Hindawi
Journal of Advanced Transportation
Volume 2022, Article ID 4672617, 8 pages
https://doi.org/10.1155/2022/4672617

Research Article

Rail Transit Prediction Based on Multi-View Graph Attention Networks

Li Wang, Xin Wang, and Jiao Wang

School of Computer Science, The Open University of China, Beijing 100039, China

Correspondence should be addressed to Li Wang; wlpolo@ouchn.edu.cn

Received 23 February 2022; Revised 16 May 2022; Accepted 24 May 2022; Published 6 July 2022

Academic Editor: Yong Zhang

Traffic prediction is the cornerstone of intelligent transportation system. In recent years, graph neural network has become the mainstream traffic prediction method due to its excellent processing ability of unstructured data. However, the network relationship in the real world is more complex. Multiple nodes and various associations such as different types of stations and lines in rail transit always exist at the same time. In an end-to-end model, the training accuracy will suffer if the same weights are assigned to multiple views. Thus, this paper proposes a framework with multi-view and multi-layer attention, which aims to solve the problem of node prediction involving multiple relationships. Specifically, the proposed model maps multiple relationships into multiple views. A graph convolutional neural network of multiple views with multi-layer attention learns the optimal regression of nodes. Furthermore, the model uses an autoencoder module to alleviate the over-smoothing problem during the training phase. With the historical dataset of Beijing rail transit, the experiment proves that the prediction accuracy of the model is generally better than the baseline traffic prediction algorithms.

1. Introduction

As the core function of the intelligent transportation system, traffic forecasting has practical significance for the actual needs of intelligent command and dispatch, traffic planning and layout, and public travel convenience. The prediction of passenger flow in and out of rail transit stations is one of the research hotspots in the field of smart transportation. An accurate passenger flow prediction method will be beneficial to the transportation system for reasonable route scheduling, road network design, crowd evacuation adjustment, and other specific applications. Most of the previous studies have focused on methods based on mathematical modeling as well as machine learning. However, in terms of rail transit, due to the unique topological structure of rail transit and the travel patterns of passengers, it is difficult to obtain efficient and accurate prediction results with the simple application of traditional methods, and related research is relatively limited.

In recent years, graph convolutional neural networks have achieved excellent performance in the field of traffic prediction by virtue of their excellent processing capabilities for non-Euclidean data. In fact, networks are ubiquitous in the real world, such as transportation networks, social networks, and recommendation networks. By modeling the network as a graph, subsequent prediction tasks can be performed. The graph-based non-Euclidean topology not only describes the connection relationship between stations, but also constrains the flow path of data. Therefore, the nongraph method can only make predictions for each station and average the prediction results, and cannot make full use of the topology of rail transit.

However, node relationships in the real world are more complex and contain many types of interrelated relationships. A view could represent a certain relationship. However, the node relationship information will be lost to an extent if only a single view is used for representation [1]. Multiple views can more accurately model different types of relationships, thereby ensuring that the model retains more comprehensive node information, which in turn enables more accurate node-level predictions. In rail transit, structurally, different types of lines and stations can be

assigned to different view features. On the other hand, from the perspective of traffic flow characteristics, the pattern of passenger travel in different time spans can be viewed as different spatial-temporal features [2]. However, when the model contains multiple node relationships at the same time, how to ensure that the model integrates different node relationships with optimal weights to achieve more accurate prediction becomes a key issue.

Since the same node has a different importance in different views, the relationships between nodes in different views should be given different weights. Conversely, the same weights will negatively affect the final prediction and weaken the meaning of the information provided by multiple views. Therefore, we design a multi-layer attention mechanism to achieve weight optimization for different views. In addition, during the training of the graph neural network, the problem of over-smoothing significantly affects the training effect as the number of network layers deepens. That is, the hidden layer representation of each node converges to the same value during the training process of the graph neural network, which eventually leads to poor training results.

In response to the above problems, we propose a traffic prediction model based on multi-view graph attention network (MV-GAT), and its main contributions can be summarized as follows:

(1) An end-to-end rail passenger flow prediction model is proposed. The proposed model achieves fine-grained multi-view modeling for rail transit characteristics at the input and node-level prediction at the output.

(2) Through the multi-layer attention module, the proposed model can assign different weights to different nodes and relationships within multiple views, thereby learning the optimal regression of nodes.

(3) In addition, the self-encoder module transfers the latent information captured by each layer of the self-encoder to the corresponding graph convolution layer, ensuring the validity of the structural information of each layer in the network, and further improving the effect of node prediction.

The model is evaluated through experiments on the Beijing rail transit historical dataset, and the superiority of the model is verified by comparison with existing models. Furthermore, multi-view and multi-layer attention have good interpretability, as shown in ablation experiments.

