID}(\mathrm{X})\mathrm{W}_{\mathrm{hid}} + \mathrm{b}\right), \tag{19}$$

where $\mathrm{W}_{\mathrm{act}} \in \mathbb{R}^{D\times D}$, $\mathrm{W}_{\mathrm{hid}} \in \mathbb{R}^{D\times D}$, $b \in \mathbb{R}^{D}$, denotes the parameters that can be learned in the linear mapping layer. The spatial output obtained after feature aggregation is defined as follows:

$$\mathrm{GFS}(X) = z \odot \mathrm{ACT}(X) + (1 - \mathrm{z}) \odot \mathrm{HID}(X), \tag{20}$$

where $\odot$ is the element product and $\mathrm{GFS}(X) \in \mathbb{R}^{T\times N\times D}$ is the fused graph space feature.

3.3.3. AGATs Module Learning Processes. In the previous two sections, we detailed multigraph feature structure learning and GANs for spatial feature learning; in this section, we illustrate in general how the AGAT module captures complex spatial correlations in traffic flow forecasting species.

First, we obtain, from the preprocessed data, $\mathcal{X} = \{X_{t-T+1}, X_{t-T+2}, X_{t-T+3}, \ldots, X_t\} \in \mathbb{R}^{T\times N\times H}$, denoting the H-dimensional traffic flow data for given N nodes over a history of T time steps. The input data are then learned through the multigraph feature structure introduced in the first section to obtain $G^* = \{G_1, G_2, G_3\}$, with $G_1 = (V, E^{\mathrm{Mi}}, A^{\mathrm{Mi}})$ representing the emergency graph structure, $G_2 = (V, E^{\mathrm{adp}}, A^{\mathrm{adp}})$ the stabilization graph structure, and $G_3 = (V, E^{\mathrm{pre}}, A^{\mathrm{pre}})$ the by road connection graph structure. We then use the gated fusion GAT module in Section 2 to aggregate the spatial features of the multigraph feature structure G^* and obtain the output $\mathrm{GFS}(X) \in \mathbb{R}^{T\times N\times D}$, which represents the fused graph space features.

Finally, as shown in the AGAT module in Figure 7, we transform the graph space features as the final output $S_{\mathrm{out}} \in \mathbb{R}^{T\times N\times D}$ of the AGAT module by layer normalization and feed forward, as denoted by the following equation:

$$S_{\mathrm{out1}} = \mathrm{Layer\ norm}\left(\mathrm{GFS}(X) + X\right), \tag{21}$$

$$S_{\mathrm{out}} = \mathrm{Layer\ norm}\left(\mathrm{ReLU}\left(S_{\mathrm{out1}}W_1^S\right)W_2^S + S_{\mathrm{out1}}\right), \tag{22}$$

where W_1^S and W_2^S are learnable parameters.

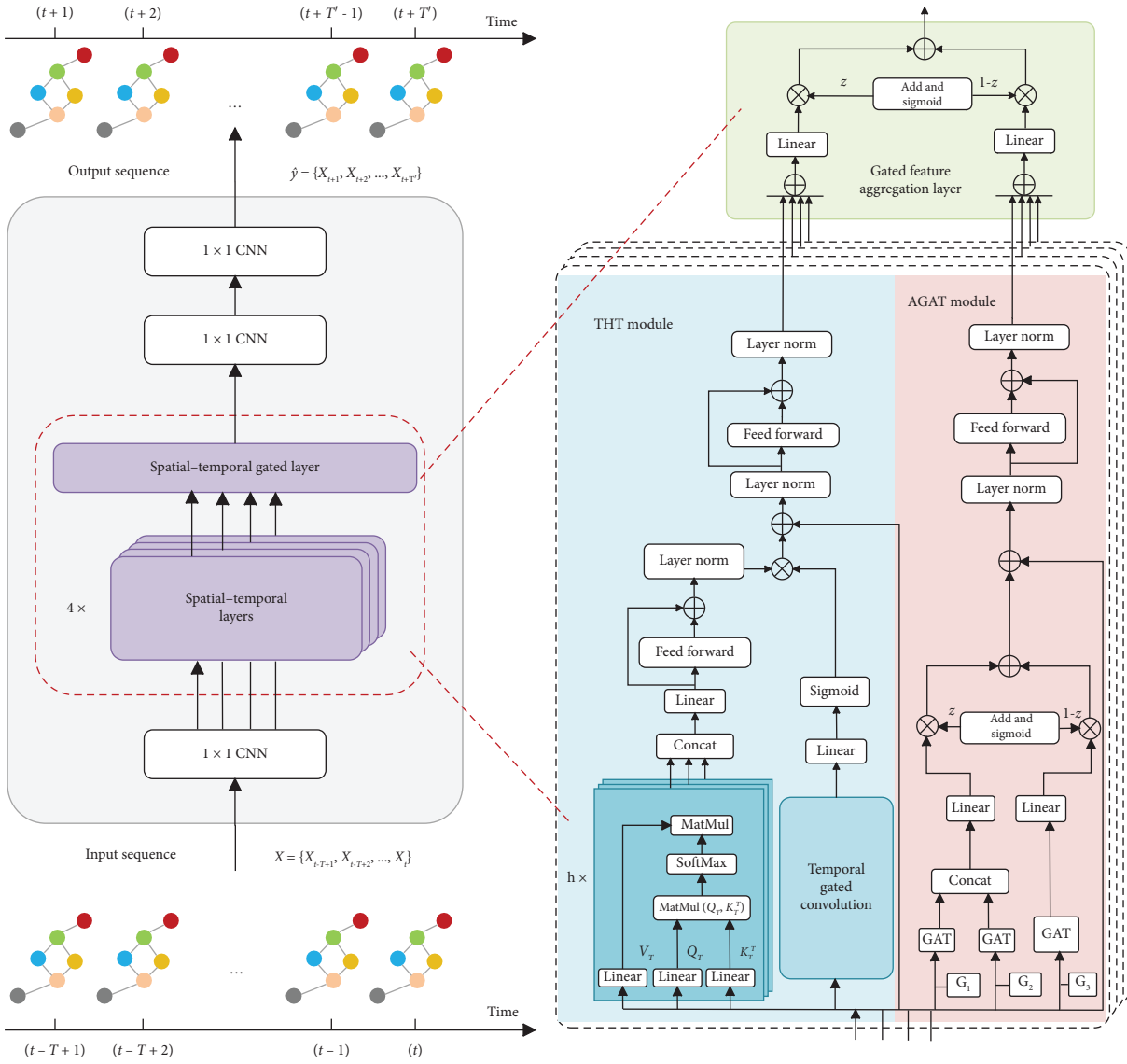

Figure 7: OGIF-GAT model; the left figure shows the overall structure of the model, and the right figure explains some of the details in the model.

