applied sciences

Article

An Urban Traffic Flow Fusion Network Based on a Causal Spatiotemporal Graph Convolution Network

Xing Xu [1], Hao Mao [2], Yun Zhao [1,*] and Xiaoshu Lü [3]

[1] School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China; xuxing3220@163.com
[2] School of Mechanical and Energy Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China; mh20220919@163.com
[3] Department of Electrical Engineering and Energy Technology, University of Vaasa, 65380 Vaasa, Finland; xiaoshu.lu@aalto.fi
* Correspondence: zy_super0201@163.com; Tel.: +86-13857186138

Abstract: Traffic flow prediction is an important part of intelligent transportation systems. In recent years, most methods have considered only the feature relationships of spatial dimensions of traffic flow data, and ignored the feature fusion of spatial and temporal aspects. Traffic flow has the features of periodicity, nonlinearity and complexity. There are many relatively isolated points in the nodes of traffic flow, resulting in the features usually being accompanied by high-frequency noise. The previous methods directly used the graph convolution network for feature extraction. A polynomial approximation graph convolution network is essentially a convolution operation to enhance the weight of high-frequency signals, which lead to excessive high-frequency noise and reduce prediction accuracy to a certain extent. In this paper, a deep learning framework is proposed for a causal gated low-pass graph convolution neural network (CGLGCN) for traffic flow prediction. The full convolution structure adopted by the causal convolution gated linear unit (C-GLU) extracts the time features of traffic flow to avoid the problem of long running time associated with recursive networks. The reduction of running parameters and running time greatly improved the efficiency of the model. The new graph convolution neural network with self-designed low-pass filter was able to extract spatial features, enhance the weight of low-frequency signal features, suppress the influence of high-frequency noise, extract the spatial features of each node more comprehensively, and improve the prediction accuracy of the framework. Several experiments were carried out on two real-world real data sets. Compared with the existing models, our model achieved better results for short-term and long-term prediction.

Keywords: traffic flow forecasting; graph convolution network; deep learning; smart city

check for updates

Citation: Xu, X.; Mao, H.; Zhao, Y.; Lü, X. An Urban Traffic Flow Fusion Network Based on a Causal Spatiotemporal Graph Convolution Network. *Appl. Sci.* **2022**, *12*, 7010. https://doi.org/10.3390/app12147010

Academic Editors: Yuchuan Du and Yu Shen

Received: 17 May 2022
Accepted: 7 July 2022
Published: 11 July 2022

1. Introduction

Three factors are involved in traffic flow, i.e., flow, occupancy, and speed, and these factors determine the extent of road congestion. Usually, traffic flow is predicted for a short term (5–30 min) or a long term (more than 30 min). Previous popular statistical methods (e.g., linear regression) perform satisfactorily in short-term prediction but are ineffective in the long term because there a range of factors that impact traffic flow. In addition, the factors and their interdependencies vary over time.

Traffic flow prediction is a kind of spatiotemporal prediction problem. For spatiotemporal prediction problems, the focus of research is how best to obtain the temporal correlation and spatial correlation. The development of traffic flow prediction methods has experienced several stages: statistical methods, machine learning methods and deep learning. Among these, deep learning methods have become a research hotspot in recent years because of their high efficiency of obtaining spatial and temporal correlations.