2. Related Work

The research content of this paper mainly involves graph convolutional neural network and graph attention mechanism.

2.1. Graph Convolutional Networks. Graph convolutional networks (GCNs) are currently used in many domains such as traffic prediction [3], recommender systems [4, 5], and traffic situation analysis [6]. On graphs, its tasks include graph classification [7], node classification [8], link prediction [9, 10], and graph pooling [11]. GCNs have different kernels that learn node embeddings to be applied to downstream tasks. For example, DeepWalk [12] and node2vec [13] are both random walk-based methods. The model SDNE [14] uses autoencoders to maintain the proximity of first- and second-order networks, using highly nonlinear functions to obtain embeddings. Existing traffic flow forecasting techniques include traditional mathematical modeling methods, such as ARIMA [15], as well as deep learning methods. Among them, deep learning methods are subdivided into nongraph-based methods, such as LSTM [16], and nongraph-based methods, such as GCN models. Traditional mathematical modeling methods as well as nongraph-based methods do not consider the topology of the graph and can only make individual predictions for individual sites. Deep learning methods based on graphs can achieve node-level prediction, but currently the mainstream methods are mainly single view [17].

Single-view graph neural networks contain only one relationship between nodes [18]. Although single view has many advantages, such as easy to understand and easy to design neural network models, it is difficult to accurately capture the complex relationships between nodes, which play a crucial role in the effectiveness of information transfer and problem solving [19]. It has been pointed out that graph data possess similarity information between different nodes, which in turn has been proposed to preserve similarity information in the hidden layer of graph convolutional neural networks [20]. However, these methods rarely exploit the multi-view prediction in end-to-end network models.

2.2. Graph Attention Mechanism. The attention mechanism was first proposed for natural language processing and has now been widely used for many sequence-related tasks. The advantage of the attention mechanism is that it can amplify the impact of important parts of a sequence, and the introduction of the attention mechanism also facilitates the use of graph neural networks. Because graph convolutional networks rely on the eigenvalues of the Laplacian matrix, it is difficult to extract convolutional operations from the overall static graph structure. In an attention network, the output at a given moment depends on the attention it allocates across multiple inputs, i.e., the learning weight assigned to each part of the input, with larger weights implying the output of the pair at that particular moment.

As the attention mechanism in the seq2seq model [21], each output is affected by the different weights assigned to the different inputs. The concept of hard attention [22] is designed as a stochastic process that uses Monte Carlo sampling methods to estimate the gradient of the module, thus enabling back-propagation of the gradient. In addition, attention mechanisms include global attention and local attention [23], as well as multi-headed attention [24]. Multi-headed attention is used to extract features more comprehensively by mapping node representations into multiple node representations through linear mapping and

combining the computational results. Inspired by the above work, the possibility of using a multi-layer attention mechanism to fuse multi-view information to reveal the deep relationships between nodes becomes one of our considerations.

3. Methodology

The necessary preliminaries are firstly illustrated, followed by introducing of the overall architecture of the proposed model, and then the details of each component are elaborated.

3.1. Preliminaries. This section will introduce some concepts and symbols used in this paper. For a regular graph G with vertex set V, the edge set E and weight W can be denoted as $G = (V; E = (e_i)_{i\in I}; W)$. For an undirected graph, the incidence matrix $H \in \mathbb{R}^{N\times I}$ can be defined as

$$h(v,e) = \begin{cases} 1, & \text{if } v \in e, \\ 0, & \text{if } v \notin e. \end{cases} \tag{1}$$

For the vertices in graph, the degree is defined as the sum of all weights connected to the vertices; for the edges in graph, the degree is defined as the total number of vertices connected by the edge:

$$\begin{cases} d(v) = \sum_{e\in E} w(e_i)h(v,e_i), & i \in I, \\ \delta(e) = \sum_{v\in V} h(v,e_i), & i \in I. \end{cases} \tag{2}$$

In the process of modeling information in real life, usually only a single view is used to represent the relationships between nodes. A single view contains only one relationship, but due to the complex relationship in real life, it is difficult to capture the comprehensive node relationship with only one view, which will inevitably lead to the omission of information, which will lead to deviations in the subsequent processing of the model. A multi-view contains various relationships between nodes. It can capture structural information more accurately than a single view and better discover implicit relationships between nodes.

Thus, a multi-view graph can be denoted as $G = (V, E^{(1)}, E^{(2)}, \ldots, E^{(m)}, X)$, which $V = \{v_i\}_{i=1}^{n}$ represents the set of nodes in the graph. $e_{i,j}^{(m)} \in E^{(m)}$ indicates the m-th view, node i is connected to node j, and $x_i \in X$ denotes node feature v_i. The node structure in Graph G can be represented by multiple adjacency matrices $\{A^{(m)}\}_{m=1}^{M}$; if $e_{i,j}^{(m)} \in E^{(m)}$, then $a_{i,j}^{(m)} = 1$; otherwise $a_{i,j}^{(m)} = 0$. In our work, the connection between the node and itself is not considered, i.e., $a_{i,i}^{(m)} = 0$.