3.4. THT. On a roadway, traffic conditions are influenced by historical traffic characteristics, which is the time series analysis of traffic flow forecasting. Traditional RNN models are good at capturing long-term macroscopic dependencies, but RNNs suffer from time-consuming iterations, ineffective parallel processing, and unstable gradient descent. As shown in Figure 7, THT module, our model consists of temporal transformer, TGC, and temporal fusion block. TGC uses 1D dilation causal convolution [37] and a gating mechanism [38] to capture the short-term characteristics of traffic conditions, by adding a factor to the standard causal convolution to control the jump distance. The field of view of causal convolution grows exponentially with increasing layer depth, so the TGC can handle longer sequences with fewer layers, thus reducing computational resource usage. At the same time, the nonrecursive nature supports parallel processing, reducing problems such as time consumption. Temporal transformer leverages a multihead self-attention mechanism to capture long-term features of traffic conditions, unlike RNN and its variants, and is more suitable for use in long sequences to capture long-term dependencies.

The THT module uses TGC to uncover short-term temporal dependencies, the time transformer to uncover long-term temporal dependencies, and the temporal fusion block fuses short-term dependence with long-term dependence.

In TGC, the 1D dilated causal convolution is implemented as follows, the input time series $X = \{x_1, x_2, \ldots, x_T\} \in \mathbb{R}^T$ and a filter $f = \{0, \ldots, k-1\} \in \mathbb{R}$, and the dilated causal convolution operation F at time step t is calculated as follows:

$$F(t) = (X *_d f)(t) = \sum_{i=0}^{k-1} f(i) \cdot X_{t-d \cdot i}. \tag{23}$$

The gating mechanism is given the traffic flow forecasting input $X \in \mathbb{R}^{T \times N \times D}$, which is subjected to a TGC operation of the following form:

$$\mathrm{TGC}(X) = \mathrm{sigmoid}(\Phi_1 * X + a) \odot \mathrm{Tanh}(\Phi_2 * X + b). \tag{24}$$

The temporal transformer module efficiently focuses on the correlation of traffic features in global time using the multihead self-attention mechanism. Input temporal feature $X \in \mathbb{R}^{T \times N \times D}$ into temporal transformer module to get output $\mathrm{TTM}(X) \in \mathbb{R}^{T \times N \times D}$.

Finally, in order to combine the consideration of long-term and short-term temporal dependence, we design a simple temporal fusion block, which transforms the output $\mathrm{TGC}(X) \in \mathbb{R}^{T \times N \times D}$ of the TGC module by linear transformation and sigmoid function and then the element-wise product with the output $\mathrm{TTM}(X) \in \mathbb{R}^{T \times N \times D}$ of the TTM module and finally transforms the temporal features by layer normalization and feed forward to obtain the final output $T_{\mathrm{out}} \in \mathbb{R}^{T \times N \times D}$ of the THT module, which is expressed as follows:

$$T_{\mathrm{out1}} = \mathrm{sigmoid}\left(\mathrm{TGC}(X) W_1^T\right) \odot \mathrm{TTM}(X), \tag{25}$$

$$T_{\mathrm{out2}} = \mathrm{Layer\,norm}\left(T_{\mathrm{out1}} + X\right), \tag{26}$$

$$T_{\mathrm{out}} = \mathrm{Layer\,norm}\left(\mathrm{ReLU}\left(T_{\mathrm{out2}} W_2^T\right) W_3^T + T_{\mathrm{out2}}\right), \tag{27}$$

where W_1^T, W_2^T, and W_3^T are the learnable weight parameters.

3.5. Gated Feature Aggregation Layer. Both spatial correlation and temporal dependence of traffic data need to be considered in traffic flow forecasting. Therefore, we believe that the spatial correlation of the road network and the temporal dependence of the traffic flow should be jointly modeled. The gated feature aggregation layer has been proposed for aggregating these two traffic characteristics.

Its structure is shown in Figure 7. Gated feature aggregation layer: We first learn gating z by the following equations:

$$z = \mathrm{sigmoid}\left(W_t \sum_{k=0}^{K} T_{\mathrm{out}}^{(k)} + W_s \sum_{k=0}^{K} S_{\mathrm{out}}^{(k)} + b\right), \tag{28}$$

where $W_t \in \mathbb{R}^{D \times D}$, $W_s \in \mathbb{R}^{D \times D}$, and $b \in \mathbb{R}^{D \times D}$ denote the parameters that can be learned in the linear mapping layer, $T_{\mathrm{out}}^{(k)}$ is the output of the THT module, and $S_{\mathrm{out}}^{(k)}$ is the output of the AGAT module, respectively, there are k layers, and k is defined in this project study as 4. The final output spatiotemporal features OUT of the spatial–temporal layers after the feature aggregation is defined as follows:

$$\mathrm{OUT} = z \otimes \sum_{k=0}^{K} T_{\mathrm{out}}^{(k)} + (1 - z) \otimes \sum_{k=0}^{K} S_{\mathrm{out}}^{(k)}. \tag{29}$$

3.6. Model Structures. The general structure of the OGIF--GAT model is shown in Figure 7. In order to accurately predict future traffic flow conditions, we develop spatial–temporal layers to jointly model the temporal and spatial dependencies of the traffic network. Before entering the spatial–temporal layers, we obtain the high-dimensional spatial input feature $\mathcal{X}' = \{X_{t-T+1}, X_{t-T+2}, X_{t-T+3}, \ldots, X_t\} \in \mathbb{R}^{T \times N \times D}$ from the historical observation $\mathcal{X} = \{X_{t-T+1}, X_{t-T+2}, X_{t-T+3}, \ldots, X_t\} \in \mathbb{R}^{T \times N \times H}$ by 1×1 CNN, and then, input the features into the 4-layer spatial–temporal layers to obtain the temporal feature $T_{\mathrm{out}}^{(4)} \in \mathbb{R}^{T \times N \times D}$ and the spatial feature $S_{\mathrm{out}}^{(4)} \in \mathbb{R}^{T \times N \times D}$. The gated feature aggregation layer is used to aggregate $T_{\mathrm{out}}^{(4)}, S_{\mathrm{out}}^{(4)}$ to get the spatiotemporal feature $\mathrm{OUT} \in \mathbb{R}^{T \times N \times D}$. In addition, we use a two-layer 1×1 CNN to complete the multistep prediction to get the final predicted value $\hat{y} \in \mathbb{R}^{\mathrm{T \times N \times H}}$. Finally, we compute the loss value L of the predicted value $\hat{y}$ with respect to the real value y by means of a loss function and update the trainable parameters by the gradient descent method.