Related Work

- Non-Depth Learning Method

The early research on traffic flow prediction generally adopted traditional statistical theory and methods, including historical average (HA), the autoregressive integrated moving average model (ARIMA), the Kalman filter, and other methods. Williams et al. [1], according to the periodic change of traffic flow in urban areas, considered that a one week lag seasonal difference applied to traffic data in unrelated sections would produce a weak average transformation. The original traffic data was transformed into a time series with stable mean and variance, and then ARIMA was used to model the transformed time series and make empirical analysis on the data of the intelligent transportation system. The accuracy of the conjecture was verified. The premise of time series modeling is that the data is stable. However, traffic flow data is usually unstable. Although weakly stable data can be obtained after processing, such methods require complex parameter estimation and a large number of high-quality historical data. Therefore, intelligent application in short-term prediction (less than 30 min) cannot meet the needs of medium- and long-term predictions. Okutani et al. [2] and Kuchipudi et al. [3] used the Kalman filter model to predict traffic flow and commuting time. In their research, they considered not only the road section under investigation, but also the sections related to that, thus introducing more spatial information to the model. The Kalman filter uses a recursive spatial model, which overcomes the limitation that time series modeling can only be applied to stationary series. However, the Kalman filter is a linear model and the reusability of its parameters is poor. With the development of data-driven methods, machine learning methods have also been applied to traffic-flow prediction. Sun Hongyu et al. [4] proposed application of the local linear regression model to short-term traffic prediction. This study was compared with previous nonparametric methods (such as neighborhood and kernel methods), and it was found that the local linear method was always better than the nearest-neighborhood or kernel-smoothing methods. Zheng et al. [5] and Wu et al. [6] used support vector regression (SVR) to fit historical data to predict traffic events. Mou et al. [7] proposed a short-term traffic flow prediction model based on wavelet denoising and a Bayesian network. In addition, machine learning methods to complete the predictions include random forests (RF) [8], gradient boosting decision tree (GBDT) [9], XGBoost (Extreme Gradient Boosting) [10]. Machine learning methods are often better than traditional statistical learning methods, but machine learning methods rely on manual construction of features, and different feature selection has a great impact on the results. On the other hand, the structure of machine learning methods is usually shallow and the expression ability is very limited.

- Deep Learning Method

Deep learning has shown strong learning ability and representation ability in many fields. It can model linear, nonlinear, stationary and non-stationary data, and has high scalability. Deep learning has been applied to traffic flow and other spatial–temporal prediction problems; this has been a gradual process with the development of deep learning. Feedforward neural networks (FNN) were an early deep learning method used in traffic flow prediction. Park et al. [11] designed a feedforward neural network with a three-layer structure: input layer, hidden layer and output layer. The number of neurons in the hidden layer was dependent on the number of input data roads. When the numbers of input data roads were 1, 3, 4, and 5, the number of neurons were 4, 5, 5, and 6, respectively, which achieved better results. Dia et al. [12] designed an object-oriented neural network model to predict short-term traffic conditions on the Pacific Expressway from Brisbane, Queensland, Australia to the Gold Coast. The input of the model included the average speed and traffic flow upstream and downstream of the expressway, and the output was the passing time of vehicles. With the development of convolutional neural networks (CNN) in recent years, FNN is generally not used to obtain spatiotemporal correlation, but as an output aggregation or external data-processing sub-module. Wu et al. [13] used FNN to combine the outputs of one CNN network and two long short-term memories (LSTM), and Panzhevi et al. [14] used FNN to learn road network features and add them to the model. However, the ability of convolutional neural networks to obtain time correlation is still insufficient.

Further studies have combined long-term and short-term memory with convolutional neural networks, to extract spatial correlation by CNN and temporal correlation by LSTM. Pu et al. [15] first used CNN to capture spatial features and added a residual neural unit to CNN to increase network depth, then used LSTM to capture temporal features, fusing the two parts to obtain traffic flow prediction results. Du et al. [16] used multi-layer LSTM and multi-layer CNN to process traffic flow data, and then fused the results obtained by the two sub-networks with a feedforward neural network. Yao et al. [17] first used CNN to extract the data of each time slice as a feature vector, and then input it into LSTM for time-series modeling. On this basis, they proposed a spatiotemporal dynamic network for prediction of taxi demand, which could dynamically learn the correlations between locations. Yu et al. [18] used a stacked autoencoder to encode traffic accident data, and LSTM to capture the time correlation of data. Shi et al. [19] proposed the CONVLSTM model, which replaces all operations of ordinary LSTM with convolution operations, so can be used to process matrix sequences. Yuan et al. [20] proposed a heterogeneous traffic-accident prediction framework based on CONVLSTM. The framework adds a variety of auxiliary data, and divides the target area into three types: urban, rural, and mixed, and trains different models of different regions for fusion, to process heterogeneous spatiotemporal data.

Because the traffic network is arranged in non-Euclidean space, but prediction models aim to use the convolution scheme that achieves excellent results in Euclidean space, the convolution for this kind of non-Euclidean space came into being. Figure 1 is a comparison of the differences between graph structure data and Euclidean spatial data. Convolution cannot be performed on graph structured data, but graph convolution can solve this problem.