The purpose of the work is to predict traffic flow with the proposed model. The input of the model is the historical transit flow data $\mathbf{X}^t = (x_1^t, x_2^t, \ldots, x_N^t) \in \mathbb{R}^{N\times C\times T}$, where N indicates the total number of vertices, C is the number of channels of the feature, and T is the time dimension. At the output end of the proposed model, node-level prediction results are supposed to be obtained, which can be denoted as $\mathbf{Y}^{\mathrm{t+m}} = (y_1^{t+m}, y_2^{t+m}, \ldots, y_N^{t+m}) \in \mathbb{R}^N$.

3.2. MV-GAT: The Proposed Model. For complex relationships between entities in the real world, it is difficult to fully grasp the node structure information if only a single view is used to represent the node relationships. In rail transit, considering only the line connections between stations ignores the relationships between stations at the feature level, such as the OD characteristics of passenger trips between stations with different time spans. During the morning and evening peak hours, large passenger trips show relatively fixed patterns, which can also be used as a view for traffic flow prediction. At the same time, it is important to avoid the problem of premature model fitting as the number of layers of the network model increases. When the model uses multiple views as input, how to fuse these views becomes a new problem. The fusion process must ensure that the model can ignore noisy information and that the most relevant information of the nodes is extracted among the multiple views.

To address the above issues, we propose the overall framework of the model, as shown in Figure 1. The basic idea is to use the multi-layer attention module to capture the node information contained in the multi-view to ensure that the best node representation can be learned, and to use the autoencoder module to ensure that the model learns the structural information between the data, which is represented as a multi-view graph.

In the multi-view module, multiple views are used to ensure complete information extraction. Specifically, in this forecasting task, the multiple views include a static view based on the connectivity of tracks and routes, and an OD view of passenger flows for three different time spans: hourly, daily, and weekly.

The autoencoder module learns the accurate data representation and mitigates the over-smoothing problem. The two parts of the input are connected to the autoencoder module and the GCN module, and each layer in the autoencoder module is guaranteed to be connected to the corresponding GCN layer, so that the structural information between nodes learned in the autoencoder can be integrated into the GCN module.

In the multi-layer attention module, multi-layer attention is used to fuse the multi-view information to obtain an optimal representation of the data. The multi-layer attention module ensures that the model learns different weights at different nodes and in different views.

3.3. Multi-View Graph Convolution. For the single-view graph, the input is $G_k = (A_k, X)$. The multi-view graphs generated by the relationship between the nodes are $G_m = (A_m, X_{att}^{(m)})$, where m is the number of views. Each input is fed into an exclusive convolution module. The output of the convolution is Z_k and Z_m. Take Z_k; for example, the output of the l-th layer of the graph convolution can be expressed as

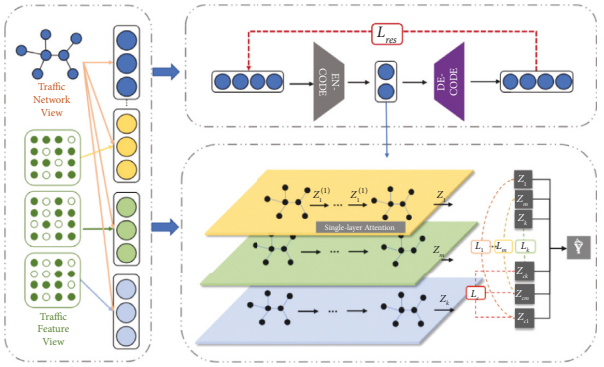

FIGURE 1: The end-to-end framework of proposed model. The MV-GAT model includes multi-view input, multi-layer attention module, and the autoencoder module. The node-level prediction results are obtained as output.

$$Z_m^{(l)} = ReLU\left(\tilde{D}^{-1/2}\widetilde{A_m}\tilde{D}^{-1/2}Z^{(l-1)}W^{(l)}\right), \tag{3}$$

where $W^{(l)}$ is the weight matrix of GCN at the l-th layer, the preliminary $Z_m^{(0)} = X_{att}^{(m)}$, and $X_{att}^{(m)}$ is the node embedding learned by single-view attention network in view m. $Z_m^{(0)} = A_m$. $\widetilde{A_m} = A_m + I$, and $\tilde{D}$ is the diagonal matrix of $\tilde{A}$.