Loss function: due to the traffic data collection process being prone to error, we use the Huber loss function, which is insensitive to outliers and more stable; the loss function is defined as follows:

$$L(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2, & |y - \hat{y}| \le \varpi, \\ \varpi |y - \hat{y}| - \frac{1}{2}\varpi^2, & |y - \hat{y}| > \varpi, \end{cases} \tag{30}$$

where y is the real traffic flow, $\hat{y}$ is the predicted traffic flow, and ϖ is the threshold.

To make this traffic flow prediction process easier to understand, we use Algorithm 1 to represent the training process of OGIF-GAT.

4. Experiment

4.1. Datasets. Our evaluation of OGIF-GAT is based on four publicly available transportation network datasets: PeMS04, PeMS08, PeMS-BAY, and METR-LA. Three of the datasets,

Input: traffic data $\mathcal{X} \in \mathbb{R}^{T\times N\times H}$, predefined graph structure G, number of layers of STLayers K, number of training epochs epochs;
Output: Multigraph feature structure G^*, learned OGIF-GAT;
1. Randomly initialize learnable parameters of OGIF-GAT;
2. $D_{\text{train}} \leftarrow \varnothing$;
3. Sliding window to construct training set from traffic data $D_{\text{train}} \leftarrow (\mathcal{X}_{\text{train}}, Y_{\text{train}})$;
4. for epoch = 0; epoch < epochs; epoch + + do;
5. Randomly select a batch $\{X, Y\}$ from D_{train};
6. Get high-dimensional features X' from features X using 1×1 CNN;
7. Get multigraph feature structure G^* from features X' and graph structure G using multigraph feature structure learning;
8. for $k = 0$; $k < K$; k++ do;
9. Get temporal features $T_{\text{out}}^{(k)}$ from features X' using THT;
10. Get spatial features $S_{\text{out}}^{(k)}$ from features X' and multigraph feature structure G^* using AGAT;
11. end for
12. Get spatiotemporal feature OUT from temporal features $T_{\text{out}}^{(k)}$ and spatial features $S_{\text{out}}^{(k)}$ using the gated feature aggregation layer;
13. Get predicted value $\widehat{y}$ from spatiotemporal feature OUT using two-layer 1×1 CNN;
14. Compute loss $L(Y, \widehat{y})$ using equation (30);
15. Update trainable parameters with gradient descent;
16. end for

Algorithm 1: Training process of OGIF-GAT.

PeMS04, PeMS08, and PeMS-BAY, were collected from the Caltrans' Performance Measurement System, while METR-LA was collected from the Los Angeles County Road Network Loop Detector. We choose datasets from different sources in order to better discuss the model's ability to generalize to different traffic environments. We describe these four datasets in detail.

PeMS04: This dataset records traffic flow data collected by 3848 sensors on 29 roads contained on 16,992 time steps from 1.1.2018 to 28.2.2018. These data were sampled at 5-min intervals, and the data characteristics include flow, occupancy, and speed, stored in .npz format. This includes three-dimensional data: The first dimension is the time step, the second dimension is the number of sensors, and the third dimension is the traffic characteristics. We remove some redundant sensors to ensure that the distance between neighboring sensors is more than 3.5 miles, yielding 307 sensors. For this experiment, we predicted the flow feature for this dataset, so we kept only the flow feature in the third dimension feature and got the final data dimension (16,992, 307, 1).

PeMS08: This dataset records traffic flow data collected by 1979 sensors on 8 roadways over 17,856 time steps from 2016.7.1 to 2016.8.31. Like the PeMS04 dataset, this dataset is sampled at 5-min intervals, and the data characteristics include flow, occupancy, and speed, stored in .npz format. We remove redundant sensors to ensure that neighboring sensors are more than 3.5 miles apart, resulting in 170 sensors. This experiment also preserves the flow features and yields the final data dimension (17,856, 170, 1).

PeMS-BAY: This dataset records traffic speed data from 52,116 time steps containing 325 sensor recordings from 2017.1.1 to 2017.5.31. Traffic information is recorded at 5-min intervals, and this data feature is the speed feature that records the average speed of vehicles (miles/hour), stored in .h5 format, with a data dimension of (52,116, 325, 1).

METR-LA: This dataset records traffic speed data collected from 2012.3.1 to 2012.6.30, 34,272 time steps containing 207 sensors. These data were sampled at 5-min intervals, and like the PeMS-BAY dataset, this data feature is a traffic speed feature that records the average speed of a vehicle (miles/hour), stored in .h5 format, with a data dimension of (34,272, 207, 1).

The above four datasets, PeMS04, PeMS08 using flow data and PeMS-BAY, METR-LA using speed data, were chosen in order to demonstrate that the model has advantages in predicting different traffic characteristics. These four datasets come from two different sources and were collected in different time periods, and in different geographical areas of the study, so the selection of these four different datasets helps to evaluate the generalization ability of the model.

We use the previous 12 time steps to forecast the following 12 time steps. The input data are normalized using the Z score method as described below:

$$\overrightarrow{X} = \frac{X - \text{mean}(X_{\text{train}})}{\text{std}(X_{\text{train}})}, \tag{31}$$

where $\text{mean}(X_{\text{train}})$, $\text{std}(X_{\text{train}})$ represent the mean and standard deviation of the training data, respectively. Next, we divided the dataset into training, testing, and validation sets in the ratio of 6:2:2.