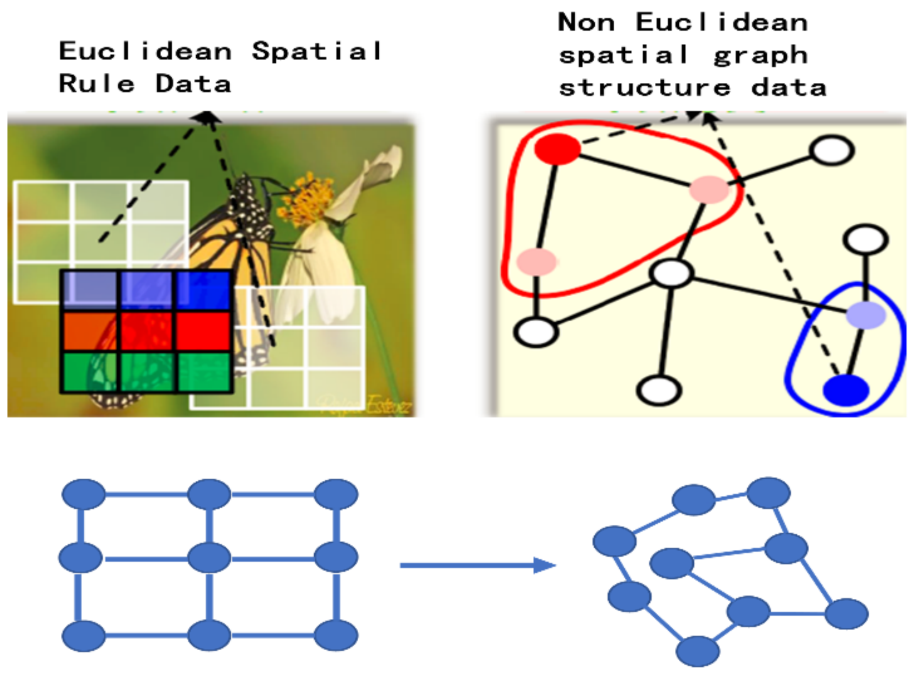

Euclidean Spatial Rule Data
Non Euclidean spatial graph structure data

Figure 1. Comparison between Euclidean and non-Euclidean spatial data.

Graph convolution is a scheme developed in recent years, which is suitable for convolution operation of non-Euclidean spatial data. The two mainstream methods of graph convolution are the spatial method and the spectral method. The spatial method performs convolution filters directly on the nodes of the graph and their neighbors. Therefore, the core of this method is to select the neighbors of nodes [21]. A heuristic linear method selects the neighborhood of each central node, which has achieved good results in social network tasks. Bruna et al. [22] (Figure 1) proposed a general graphical convolution framework (GCN) based on the Laplace matrix, and then Seo et al. [23] optimized the method by using Chebyshev polynomials instead of a convolution kernel (chebnet). In recent years, GNN has been widely used in computer vision [24,25], recommendation systems [26,27], traffic prediction, and other fields. Li et al. [28] proposed the DCRNN model, which uses bidirectional random walk on the graph and encoder–decoder architecture with planned sampling to obtain spatial correlation and temporal correlation, respectively. The STGCN model proposed by Yu et al. [29] uses a graph convolution network and Chebyshev network. Two graph neural networks and a one-dimensional CNN model spatial and temporal correlation, respectively, to form a spatiotemporal convolution module, and the final model is obtained by stacking two spatiotemporal convolution modules. Guo et al. [30] proposed a spatiotemporal attention mechanism, based on a graph convolution network and attention mechanism, to study the dynamic spatiotemporal correlation of traffic data. Spatial attention is used to capture the complex spatial correlation between different locations, and temporal attention is used to capture the dynamic correlation between different time slices. Geng et al. [31] indicated that STMGCN could calculate predictions of regional taxi demand. In their study, three graphs were constructed from the perspectives of connectivity between regions, location relationship between regions, and functional similarity between regions. The graph convolution network was used to extract the temporal correlation and spatial correlation from the three graph representations, and finally to calculate the average values to obtain prediction results. Zhong, W, et al. [32] proposed a heterogeneous spatiotemporal prediction framework for traffic prediction using incomplete historical data. This method uses a recurrent neural network to capture the time correlation and simultaneously estimate the missing values. At the same time, it models the dynamic correlation between road segments from aspects of their geography and history. It is also able to supplement the missing values. Siddiqi et al. [33] proposed a denoising automatic encoder to generate a clearer version of the input signal, and optimized the model by adjusting the superparameter. This was shown to be an effective method for supplementing traffic data with missing data. Joelianto et al. [34] proposed a data interpolation method based on spatiotemporal probabilistic principal component analysis (PPCA), which was more robust for the supplementation of extreme missing values and successfully accounted for incomplete feature capture caused by missing values. Liu et al. [35] proposed a model of FSTGCN for fusing vehicle speed features to further capture spatial features, while building a dynamic adjacency matrix of multiple graphs to better explore hidden spatial information.