It is difficult for multi-view convolution to learn the commonality between different views only by learning each view individually, so multi-view convolution is supposed to be added to extract common information between different views. The proposed model uses previously constructed input graphs G_k and G_m as inputs to multi-view convolution, the output of multi-view convolution module is Z_c, and the output of the l-th layer of the convolution can be expressed as

$$Z_c^{(l)} = ReLU\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}Z^{(l-1)}W^{(l)}\right), \tag{4}$$

where $W^{(l)}$ is the weight matrix of the l-th layer of GCN, the preliminary Z is $Z^{(0)} = X$, $\tilde{A} = A + I$, and $\tilde{D}$ is the diagonal matrix of $\tilde{A}$.

3.4. Autoencoder Module. The proposed method introduces an autoencoder to learn the structural information of the data and pass the learned information to the corresponding GCN layers, and the added autoencoder module also helps to alleviate the over-smoothing problem of the GCN.

Assuming that the autoencoder has L layers, the expression learned in the l-th layer in the autoencoder is $H^{(l)}$:

$$H^{(l)} = ReLU\left(W_e^{(l)}H^{(l-1)} + b_e^{(l)}\right). \tag{5}$$

In the formula, *ReLU* is the activation function of the fully connected layer, and $W_e^{(l)}$ and $b_e^{(l)}$ are the weight matrix and bias of the l-th layer in the autoencoder. In addition, $H^{(0)}$ is the feature matrix X. Then, the input data of decoding part are reconstructed through the fully connected layer.

$$H^{(l)} = ReLU\left(W_d^{(l)}H^{(l-1)} + b_d^{(l)}\right). \tag{6}$$

Here, $W_d^{(l)}$ and $b_d^{(l)}$ are the weight matrix and bias of the l-th layer of decoder. In order to pass the node representation into the GCN module, the node representations are learned from the autoencoder, such as $H^{(1)}, H^{(2)}, \ldots H^{(L)}$. After being passed into the GCN module, the GCN can hold two different kinds of information, the data itself and the data structure. For example, the output of l-th layer learned in the single view can be expressed as $Z_k^{(l)}$.

The representation $H^{(l)}$ learned by the autoencoder can reconstruct the data itself and contains a different valuable information. Combining the two representations leads to a more complete representation.

$$\widetilde{Z}_k^{(l-1)} = (1-\epsilon)Z_k^{(l-1)} + \epsilon H^{(l-1)}. \tag{7}$$

Here, ϵ is the balance coefficient with an initial setting of 0.5. In this way, the autoencoder and GCN can be connected layer by layer. We use ReLU as the activation function to solve the gradient vanishing problem.

3.5. Multi-Layer Attention. Since the model takes multiple views as input, the proposed method designs a multi-layer attention module to effectively integrate the node representations learned in different views to form an optimal combination. First, the proposed method uses a single-view attention layer to learn the influence of different neighbor nodes on the predicted node in the same view. Then, a multi-view attention layer is used to learn the influence of different views on the predicted node. Finally, the two parts are combined to obtain the optimal weighted combination of the nodes to be predicted.

In the single-view attention layer, the influence of different neighbor nodes on the predicted node in each view can be learned. Since each node plays a different role in the process of node embedding, the impact on the final node prediction result is also different. Self-attention is thereby used to learn the weights between each node. For instance, in the view m, calculating the attention index of a pair of nodes (i, j) can be formulated as

$$e_{ij}^{(m)} = att\left(x_i, x_j\right). \tag{8}$$

Here, *att* represents the attention mechanism, and since the multiple views are undirected graphs, the importance of node i to node j is the same as node j to node i. Therefore, $e_{ij}^{(m)}$ is a symmetric matrix.

After calculating the $e_{ij}^{(m)}$ of node j, the weight coefficient is normalized as

$$\begin{aligned}\alpha_{ij}^{(m)} &= \mathrm{softmax}_j\left(e_{ij}^{(m)}\right)\\ &= \frac{\exp\left(\mathrm{LeakyReLU}\left(a_m^T\cdot\left[x_i\|x_j\right]\right)\right)}{\sum_{k\in N}\exp\left(\mathrm{LeakyReLU}\left(a_m^T\cdot\left[x_i\|x_k\right]\right)\right)}.\end{aligned} \tag{9}$$

In the equation, $\|$ represents the connection operation, and a_m^T is the attention vector in the single view. The node embedding of node i in the view can be obtained by the feature aggregation of neighbor nodes with feature

coefficients. Multi-head attention is utilized in order to make the training process more stable. Softmax and ReLU are both activate functions. Specifically, the single-view attention layer repeats K times and connects the learned embedding to a specific view. The learned node embedding and feature matrix are spliced to get $X_{att}^{(m)}$. In the following equation, $z_i^{(m)}$ is the embedding of node i learned in the view m.

$$z_i^{(m)} = \Big\Vert_{k=1}^{K} \text{Sigmoid}\left(\sum_{j \in N} \alpha_{ij}^{(m)} \cdot x_j \right). \tag{10}$$

A single view contains only one type of relationship between nodes, while a multi-view contains relationships between different nodes. To learn more comprehensive node embeddings, it is necessary to integrate multiple node embeddings learned from different views. For different nodes or associations, the weights assigned to different views are different, so it is necessary to design a multi-view attention layer that automatically assigns different weights to different views to solve this problem.