4.2. Evaluation Metrics. This experiment uses three common metrics to measure the performance of different models:

$$\text{mean absolute error (MAE)} = \frac{1}{N \times T} \sum_{t=1}^{T} \sum_{n=1}^{N} \left| Y_t^n - \overrightarrow{Y_t^n} \right|, \tag{32}$$

$$\text{MAPE} = \frac{1}{N \times T} \sum_{t=1}^{T} \sum_{n=1}^{N} \frac{\left| Y_t^n - \overrightarrow{Y_t^n} \right|}{Y_t^n}, \tag{33}$$

$$\text{RMSE} = \sqrt{\frac{1}{N \times T} \sum_{t=1}^{T} \sum_{n=1}^{N} \left(Y_t^n - \overrightarrow{Y_t^n} \right)^2}, \tag{34}$$

where N is the total number of nodes, T is the total time step, and Y_t^n and $\overrightarrow{Y_t^n}$ represent the factual outcome of the nth node's sample at moment t and the outcome predicted by our model, respectively.

4.3. Experimental Settings. The experiment is to use the traffic characteristics in the last hour to predict the traffic characteristics in the coming hour, so the input and output time step is $t = 12$. The number of traffic features on the dataset $H = 1$, which represents traffic flow in PeMS04 and PeMS08 datasets and traffic speed in PeMS-BAY and METR-LA datasets. The number of high-dimensional spatial features was set to $D = 32$. We use Adam optimizer to train our model. The batch size is 16, and the learning rate is 0.001. The temporal transformer module in MSA has header $h = 4$. Spatial–temporal layers have $k = 4$ layers.

4.4. Baseline Methods. We compared the OGIF-GAT model to eight cutting-edge baseline models:

1. GAT: GATs perform attention operations on the graph structure to focus on the neighborhood node characteristics and get the weights of the influential nodes.
2. LSTM: Long short-term memory networks handle temporal tasks through memory cells and gating mechanisms.
3. TCN [39]: TCN is time series prediction model consisting of stacked CNNs.
4. STGCN: Spatiotemporal graph convolutional network consists of graph convolution and gated causal convolution to obtain spatial correlation and temporal correlation, respectively.
5. ASTGCN: Attention based spatial–temporal graph convolutional networks capture dynamic spatiotemporal correlations by a spatiotemporal attention mechanism, which employs graph convolution to capture spatial features and generalized standard convolution to capture temporal features.
6. STSGCN: Spatial–temporal synchronous graph convolutional networks design spatiotemporal synchronous modeling mechanisms to capture local spatiotemporal correlations.
7. AdpSTGCN [13]: Adaptive spatial–temporal graph convolutional network utilizes a multihead attention mechanism to construct multigraph features and introduces an adaptive graph convolutional approach to capture complex spatial correlations, in addition to designing a cascading structural framework for spatiotemporal dynamics modeling.
8. DSTAGNN [40]: Dynamic spatial–temporal aware graph neural network proposes a data-driven dynamic spatiotemporal aware graph based on data and designs a new GNN architecture with an improved multihead attention mechanism and adds multiscale gated convolutions to obtain a wide range of dynamic time dependency.

4.5. Experimental Results. Performance comparisons: Table 1 represents the comparison of the performance of different deep learning methods on four different datasets for predicting results at 15 min, 30 min, and 1 hour in the future. Our model OGIF-GAT obtains relatively good prediction results in both short-term and long-term prediction on both datasets. Among them, GAT only considers spatial correlation and ignores temporal correlation, so the overall prediction ability is poor, but the prediction performance is little affected by the prediction time step, while LSTM and TCN only consider temporal correlation lacks and ignore spatial correlation, and their prediction of the short-term temporal performance is not bad but is not good at making long-term time predictions. STGCN, ASTGCN, and STSGCN consider temporal and spatial correlation to improve the prediction performance; however, they only focus on the predefined static graph structure in the space, AdpSTGCN, DSTAGNN, and our OGIF-GAT model all propose data-driven multigraph feature structure based on the data, and thus, they all achieve a high level in the prediction performance.

Our model OGIF-GAT outperforms all baseline methods on each time forecasting on both datasets because the model incorporates a multigraph structure learning module on the spatial side, which is able to learn an optimal graph structure from static and dynamic structures, respectively. The spatial correlation information contained in the map structure is more accurate and abundant than that contained in the traditional road connection map structure. On the temporal side, the model extracts and incorporates the short-term and

Table 1: Performance comparison of different methods on four different datasets.

Datasets	Methods	15 min			30 min			60 min		
		MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
PeMS04	GAT	23.54	37.40	15.35	24.63	38.82	15.83	27.82	42.94	18.11
	LSTM	21.99	34.33	14.40	25.30	39.37	16.52	32.90	50.55	21.86
	TCN	21.74	33.97	14.28	25.05	38.77	16.46	33.05	50.07	22.15
	STGCN	23.13	36.41	15.59	25.08	39.04	16.60	30.39	46.00	20.52
	ASTGCN	19.60	30.56	13.44	21.34	33.53	14.85	25.38	39.12	17.71
	STSGCN	19.80	31.58	13.41	21.31	33.84	14.27	24.47	38.46	16.27
	AdpSTGCN	18.21	29.12	13.16	19.85	31.11	14.53	22.02	34.91	17.84
	DSTAGNN	18.45	29.84	12.13	19.32	31.60	12.63	21.07	34.60	13.76
	OGIF-GAT	**18.89**	**30.42**	**12.24**	**19.86**	**31.98**	**12.85**	**21.51**	**34.43**	**14.01**
PeMS08	GAT	20.67	32.66	17.78	21.30	33.36	18.00	24.06	37.32	20.27
	LSTM	17.59	27.47	11.76	20.25	32.06	13.08	26.51	41.10	18.51
	TCN	17.19	26.67	11.68	19.67	30.77	13.36	25.79	39.41	17.86
	STGCN	20.22	31.71	14.67	21.32	33.47	15.39	24.87	38.21	18.48
	ASTGCN	16.56	25.67	10.46	18.70	28.83	11.49	22.96	34.62	13.77
	STSGCN	16.60	25.37	10.98	17.75	27.28	11.56	20.12	30.64	13.04
	AdpSTGCN	14.71	23.02	10.51	15.91	25.75	10.78	18.83	29.86	11.56
	DSTAGNN	14.81	23.20	9.66	15.72	24.89	10.21	17.61	**27.80**	11.20
	OGIF-GAT	**15.08**	**23.98**	**10.05**	**16.31**	**26.11**	**10.61**	**18.04**	**28.61**	**11.35**
PeMS-BAY	GAT	4.04	7.63	11.16	4.16	7.95	11.64	4.40	8.68	12.58
	LSTM	1.32	2.67	2.74	1.71	3.64	3.75	2.14	4.54	4.85
	TCN	1.49	3.22	3.23	2.03	4.68	4.71	2.86	6.57	7.18
	STGCN	2.14	4.32	4.68	2.39	4.94	5.43	2.83	5.99	6.84
	ASTGCN	1.49	3.08	3.00	1.95	4.27	5.21	2.52	5.67	6.23
	STSGCN	2.78	8.74	6.32	2.21	5.17	4.66	2.66	6.58	5.74
	AdpSTGCN	1.33	2.82	2.76	1.66	3.74	3.73	1.97	4.51	4.67
	DSTAGNN	1.42	3.06	3.09	1.80	4.10	4.10	2.16	4.95	5.12
	OGIF-GAT	**1.32**	**2.88**	**25.6**	**1.65**	**3.57**	**3.57**	**1.932**	**4.44**	**4.60**
METR-LA	GAT	5.75	10.69	21.13	5.92	10.88	21.50	6.32	11.33	22.44
	LSTM	3.30	7.59	8.06	3.91	8.94	10.07	4.82	10.84	12.75
	TCN	3.15	7.01	8.01	3.94	8.98	10.25	5.18	11.37	13.81
	STGCN	3.71	8.00	9.43	4.21	9.27	10.97	5.03	10.86	13.87
	ASTGCN	2.85	5.51	8.22	3.28	6.62	9.23	3.86	7.87	12.56
	STSGCN	3.35	7.41	8.10	3.84	8.58	9.78	4.62	10.12	12.44
	AdpSTGCN	2.62	5.14	6.69	3.00	6.16	8.01	3.47	7.26	9.62
	DSTAGNN	3.96	9.77	8.66	4.95	12.25	10.45	6.32	15.00	13.00
	OGIF-GAT	**2.67**	**5.84**	**6.53**	**3.14**	**7.25**	**8.06**	**3.52**	**8.13**	**9.13**