There are several problems with the above methods: (1) Some only pay attention to the spatial correlation of traffic flow data and ignore the temporal correlation. (2) Others pay attention to the temporal correlation and ignore the spatial correlation. (3) In some methods, although the spatial and temporal correlation are calculated at the same time, the model for calculating the temporal correlation has a large number of parameters and requires large resource consumption, while the model for calculating spatial correlation pays too much attention to the high-frequency spatial feature information and ignores the low-frequency information.

To solve such problems, we propose several strategies for effectively building models to simulate the temporal dynamics and spatial correlation of traffic flows. We used graph convolution to capture the spatial features of traffic flow. To avoid the high weight impact resulting from high-frequency data features, an LPF was designed in the graph convolution. This can suppress high-frequency data features and strengthen the low-pass filter of low-frequency data features, and thus make the signals on the whole graph smoother and improve their robustness. To address the inherent defects of recurrent networks, C-GLU was used on the time axis to capture the features in time series, and LSTM was not used. In this way, the training was accelerated and the efficiency was improved. Finally, the two parts were combined to extract spatial and temporal features. We validated our model by using two real datasets. The experiment results show that our model is better than the five existing typical models.

2. Preparations

2.1. Traffic Networks

We defined the traffic flow network as a directionless graph $G_t = \{V_t, E, A\}$. As shown in Figure 2, G_t is the traffic network graph at time t, V_t is the set of $|V| = n$ nodes at time t, E is the set of sides, representing the connectivity between the nodes, $A \in \mathbb{R}^{n\times n}$ is the adjacency matrix, and $A_{i,j} = A_{j,i}$ defines the connectivity between node i and node j.

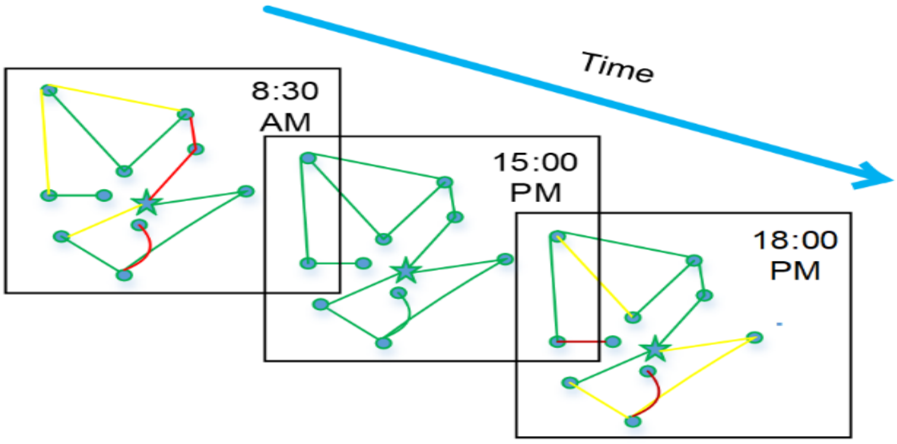


Figure 2. Composition g of traffic flow data in time and space. Red, yellow and green represent congestion, ordinary, and unblocked road conditions, respectively.

We used $x_t^{c,i} \in R$ to represent node i, the c_t feature value at time t, and $c_t \in (f_t, o_t, s_t)$, $x_t^i \in \mathbb{R}^T$ to represent the values of all the features of node i at time t. f_t represents the traffic flow features at time t in the traffic network graph G on time series. $X_t = \left(x_t^1, x_t^2, \ldots, x_t^n\right)^T \in \mathbb{R}^{N\times C}$ represents the values of all the features of all nodes at time t, and $X = (X_1, X_2, \ldots, X_\tau)^T \in \mathbb{R}^{\tau\times N\times C}$ represents the values of all the features of all nodes on the time slice. Finally, we set the traffic flow value $y_t^i = x_t^{f,i}$ of any node i at any time t in the future: G_{t+1}, G_t, G_{t-1}.