The input of multi-view attention layer is the single-view graph convolution Z_k and $Z_{(m)}$ and the multi-view convolution Z_c, and the attention mechanism $att(Z_k, Z_{(m)}, Z_c)$ learns the weights corresponding to different views $(\alpha_k, \alpha_{(m)}, \alpha_c)$:

$$\left(\alpha_k, \alpha_{(m)}, \alpha_c\right) = att\left(Z_k, Z_{(m)}, Z_c\right). \tag{11}$$

Here, $\alpha_k, \alpha_{(m)}, \alpha_c$ are the attention weights of different views, respectively. For node i, a nonlinear transformation is applied on the node embedding, and then the shared attention vector q is taken to calculate the attention value ω_m^i.

$$\omega_m^i = q^T \cdot \tanh\left(W \cdot \left(z_m^i\right)^T + b\right). \tag{12}$$

The W is weight matrix and b is bias. The attention index of node i in other embedding matrices can be obtained in the same way. Then, the final weight can be calculated by normalizing multiple attention values.

$$\begin{aligned} \alpha_m^i &= \text{softmax}\left(\omega_m^i\right) \\ &= \frac{\exp\left(\omega_m^i\right)}{\exp\left(\omega_m^i\right) + \exp\left(\omega_m^i\right) + \exp\left(\omega_m^i\right)}. \end{aligned} \tag{13}$$

The multiple embeddings are then linearly combined. The larger the α_m^i, the more important the view is.

$$Z = \alpha_k \cdot Z_k + \alpha_{(m)} \cdot Z_{(m)} + \alpha_c \cdot Z_c. \tag{14}$$

The above multi-view attention module solves the problem of assigning different weights to the views, thereby enabling adaptive inter-view importance learning.

3.6. Objective Function. In order to allow the convolution to capture richer information, we increase the difference between Z_k, Z_m, Z_c. Here, we take advantage of the Hilbert–Schmidt independence criterion (HSIC) to measure the independence between the outputs:

$$\text{HSIC}\left(Z_i, Z_j\right) = (n-1)^{-2} tr\left(RK_iRK_j\right). \tag{15}$$

Here, K_iK_j is the Gram matrix, $k_{i,ij} = k_i(z_i^i, z_i^j)$, $k_{j,ij} = k_j(z_j^i, z_j^j)$. And $R = I - 1/nee^T$. I is the identity matrix, and e is the corresponding identity column vector. In the same way, all other views are also calculated by HSIC, denoted as L_s.

The multi-view loss function is supposed to learn as much consistency between different views as possible. After normalizing the matrices $\{Z_{ci}\}_{i=1}^4$ to $\{Z_{\text{cinor}}\}_{i=1}^4$ with L2 normalization, the similarity between nodes $\{S_i\}_{i=1}$ is calculated, and the sum is denoted as L_m.

$$\{S_i\}_{i=1} = Z_{cinor} \cdot Z_{cinor}^T. \tag{16}$$

Since in the autoencoder module, the output of the decoder is the reconstructed original data. The node-level traffic flow prediction results will be output through a complete fully connected layer, and the multi-channel is mapped to a single channel, which can be expressed as

$$L_p = \sum_t \left\| \left(\hat{X}^t + b\right) - X^t \right\|^2. \tag{17}$$

The final loss function is L, where a, b are the parameters.

$$L = L_{\text{p}} + aL_m + bL_s. \tag{18}$$

4. Experiments and Results

The proposed MV-GAT model is evaluated by comparing it with state-of-the-art baselines. The experimental dataset and baselines are first introduced, followed by the parameter setup. Finally, the experimental results and experimental analysis are presented.

4.1. Experiment Setting. We adopt the historical data of Beijing metro as the experimental dataset. MetroBJ [25] is a five-month passenger flow dataset, formally collected in 2015, with a granularity of 5 minutes. The dataset covers the entire subway network with 325 stations and 22 lines, covering the daily traffic data in July, August, September, November, and December. The time horizon is five months, covering weekdays and weekends. This time series contained in this dataset is long enough for us to divide multiple time spans to build multiple feature-level views.