Note: The bold values represent the optimal performance.

long-term time dependencies separately, so that our model has a more significant improvement in both short-term time forecasting and long-term time forecasting. AdpSTGCN and DSTAGNN models are the state-of-the-art methods within the last 2 years, our model is slightly lower than the DSTAGNN model on some metrics data in the first three datasets, but similar, but in the METR-LA dataset, the performance of the DSTAGNN model suddenly becomes poor, which indicates that our model has a better generalization to different datasets and is capable of traffic flow prediction on most datasets. The AdpSTGCN model is a little better than our model on 15-min prediction, while the performance is not as good as our model on 60-min prediction, indicating that our model is more suitable for longer traffic flow prediction; this is because although AdpSTGCN obtains local dynamic spatial features through the digit-driven method, it ignores the overall static spatial hidden features, while the static features do not change over time and are more suitable for long-term traffic flow prediction.

Figure 8 shows in detail the predictive performance of the model at 12 time steps, and in general, as the time step gets longer, the model performance weakens. In short-term forecasting, LSTM and TCN both achieve better prediction results, while GAT does not perform well enough, but as the time step becomes longer, the prediction performance of LSTM as well as TCN, which are the two models that only consider temporal correlation, degrade the fastest. ASTGCN, STSGCN, and our model OGIF-GAT all perform better in overall prediction, but our model shows the best overall performance.

4.6. Ablation Study. In order to verify the validity of the multigraph feature fusion structure, we designed ablation experiments to verify that each graph structure part (A^{Mi}, A^{adp}, A^{pre}) in the multigraph structure is valid. We designed four combinations of graph features: 1. using only the predefined road connection graph structure A^{pre}; 2. adding an emergency graph structure A^{Mi} to the predefined road connection graph structure A^{pre}; 3. adding a stabilization graph structure A^{adp} to the predefined road connection graph structure A^{pre}; and 4. adding both the emergency graph structure A^{Mi} and the stabilization graph structure A^{adp} to the predefined road connection graph structure A^{adp}. It was used to compare the experiments on PeMS04, PeMS08, PeMS-BAY, and METR-LA, and the results are shown in Table 2. The experimental results show that both the emergency graph structure and stabilization graph structure can improve the accuracy of traffic flow forecasting, while the simultaneous fusion of the emergency graph structure and the stabilization graph structure can lead to the best model performance. This suggests that both bursty and steady-state hidden graph structures can introduce new and useful information to the predefined real road connection scenarios, thus improving the model's proficiency to capture spatial relationships.

In order to verify the effectiveness of the improved GAN, we design ablation experiments: We change all the improved GANs in the model to the traditional GAN and conduct comparison experiments on four datasets. The following prediction results are finally obtained, as shown in Table 3. The experiment proves that the improved GAN, focusing on its edge features and influencing the nodes to focus on the weights of neighboring nodes at different distances, improves the prediction performance of the model.

4.7. Parametric Analysis. The design of different hyperparameters affects the final prediction performance of this model, and we design the following experiments to explore how the number of STLs k and the number of attention heads h in the time-converter module affect the model performance. (1) The number of STLs k, different STLs extract different aspects of spatiotemporal properties in the traffic data, fewer STLs will lead to insufficiently rich spatiotemporal properties being extracted, and more STLs can capture more complex spatiotemporal patterns; however, too many STLs will waste more computational resources and overfitting to the training data, so the appropriate number of STLs k is one of the parameters necessary for the better performance of the model, and we choose $k = [2, 4, 8]$ to analyze the performance of the model. (2) The number of attentional heads h in the temporal converter module, like the number of STLs, and more attentional heads can capture more complex temporal dependencies, so we choose $h = [2, 4, 8]$ for performance analysis. Experiments are carried out on four separate datasets, with the results illustrated in Figure 9. The optimal performance of our model is observed when the hyperparameters k and h are both set to 4.

4.8. Peak Hour Performance Analysis. The model faces different challenges at each time of the day, and during the morning and evening peak hours, a more diverse set of factors affecting traffic flow, such as different combinations of work-related, school-related, and leisure-related, can make it more difficult to predict their traffic flow conditions. We, therefore, analyze the model performance for the morning and evening peak hours. From the traffic dataset, we find that the morning and evening peak hours are specifically 7:00–9:00 and 16:00–18:00 on weekdays, while nonweekdays show a single peak. Finally, the peak performance MAE, RMSE, and MAPE values in different datasets are obtained as shown in Table 4.