2.2. Graph Neural Networks

It is known that typical convolution has brought great success in natural language processing and image processing. However, it can only process data of fixed input dimensions (dimension consistency) and the local input data must be sequential (series sequence); i.e., it is only applicable to regular data generated within Euclidean space. Obviously, traffic networks are not regular data, so how can convolution operations be performed on non-Euclidean spatial data? There are two basic methods for the application of CNN to non-Euclidean data. One is to expand the spatial definition of convolution [21], and the other is based on spectral theory and graph Fourier transform [22]. The former involves re-arranging the vertexes into a grid form that can be processed by normal convolution operations. The latter introduces spectral theory and converts the data from the spatial domain to the spectral domain for processing, and then back to the spatial domain after processing. This is known as spectral graph convolution.

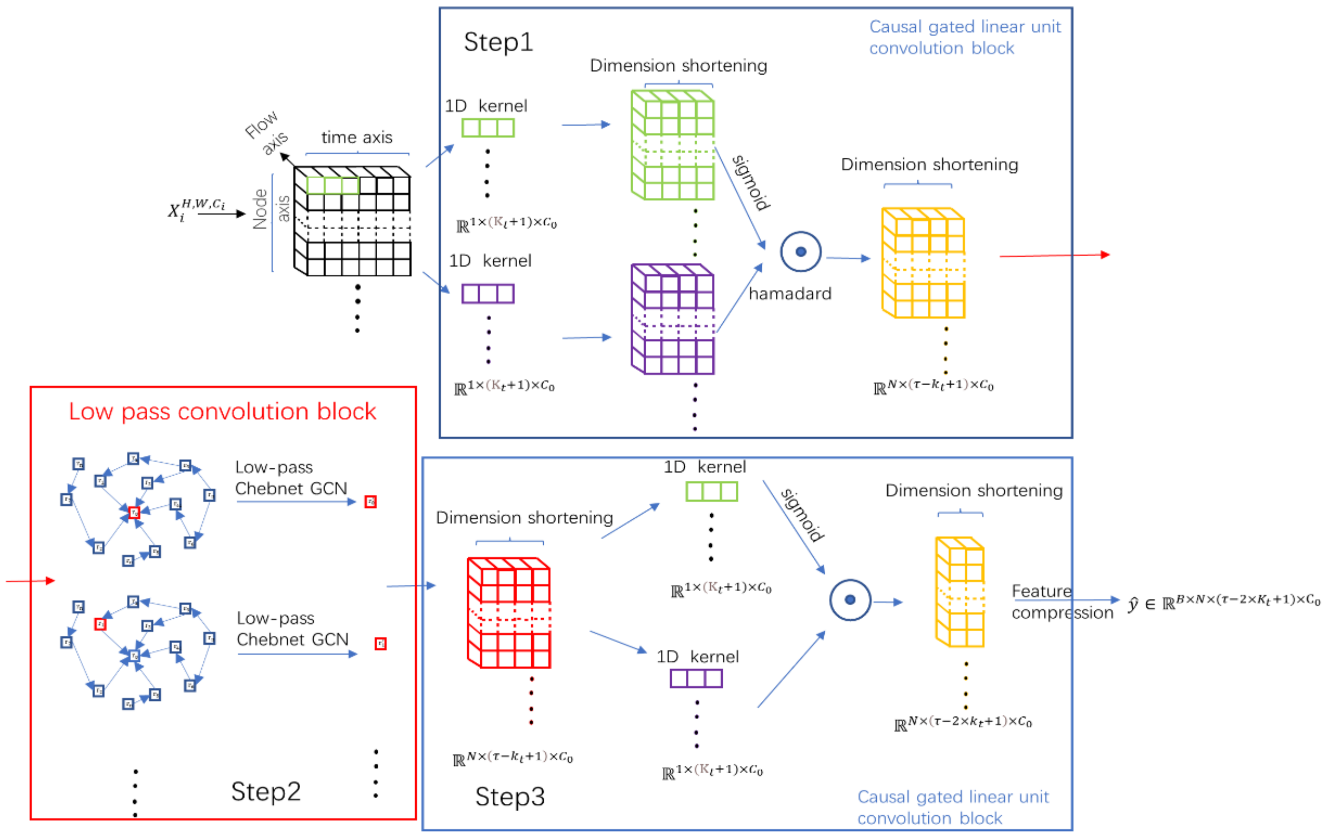
Our method is based on spectral graph convolution. The first graph convolution problem was how to define the Fourier transform on the graph, and the Laplacian matrix was introduced to fulfill the Fourier transform on the graph [22]. The graph Laplacian matrix is defined as $L = D - A$. $D \in \mathbb{R}^{N\times N}$ represents the diagonal matrix of the sum of the weights of all sides starting from node i. $D_{ii} = \sum jA_{i,j}$, the normalized Laplacian matrix, is defined as $L = I_n - D^{-1/2}AD^{-1/2}$. The features of the Laplacian matrix are decomposed into $L = U\Lambda U^T$, where $U = (\overline{u_1}, \ldots, \overline{u_n})$ is the feature vector of the Laplacian matrix. The diagonal matrix $\Lambda = diag([\lambda_1, \ldots, \lambda_n]) \in \mathbb{R}^{N\times N}$ is composed of the feature values of the Laplacian matrix. Take the flow at time t as an example. The flow signal on the graph is defined as $x = x_t^f \in \mathbb{R}^N$. Through the graph Fourier transform, the flow signal is transformed from the spatial domain to the spectral domain, which is defined as $\hat{x} = U^T x \in \mathbb{R}^N = (\hat{x}(\lambda_1), \ldots, \hat{x}(\lambda_n))$. Finally, the signal must be returned from the spectral domain to the spatial domain, because U is an orthogonal matrix. Therefore, the graph inverse Fourier transform is defined as $x = U\hat{x}$. Graph convolution is a convolutional operation performed by using diagonal Laplacian operators in the graph Fourier transform, in place of the operators employed in conventional convolution [36]. The signal x on the traffic network graph G is filtered through the kernel $g_\theta = diag(U^T g)$, where $U^T g = \hat{g} = (\hat{g}(\lambda_1), \ldots, \hat{g}(\lambda_n))$:

$$x *_G g_\theta = U\left(U^T x \odot U^T g\right) = U g_\theta(\Lambda) U^T x \tag{1}$$

3. Model Framework

3.1. Network Structure

The overall framework of the CGLGCN model proposed in this paper is illustrated in Figure 3. It consists of a causal gated–LPF convolution block. Each causal gated LPF convolution block consists of two C-GLU temporal convolution blocks plus a LPF convolution block, forming a “sandwich” structure as shown in the figure. The causal gated linear unit is responsible for capturing temporal features, and low-pass graph convolution is responsible for capturing spatial features.



Step1
Causal gated linear unit convolution block
Dimension shortening
Flow axis
time axis
X_i^{H,W,C_i}
Node axis
1D kernel
$\mathbb{R}^{1\times(K_t+1)\times C_0}$
sigmoid
hamadard
Dimension shortening
$\mathbb{R}^{N\times(\tau-k_t+1)\times C_0}$
Low pass convolution block
Low-pass Chebnet GCN
Low-pass Chebnet GCN
Step2
Dimension shortening
1D kernel
sigmoid
Dimension shortening
Feature compression
$\hat{y} \in \mathbb{R}^{B\times N\times(\tau-2\times K_t+1)\times C_0}$
$\mathbb{R}^{N\times(\tau-k_t+1)\times C_0}$
$\mathbb{R}^{1\times(K_t+1)\times C_0}$
$\mathbb{R}^{N\times(\tau-2\times k_t+1)\times C_0}$
Step3
Causal gated linear unit convolution block

Figure 3. Causal gated low-pass filter graph convolutional neural network model. The framework CGLGCN is composed of a causal gated low-pass convolution block and a fully connected output layer. Each causal gated low-pass convolution block is composed of two causal gated linear unit convolution blocks sandwiching a low-pass graph convolution block.

3.2. LPF Convolution Module

Usually, a traffic network will produce an irregular graph; even if junctions are deemed as the vertex, roads as sides, and the road network as a graph of points and sides, it cannot be deemed as a grid. This is because the distances between junctions are different and the spatial relationship between the junctions would be ignored if the road network were deemed as a grid. The spatial domain information captured by such convolution is not reasonable. Therefore, we introduced graph convolution and defined convolution directly on the graph data to capture the features that impacted different weights on junctions. The graph convolution operations in Formula (1) require the decomposition of Laplacian matrix features. The calculation complexity is $O\left(n^2\right)$, and great challenges can be encountered in cases of large-scale graph data caused by large numbers of nodes. Therefore, we used Chebyshev polynomials as the convolutional kernel and effectively solved this problem.