The dataset contains the desensitized swipe ID, the line station and time of entering the subway, and the traffic flow data of the line station and time of leaving the subway. In the actual use of this method, firstly, a node set containing 325 nodes is constructed based on the subway stations in Beijing in this dataset, and a basic view containing 22 edges is constructed with reference to the subway network lines. On this basis, the DBSCAN algorithm is used to cluster the historical passenger flow data under three different time spans of hours, days, and weeks, and construct corresponding multi-views. Compared with the traditional

k-means algorithm, the DBSCAN algorithm does not need to input the number of clusters k and can find clusters of any shape, and at the same time, it can find outliers during clustering. Finally, the traffic flow values of each node in the next 5 minutes, 10 minutes, and 15 minutes are output to calculate the accuracy of the proposed model.

The comparison methods include two categories of nongraph methods and graph-based methods. The compared methods contain autoregressive integrated moving average (ARIMA) model [26], support vector regression (SVR) [27], and long short-term memory (LSTM) [28]. Graph-based deep learning methods contain temporal graph convolutional network (T-GCN) [29], spatio-temporal graph convolutional network (STGCN) [30], and diffusion convolutional recurrent neural network (DCRNN) [31]. The detailed parameter settings are listed as follows.

(1) ARIMA: ARIMA is a common time series forecasting methods. The degree of differencing d, lag order p, and the order of moving average q are determined with the "auto arima" in the "pyramid" library.

(2) SVR: One improvement of SVR is the tolerated deviation ε when calculating the loss. During training, the model with linear kernel has a penalty term C of 0.1 and a deviation ε of 0.1.

(3) LSTM: The compared LSTM model has hidden layers of [31] recurrent units. During the training phase, the batch size is 32, the activation function is sigmoid, and the learning rate is set to 10^{-2}.

(4) T-GCN: The temporal graph convolutional network has hidden units of GRU. The batch size is set to 64 while training, and the learning rate is set to 10^{-3}.

(5) STGCN: The spatial-temporal graph convolutional network has two convolution blocks with channel of [64,16,64]. The convolution kernel size is 3, and the batch size is 64.

(6) DCRNN: The diffusion convolutional recurrent neural network is a data-driven traffic prediction model with autoencoder framework. It has two RNN layers of 64 units. The batch size is set to 64, and the learning rate is set to 10^{-3}.

To quantitatively evaluate the prediction accuracy of the proposed method, the results of the experiments take mean absolute error (MAE) and root mean square error (RMSE) as performance metrics:

$$\begin{aligned} \mathrm{MAE} &= \sum_{i=1}^{T}\sum_{j=1}^{N}\frac{\left|X_{ij}-\widehat{X}_{ij}\right|}{T*N} \\ \mathrm{RMSE} &= \sum_{i=1}^{T}\sum_{j=1}^{N}\frac{\left(X_{ij}-\widehat{X}_{ij}\right)^{2}}{(T*N)^{1/2}}, \end{aligned} \tag{19}$$

where X_{ij} is the ground truth, the $\widehat{X}_{ij}$ is prediction value, T is the time length, and N is the node number. When MAE and RMSE are used as evaluation indicators, the lower the value, the higher the accuracy. All experiments are tested with the platform of CPU of "Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90 GHz" and GPU of "NVIDIA GTX 2080Ti." The number of epochs of training phase is 50, and the batch size is 64. The learning rate is set to 10^{-2} and decreases to 10^{-4} gradually.

4.2. Experiment Results. To fully utilize the different views over multiple time spans, we use the data of a whole month as the experimental data. The experiments use ten-fold cross-validation to get stabler results. Considering the size of dataset per month, the training set, testing set, and valid set are split with 8 : 1 : 1 on the time dimension. The experimental results are shown in Table 1.

Table 1 shows the prediction accuracy when the historical data of July and September are used as the experimental dataset. As can be seen from the results, the accuracy of the ARIMA method is significantly lower than that of the machine learning and deep learning methods. SVR significantly outperforms ARIMA, and at the same time, LSTM is better than SVR by virtue of modeling long- and short-term sequences. T-GCN, an earlier method that combines graph networks with time series dependency, achieves similar accuracy to the relatively mature LSTM.

As a classical framework, STGCN has achieved more accurate prediction results, especially in the medium-term prediction of the next 45 minutes, where obvious advantages can be seen. With a unique architecture, DCRNN also achieves accurate results. Among all methods, our proposed method achieves better accuracy, especially on short-term predictions of 15 minutes and 30 minutes. Compared with machine learning methods and graph-based deep learning methods, there are significant improvements. More experimental results of flow prediction are shown in Table 2.

It can be seen from Table 2 that the prediction accuracy of each method is similar to that presented in Table 1. It shows that the rail transit shows a basically stable operation law in each month. It is worth mentioning that, similar to the previous set of experiments, STGCN achieves a clear advantage in 45-minute prediction results. It reflects the complexity of traffic forecasting from the side. In many cases, it is difficult to solve short-term forecasting, medium-term forecasting, and even long-term forecasting problems simultaneously with one model.