We found that the evening peak usually has higher errors in MAE, RMSE, and MAPE, indicating that it is more difficult to predict, and data on specific factors affecting traffic in the evening peak hour, such as the exact time of school dismissal, and the distribution of off-duty hours in different industries, can be incorporated into the model, which can improve its prediction accuracy.

It is meaningful to predict the traffic flow data of morning and evening peaks, and it is more necessary to relieve traffic congestion for working staff and school

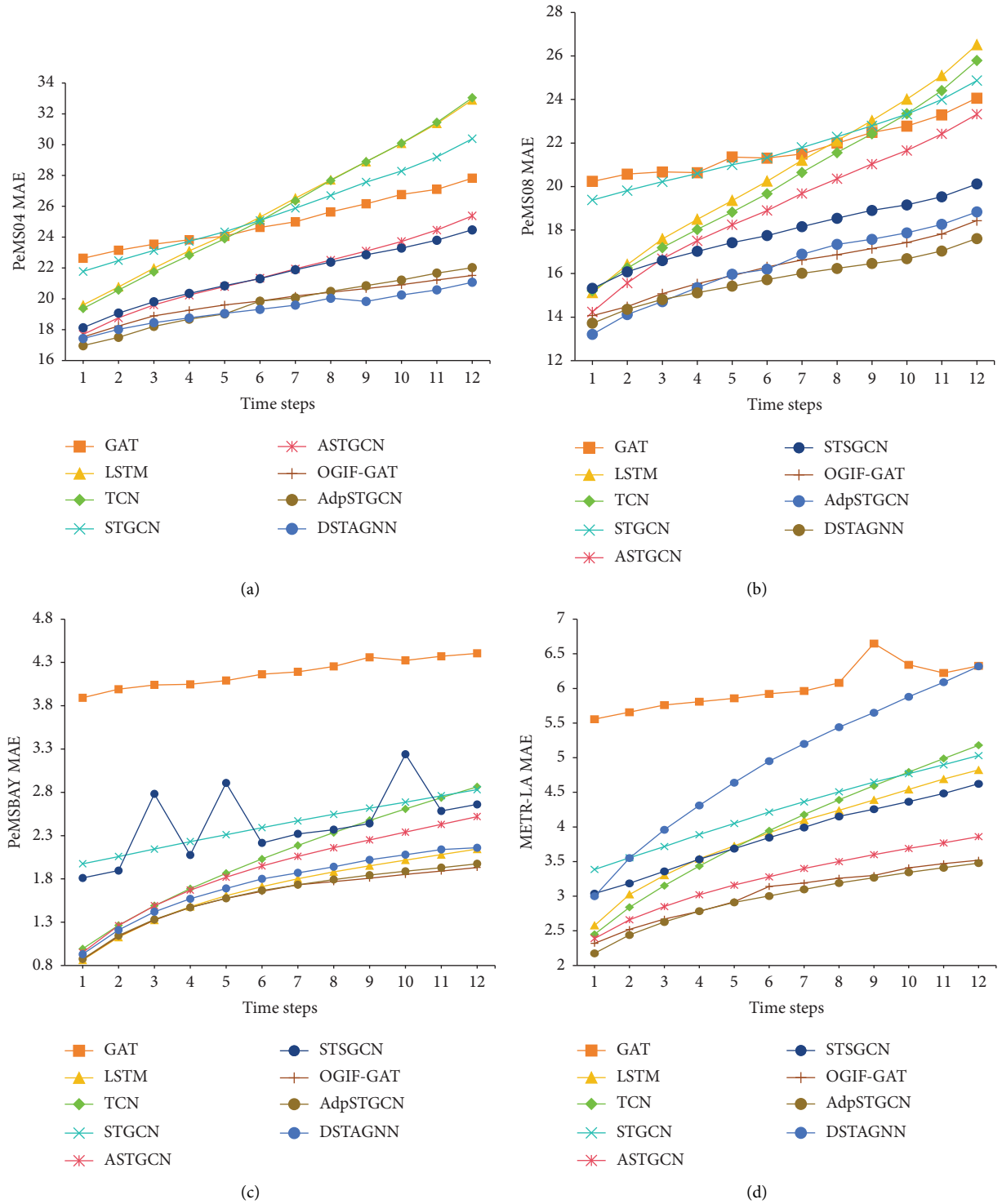

Figure 8: Model prediction results at different time steps. (a) Prediction results of different methods for PeMS04. (b) Prediction results of different methods for PeMS08. (c) Prediction results of different methods for PeMS-BAY. (d) Prediction results of different methods for METR-LA.

students. ITS can manage the traffic according to the predicted morning and evening peak traffic flow data to ease road congestion and bring convenience for people to travel.

4.9. Visualization. To get a more intuitive feel for data in real-time forecastings of the different models, Figure 10 shows the flow rate at 288 points in time collected by Sensor

Table 2: Comparative experimental results of four graph feature combinations on PeMS04 and PeMS08.

Dataset	Adjacency matrix	MAE	RMSE	MAPE (%)
PeMS04	$[A^{pre}]$	20.23	32.36	13.17
	$[A^{pre}, A^{Mi}]$	19.94	32.19	12.89
	$[A^{pre}, A^{adp}]$	19.86	32.02	13.14
	$[A^{pre}, A^{Mi}, A^{adp}]$	19.81	31.80	12.88
PeMS08	$[A^{pre}]$	17.25	27.23	11.76
	$[A^{pre}, A^{Mi}]$	16.18	25.83	11.42
	$[A^{pre}, A^{adp}]$	16.31	26.03	11.45
	$[A^{pre}, A^{Mi}, A^{adp}]$	16.04	25.72	11.39
PeMS-BAY	$[A^{pre}]$	1.81	3.95	3.88
	$[A^{pre}, A^{Mi}]$	1.74	3.86	3.80
	$[A^{pre}, A^{adp}]$	1.71	3.81	3.78
	$[A^{pre}, A^{Mi}, A^{adp}]$	1.63	3.75	3.72
METR-LA	$[A^{pre}]$	3.52	7.67	8.38
	$[A^{pre}, A^{Mi}]$	3.36	7.53	8.32
	$[A^{pre}, A^{adp}]$	3.47	7.54	8.25
	$[A^{pre}, A^{Mi}, A^{adp}]$	3.11	7.38	8.11

Table 3: Comparison of experimental results of the model using traditional graph attention networks on four datasets.