Using Chebyshev polynomial approximation, the kernel g_θ can be limited to the polynomial of Λ in order to localize the filter and reduce the number of parameters. For example $g_\theta(\Lambda) = \sum_{k=0}^{k=k-1} \theta_k \Lambda^k$, where the kernel is replaced with Chebyshev polynomial approximation, can be approximately considered as $g_\theta(\Lambda) = \sum_{k=0}^{k=k-1} \theta_k T_k\left(\hat{\Lambda}\right)$; after Chebyshev polynomials are introduced, the graph convolution is changed into:

$$x *_G g_\theta = U g_\theta(\Lambda) U^T x = U \sum_{k=0}^{k=k-1} \theta_k T_k\left(\hat{\Lambda}\right) U^T x = \sum_{k=0}^{k=k-1} \theta_k T_k\left(\hat{L}\right) x \tag{2}$$

where $\theta \in R^k$ is the polynomial coefficient, k is the size of the graph convolution kernel which determines the number of node orders covered by the target node convolution $\hat{L} = \frac{2}{\lambda_{max} - 2} L - I_n$, and λ_{max} is the maximum feature value of the Laplacian matrix.

Chebyshev graph convolution is a filter in which the weight of high-frequency signal increases with the rising k value.

$$g\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k \tag{3}$$

As the weight of high-frequency features increases, it interferes with researchers' ability to capture low-frequency features, which in turn affects the prediction accuracy. To solve this problem, we propose a LPF to replace it, which can increase the weight of low-frequency signals and suppress high-frequency signals. The data processed by LPF convolution are smoother, and the interference of high-frequency noise can be avoided.

From the signal x on the graph, we can obtain:

$$x = \alpha_1 \overline{u_1} +, \ldots, + \alpha_n \overline{u_n} \tag{4}$$

Because the Laplacian matrix is a symmetric positive semi-definite matrix, we can obtain:

$$\overline{u_i u_i}^T = 1\ ,\ \overline{u_i u_j}^T = 0,\ \ i \neq j \tag{5}$$

In formula (1) $U g_\theta(\Lambda) U^T x = \left(\theta_1 \overline{u_1 u_1}^T +, \ldots, + \theta_n \overline{u_n u_n}^T\right) x$, and we can obtain:

$$\overline{u_n u_n}^T x = \alpha_1 \overline{u_1 u_1}^T \overline{u_1} +, \ldots, + \alpha_n \overline{u_n u_n}^T \overline{u_n} = \alpha_n \overline{u_n} \tag{6}$$

Therefore, $\left\{\overline{u_n u_n}^T\right\}_{n=1}^{n}$ can be considered a set of basic filters in graph convolution, and allowing only the passage of feature values associated with $\overline{u_n}$. α_n is equivalent to the weight assigned to the filters. Polynomial approximation graph convolution includes a set of combined filters $\left\{\hat{L}^k\right\}_{k=0}^{k-1}$. So, for the basic filters of polynomial approximation graph convolution, the basic filter of high-frequency signals has a greater weight, $\alpha_n = {\lambda_n}^k$. We therefore considered that an LPF could be designed to replace the previous filters.

Our LPF is defined as:

$$f(\lambda_i) = 1 - \frac{1}{2} \lambda_i \tag{7}$$

We bring it into the Laplacian matrix and replace Λ with Λ_s. $\Lambda_s = diag\left(\left\{1 - \frac{1}{2}\lambda_i\right\}_{i=1}^{n}\right)$, so we define the kernel of the graph convolution as:

$$g_\theta = \sum_{k=0}^{k-1} \theta_k (\Lambda_s)^k \tag{8}$$

Then, our LPF convolution can yield:

$$\begin{aligned} x *_G g_\theta &= U \sum_{k=0}^{k-1} \theta_k (\Lambda_s)^k U^T x \\ &= \left\{\theta_0 I + \theta_1 \left(\left(1 - \frac{1}{2}\lambda_1\right)\overline{u_1 u_1}^T + \ldots \left(1 - \frac{1}{2}\lambda_n\right)\overline{u_n u_n}^T\right) + \ldots \right. \\ &\left. + \theta_{k-1}\left(\left(1 - \frac{1}{2}{\lambda_1}^{k-1}\right)\overline{u_1 u_1}^T + \ldots + \left(1 - \frac{1}{2}\lambda_n\right)^{k-1}\overline{u_n u_n}^T\right)\right\} x \\ &= \left\{\theta_0 I + \theta_1\left(I - \frac{1}{2}L\right) + \ldots + \theta_{k-1}\left(I - \frac{1}{2}L\right)^{k-1}\right\} x \end{aligned} \tag{9}$$

Through LPF convolution, a higher weight can be assigned to the basic filter of the low-frequency signal, but the features of the high-frequency signal are not given up. Thus, we can obtain better flow features on the traffic network graph without missing the flow features of the high-frequency signal.