To prominently compare the role of each module of the proposed model, we design a set of ablation contrast experiments, as shown in Table 3. In this set of ablation experiments, we mainly compared the difference between single view and multi-view, and the role of the autoencoder.

The experimental data adopt the passenger flow data of Beijing rail transit in July. We first tested the single-view network model without the autoencoder module. The single view is the graph of the rail transit network. While removing the autoencoder, other parts of the proposed model remain unchanged. It can be seen that the prediction accuracy of this method is unsatisfactory, and it cannot even beat the STGCN model on this dataset. In the case of single view, whether the multi-layer attention mechanism has the effect of negative optimization is a new problem worth investigating.

Table 1: Accuracy results of rail passenger flow prediction experiment.

Methods	July		September	
	MAE	RMSE	MAE	RMSE
ARIMA	18.34/20.12/23.32	29.14/33.37/36.81	19.64/21.31/24.06	30.74/34.30/36.66
SVR	14.73/16.55/18.26	25.24/31.33/32.71	15.89/16.71/17.36	28.30/33.01/34.09
LSTM	10.76/12.27/12.86	21.22/22.33/23.74	11.95/12.56/13.77	23.95/26.43/28.34
T-GCN	10.88/12.46/12.73	20.93/22.72/24.61	11.00/12.77/13.75	23.98/25.98/28.35
STGCN	9.04/10.29/**10.88**	19.38/20.49/22.51	10.99/11.95/13.42	21.03/23.03/**24.06**
DCRNN	8.41/9.73/11.56	19.43/23.76/25.77	8.72/9.24/12.73	20.94/**22.01**/25.82
MV-GAT	**8.35/9.57**/11.60	**19.41/21.84/22.38**	**8.67/9.13/12.45**	**20.85**/22.13/25.67

Table 2: Accuracy results of rail passenger flow prediction experiment.

Methods	November		December	
	MAE	RMSE	MAE	RMSE
ARIMA	14.22/18.81/24.39	30.06/33.52/35.24	18.39/20.52/24.22	30.24/33.81/35.06
SVR	13.92/16.52/16.12	26.89/28.75/28.56	14.12/15.75/16.92	26.56/28.52/28.89
LSTM	11.49/13.79/14.05	21.36/22.33/25.50	11.05/12.33/14.49	21.50/22.79/25.36
T-GCN	10.99/12.76/13.44	21.48/23.68/25.62	10.90/13.78/13.34	21.42/23.74/25.85
STGCN	9.06/10.65/11.32	21.74/22.22/**23.55**	8.20/10.26/**11.25**	20.72/22.28/**23.25**
DCRNN	8.83/9.71/11.79	21.52/23.15/26.04	8.24/9.65/11.87	21.28/23.24/26.95
MV-GAT	**8.80/9.62/11.20**	**20.83/22.18**/23.39	**8.15/9.57**/11.43	**20.64/22.13**/24.72

Table 3: Ablation contrast experiment.

Methods	July	
	MAE	RMSE
Single view w/o autoencoder	9.21/10.31/12.38	19.68/22.49/23.51
Single view w/autoencoder	8.98/10.20/12.21	19.61/22.27/23.19
Multi-view w/o autoencoder	8.44/9.61/11.69	19.50/21.91/22.49
Multi-view w/autoencoder	**8.35/9.57/11.60**	**19.41/21.84/22.38**

By adding the autoencoder module to the single-view model, the prediction accuracy is improved, but the improvement is relatively limited. The autoencoder module can alleviate the gradient vanishing problem during training to a certain extent, especially for graph convolutional deep network models with many layers. Limited by the graph scale of the dataset used in this experiment, the number of layers in the network model is not many. Therefore, in the deeper graph convolution prediction model, it is worth looking forward to whether the autoencoder module can play a larger role.

After the introduction of multi-view, the prediction accuracy of the model is significantly improved compared to single view, with or without an autoencoder module. Among them, the model achieves the best prediction results when the multi-view module and the autoencoder module coexist.

5. Conclusions

This paper proposes a multi-view and multi-layer attention-based GCN model for the problem of rail traffic flow prediction. Considering that it is difficult to fully express the relationship between nodes in the node classification problem using only a single view, this model introduces multi-view and utilizes a multi-layer attention mechanism and an autoencoder module to achieve more accurate temporal prediction. Experimental results on the Beijing dataset show that our model outperforms other nongraph and graph-based benchmark methods. In the future, we will optimize the framework of the proposed method and try to design models for directed graphs. We also want to explore more comprehensively the application of graph-based deep learning in intelligent transportation systems.