Dataset	MAE	RMSE	MAPE (%)
PeMS04	19.89	31.88	12.94
PeMS08	16.24	25.82	11.45
PeMS-BAY	1.67	3.80	3.78
METR-LA	3.29	7.56	8.14

Figure 9: Effect of the number of spatiotemporal layers k and the number of attention heads h on model performance.

Table 4: Comparison of model performance on four datasets during morning and evening peak hours.

Dataset	MAE		RMSE		MAPE (%)	
Time	Morning peak	Evening peak	Morning peak	Evening peak	Morning peak	Evening peak
PeMS04	19.85	19.88	31.82	31.86	12.90	12.98
PeMS08	16.14	16.25	25.75	25.82	11.41	11.49
PeMSBAY	1.67	1.68	3.76	3.82	3.74	3.77
METR-LA	3.17	3.25	7.38	7.41	8.10	8.16

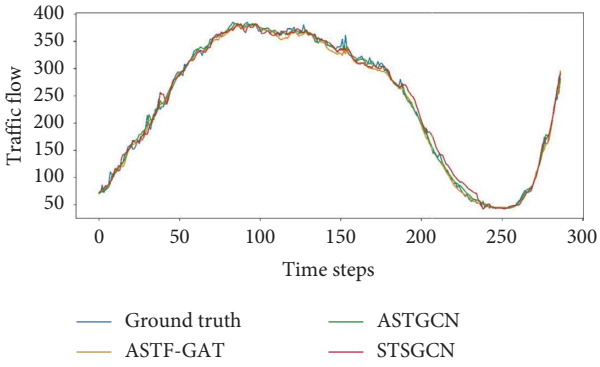

Figure 10: Visualization results of real-time data comparison of different models on PeMS08.

55 every five minutes from 0:00 to 24:00 and the results of the different models that implement predictions for that flow rate. We can find that although both can predict the real-time data very accurately, our model performs smoothly. It shows that our model is not overfitted and is little influenced by noise within the training data, which strongly demonstrates that our proposed model can handle complex traffic situations with different traffic characteristics, number of roads, and missing value ratio.

5. Conclusions and Outlook

In this study, we introduce an OGIF-GAT. In order to overcome the limitations of existing approaches that cannot accurately extract the spatial relationship on the traffic map, OGIF-GAT proposed a multigraph feature fusion structure that can statically and dynamically capture the road-space relationship under stable conditions and the road-space relationship under emergency conditions and learn an optimal graph adjacency matrix. OGIF-GAT improves the traditional GAN model to be more accurate and applicable to multigraph feature fusion structures to capture spatial correlations in traffic flow forecasting. To validate the proposed model, we conducted numerous comparative experiments across four independent datasets, and the findings confirmed that our method outperforms current techniques.

Our model does not take into account the cyclical nature of traffic flow variations, such as long and stable cyclical variations such as a day, a week, and so on. In addition, if traffic flow forecasting is applied to ITS on a large scale, people's traffic trips or driverless will change their trips and different roads according to the predicted results, and smart cities will control traffic light time in advance according to the predicted results in order to cope with the traffic congestion, so the traffic flow forecasting model should also take these factors into account for more accurate and real-time changes in traffic flow forecasting.

6. Discussion

Accurate real-time traffic flow prediction can help the intelligent traffic management system to carry out reasonable traffic control on roads at different times, and the navigation system can formulate the optimal route choice based on the optimization algorithm according to the results of this study, which can help the driver avoid congested road sections and drive more quickly and safely. With the development of autonomous driving technology, researchers should consider using traffic flow prediction to control future urban traffic flow. Peng et al. [41] concluded that connected and autonomous vehicles (CAVs) will coexist with human-driven vehicles (HDVs) for an extended period, so they developed a hybrid control framework that integrates a platoon control strategy based on the "catch-up" mechanism with lane management for CAVs. The results of this study resulted in a significant improvement in roadway capacity. In the future, intelligent transportation is one of the important components of the smart city, including road traffic management, public transportation system management, self-driving cars, and travel information system; real-time traffic flow information is the key to rational intelligent traffic management.

Data Availability Statement

The data that support the findings of this study are openly available in GitHub at https://github.com/Flc-1217/OGIF-GAT.

Conflicts of Interest

The authors declare no conflicts of interest.

Funding

The funding grant is for all authors of the paper. This work was supported by the National Key Research and Development Program of China (grant number 2019YFE0126100).

Acknowledgments

This work was supported by the National Key Research and Development Program of China (grant numbers 2019YFE0126100).

References

[1] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-Driven Intelligent Transportation Systems: A Survey," *IEEE Transactions on Intelligent Transportation Systems* 12, no. 4 (2011): 1624–1639, https://doi.org/10.1109/tits.2011.2158001.

[2] J. Tanimoto and X. An, "Improvement of Traffic Flux with Introduction of a New Lane-Change Protocol Supported by Intelligent Traffic System," *Chaos, Solitons & Fractals* 122 (2019): 1–5, https://doi.org/10.1016/j.chaos.2019.03.007.

[3] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic Flow Prediction With Big Data: A Deep Learning Approach," *IEEE Transactions on Intelligent Transportation Systems* 16, no. 2 (2014): 865–873.

[4] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "Spatiotemporal Recurrent Convolutional Networks for Traffic Prediction in Transportation Networks," *Sensors* 17, no. 7 (2017): 1501, https://doi.org/10.3390/s17071501.

[5] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long Short-Term Memory Neural Network for Traffic Speed Prediction Using Remote Microwave Sensor Data," *Transportation Research Part C: Emerging Technologies* 54 (2015): 187–197, https://doi.org/10.1016/j.trc.2015.03.014.

[6] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems* 28, no. 10 (2016): 2222–2232.

[7] R. Dey and F. M. Salem, "Gate-variants of Gated Recurrent Unit (GRU) Neural Networks," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)* (IEEE, 2017), 1597–1600.

[8] J. Liu, Y. Kang, H. Li, H. Wang, and X. Yang, "STGHTN: Spatial-Temporal Gated Hybrid Transformer Network for Traffic Flow Forecasting," *Applied Intelligence* 53, no. 10 (2023): 12472–12488, https://doi.org/10.1007/s10489-022-04122-x.