Data Availability

The data supporting this proposed model are from previously reported studies and datasets, which have been cited. The processed data are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The research of this article was funded by Building Project for Continuing Education Online Resource Platform and Characteristic Professional High-Level Team of Colleges and

Universities and Beijing Municipal Education Commission, nos. 2019-630 and 2019.12.

References

[1] X. Wang, H. Ji, C. Shi, and B. Wang, "Heterogeneous graph attention network," in *Proceedings of the The world wide web conference*, pp. 2022–2032, San Francisco, CA, USA, May 2019.

[2] Y. Wang, Y. Zhang, X. Piao, H. Liu, and K. Zhang, "Traffic data reconstruction via adaptive spatial-temporal correlations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 4, pp. 1531–1543, 2019.

[3] J. Wang, Y. Zhang, L. Wang, X. Piao, and B. Yin, "Multitask hypergraph convolutional networks: a heterogeneous traffic prediction framework," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[4] R. Ying, R. He, K. Chen, P. Eksombatchai, and W. L. Hamilton, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983, London, U.K, August 2018.

[5] W. Fan, Y. Ma, Q. Li, Y. He, and E. Zhao, "Graph neural networks for social recommendation," in *Proceedings of the The World Wide Web Conference*, pp. 417–426, San Francisco, CA, USA, May 2019.

[6] G. Huo, Y. Zhang, B. Wang, and Y. Hu, "Text-to-Traffic generative adversarial network for traffic situation generation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2623–2636, 2021.

[7] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the 32 AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, New Orleans, LA, USA, February 2018.

[8] J. Wu, J. He, and J. Xu, "Net: degree-specific graph neural networks for node and graph classification," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 406–415, Anchorage, Alaska, USA, July 2019.

[9] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proceedings of the International Conference on Machine Learning. PMLR*, pp. 7134–7143, Long Beach, CA, USA, June 2019.

[10] T. N. Kipf and M. Welling, "Variational graph autoencoders," 2016, https://arxiv.org/abs/1611.07308.

[11] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, and K. Lerman, "Mixhop: higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proceedings of the international conference on machine learning. PMLR*, pp. 21–29, Long Beach, CA, USA, June 2019.

[12] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, New York, NY, USA, February 2014.

[13] A. Grover and J. Leskovec, "node2vec: scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, New York, NY, USA, July 2016.

[14] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234, New York, NY, August 2016.

[15] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: theoretical basis and empirical results," *Journal of Transportation Engineering*, vol. 129, no. 6, pp. 664–672, 2003.

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

[17] X. Wang, M. Zhu, D. Bo, P. Cui, and C. Shi, "Am-gcn: adaptive multi-channel graph convolutional networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1243–1253, New York, NY, USA, August 2020.

[18] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *Proceedings of the Web Conference 2020*, pp. 1400–1410, Taipei Taiwan, April 2020.

[19] H. Nt and T. Maehara, "Revisiting graph neural networks: all we have is low-pass filters," 2019, https://arxiv.org/abs/1905.09550.

[20] H. Gao, J. Pei, and H. Huang, "Conditional random field enhanced graph convolutional neural networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 276–284, Anchorage, Alaska, USA, May 2019.

[21] K. Cho, B. Van Merriënboer, C. Gulcehre et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, https://arxiv.org/abs/1406.1078.

[22] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, and R. Salakhutdinov, "Show, attend and tell: neural image caption generation with visual attention," in *Proceedings of the International conference on machine learning. PMLR*, pp. 2048–2057, Lille, France, July 2015.

[23] M. T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, https://arxiv.org/abs/1508.04025.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, and A. N. Gomez, "Attention is all you need," 2017, https://arxiv.org/abs/1706.03762.

[25] J. Wang, Y. Zhang, Y. Wei, Y. Hu, X. Piao, and B. Yin, "Metro passenger flow prediction via dynamic hypergraph convolution networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7891–7903, 2021.

[26] S. Lee and D. B. Fambro, "Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1678, no. 1, pp. 179–188, 1999.

[27] R. Chen, C.-Y. Liang, W.-C. Hong, and D.-X. Gu, "Forecasting holiday daily tourist flow based on seasonal support vector regression with adaptive genetic algorithm," *Applied Soft Computing*, vol. 26, pp. 435–443, 2015.

[28] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic flow prediction," in *Proceedings of the 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 324–328, IEEE, Wuhan, China, November 2016.

[29] L. Zhao, Y. Song, C. Zhang, T. Lin, M. Deng, and H. Li, "T-gcn: a temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, 2019.

[30] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," 2017, https://arxiv.org/abs/1709.04875.

[31] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: data-driven traffic forecasting," 2017, https://arxiv.org/abs/1707.01926.