[9] B. Huang, H. Dou, Y. Luo, J. Li, J. Wang, and T. Zhou, "Adaptive Spatiotemporal Transformer Graph Network for Traffic Flow Forecasting by Iot Loop Detectors," *IEEE Internet of Things Journal* 10, no. 2 (2023): 1642–1653, https://doi.org/10.1109/jiot.2022.3209523.

[10] B. Yu, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," *arXiv preprint arXiv:1709* (2017): 04875.

[11] L. Zhao, Y. Song, C. Zhang, et al., "T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction," *IEEE Transactions on Intelligent Transportation Systems* 21, no. 9 (2020): 3848–3858, https://doi.org/10.1109/tits.2019.2935152.

[12] Z. Geng, J. Xu, R. Wu, et al., "STGAFormer: Spatial–Temporal Gated Attention Transformer Based Graph Neural Network for Traffic Flow Forecasting," *Information Fusion* 105 (2024): 102228, https://doi.org/10.1016/j.inffus.2024.102228.

[13] X. zhang, X. Chen, H. Tang, Y. Wu, H. Shen, and J. Li, "AdpSTGCN: Adaptive Spatial-Temporal Graph Convolutional Network for Traffic Forecasting," *Knowledge-Based Systems* 301 (2024): 112295, https://doi.org/10.1016/j.knosys.2024.112295.

[14] K. Guo, D. Tian, Y. Hu, et al., "Contrastive Optimized Graph Convolution Network for Traffic Forecasting," *Neurocomputing* 602 (2024): 128249, https://doi.org/10.1016/j.neucom.2024.128249.

[15] Y. Peng, Y. Jiang, J. Lu, and Y. Zou, "Examining the Effect of Adverse Weather on Road Transportation Using Weather and Traffic Sensors," *PLoS One* 13, no. 10 (2018): e0205409, https://doi.org/10.1371/journal.pone.0205409.

[16] M. S. Ahmed and A. R. Cook, "Analysis of Freeway Traffic Time-Series Data by Using Box-Jenkins Techniques" (1979).

[17] I. Okutani and Y. J. Stephanedes, "Dynamic Prediction of Traffic Volume through Kalman Filtering Theory," *Transportation Research Part B: Methodological* 18, no. 1 (1984): 1–11, https://doi.org/10.1016/0191-2615(84)90002-x.

[18] C. M. Kuchipudi and S. I. J. Chien, "Development of a Hybrid Model for Dynamic Travel-Time Prediction," *Transportation Research Record: Journal of the Transportation Research Board* 1855, no. 1 (2003): 22–31, https://doi.org/10.3141/1855-03.

[19] G. A. Davis and N. L. Nihan, "Nonparametric Regression and Short-Term Freeway Traffic Forecasting," *Journal of Transportation Engineering* 117, no. 2 (1991): 178–188, https://doi.org/10.1061/(asce)0733-947x(1991)117:2(178).

[20] C.-H. Wu, J. M. Ho, and D. T. Lee, "Travel-time Prediction with Support Vector Regression," *IEEE Transactions on Intelligent Transportation Systems* 5, no. 4 (2004): 276–281, https://doi.org/10.1109/tits.2004.837813.

[21] J. Zhang, Y. Zheng, and D. Qi, "Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 31, no. 1 (2017), https://doi.org/10.1609/aaai.v31i1.10735.

[22] T. N. Kipf and M. Welling, "Semi-supervised Classification with Graph Convolutional Networks," *arXiv preprint arXiv:1609.02907* (2016).

[23] A. Vaswani, "Attention Is All You Need," *Advances in Neural Information Processing Systems* (2017).

[24] J.. Devlin, "Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805* (2018).

[25] A.. Dosovitskiy, "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale" (2020).

[26] Z. Liu, Y. Lin, Y. Cao, et al., "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), 10012–10022.

[27] Y. Zou, "Combination of Multiple Transformer Models for Short-Term Freeway Traffic Speed Prediction Based on Bayesian Model Averaging," *Multidisciplinary Science Journal* (2025).

[28] X. Zong, Y. Qi, H. Yan, and Q. Ye, "An Intelligent Deep Learning Framework for Traffic Flow Imputation and Short-Term Prediction Based on Dynamic Features," *Knowledge-Based Systems* 300 (2024): 112178, https://doi.org/10.1016/j.knosys.2024.112178.

[29] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," *Stat* 1050, no. 20 (2017): 10–48550.

[30] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph Wavenet for Deep Spatial-Temporal Graph Modeling," *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence* (2019): 1907–1913, https://doi.org/10.24963/ijcai.2019/264.

[31] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting,"

Advances in Neural Information Processing Systems 33 (2020): 17804–17815.

[32] M. Kaya and H. Ş. Bílge, "Deep Metric Learning: A Survey," *Symmetry* 11, no. 9 (2019): 1066, https://doi.org/10.3390/sym11091066.

[33] W. Hu, M. Fey, M. Zitnik, et al., "Open Graph Benchmark: Datasets for Machine Learning on Graphs," *Advances in Neural Information Processing Systems* 33 (2020): 22118–22133.

[34] G. Li, C. Xiong, T. Ali, and B. Ghanem, "Deepergcn: All You Need to Train Deeper Gcns" (2020).

[35] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and E. D.. George, "Neural Message Passing for Quantum Chemistry," in *International Conference on Machine Learning* (PMLR, 2017), 1263–1272.

[36] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful Are Graph Neural Networks?" *arXiv preprint arXiv:1810.00826* (2018).

[37] F. Yu and V. Koltun, "Multi-scale Context Aggregation by Dilated Convolutions" (2015).

[38] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language Modeling with Gated Convolutional Networks," in *International Conference on Machine Learning* (PMLR, 2017), 933–941.

[39] S. Bai, J. Zico Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv preprint arXiv:1803.01271* (2018).

[40] S. Lan, Y. Ma, W. Huang, W. Wang, H. Yang, and P. Li, "Dstagnn: Dynamic Spatial-Temporal Aware Graph Neural Network for Traffic Flow Forecasting," in *International Conference on Machine Learning* (PMLR, 2022), 11906–11917.

[41] Y. Peng, D. Liu, S. Wu, X. Yang, Y. Wang, and Y. Zou, "Enhancing Mixed Traffic Flow with Platoon Control and Lane Management for Connected and Autonomous Vehicles," *Sensors* 25, no. 3 (2025): 644, https://doi.org/10.3390/s25030644.