3.3. Causal Gated Linear Unit (C-GLU)

Although RNN-based models are widely used in time series analysis, recurrent networks used for flow prediction have disadvantages including long iteration time, complex gate mechanisms, and slow response to dynamic changes. In contrast, CNNs show such advantages as fast training, simple structure, and independence from the previous step [37]. We performed convolution on the time axis to capture the feature relationships of traffic flow in the time dimension. Causal convolution can simply and effectively capture long-term dependencies, so we used causal convolution to capture the feature connections of the time dimension. Meanwhile, in order to alleviate the problem of gradient disappearance, we used a residual connection gating mechanism of to enable the causal convolution to learn the features relationship for longer.

As shown in Figure 3, the first step of the causal gated convolution block is to perform one-dimensional causal convolutions of data. Each convolution without filling explores the k_t order neighborhood of each node on the traffic network graph G, and each convolution shortens the time series of $k_t - 1$. So, the i node data can be considered as the data $Y \in \mathbb{R}^{N \times \tau \times C_i}$ with a slice time series length of τ and the number of channels C_i, and the convolution kernel can be defined as $\Gamma \in \mathbb{R}^{1 \times C_i \times 2C_0}$. In the first convolution, one-dimensional causal convolution is performed; in the second convolution, nonlinear change is performed after the one-dimensional causal convolution is completed. The data of the two parts are the two gates of the causal gated linear unit, and the two data outputs are recorded as $[P, Q] \in \mathbb{R}^{N \times (M - k_t + 1) \times 2C_0}$. Finally, the data of the two parts are recorded with the Hadamard product. Therefore, the convolution of the causal gated linear unit can be defined as:

$$\Gamma *_f Y = P \odot sigmoid(Q) \in \mathbb{R}^{N \times (\tau - k_t + 1) \times C_0} \tag{10}$$

3.4. Fusion of Causal Gated-LPF Convolutions

Because the input traffic flow data are recorded as $x_t \in \mathbb{R}^{\tau \times N \times C_t}$ after time slicing, and then recorded as $x_{t+1} \in \mathbb{R}^{(\tau - k_t + 1) \times N \times C_t}$ after the temporal and spatial features are extracted by the causal gated graph convolution module, the causal gated graph convolution and the LPF convolution are fused and recorded as:

$$out = Relu\left(\Gamma_1 *_f \left(Relu\left(\left(\Gamma_0 *_f x_t\right) *_G g_\theta\right)\right)\right) \tag{11}$$

Γ_0 is the convolutional kernel of the first C-GLU convolution module. Γ_1 is the convolutional kernel of the second C-GLU convolution module. g_θ is the convolutional kernel of the intermediate LPF convolution module, and the rectified linear unit (RELU) is the activation function of the graph convolution module and the full connection layer at the final output. $*_f$ represents the causal gated convolution operation, $*_G$ represents the graph convolution operation.

4. Experiment

4.1. Data

We validated our model on two highway traffic datasets, PeMS04 and PeMS08, collected in California. Both datasets were collected by the Caltrans Performance Measurement System (PeMS) in real time every 30 s, and the traffic flow data were aggregated once every 5 min. The system has more than 45,514 detectors deployed on highways in the major metropolitan areas of California. The datasets involve three kinds of traffic measurements: flow, occupancy, and speed. We selected the data from Californian areas 4 and 8 as the dataset for this study.

PeMS04 comprises traffic data from the San Francisco Bay Area, collected by 3848 detectors on 29 roads in this area over 59 days from January to February 2018. The data were collected every 30 s and aggregated every 5 min. We selected the data from the first 45 days as the training set, and the remaining data as the test set.

PeMS08 is traffic data from San Bernardino, collected by 1979 detectors on 8 streets in this area over 62 days from July to August 2016. The data were collected every 30 s and aggregated every 5 min. We selected the data from the first 48 days as the training set, and the remaining data as the test set.

As shown in Figure 4, through visual processing of the two data sets, we found that the traffic flow is the most direct indicator of intersection congestion, and that change of traffic flow can help us to predict congestion.

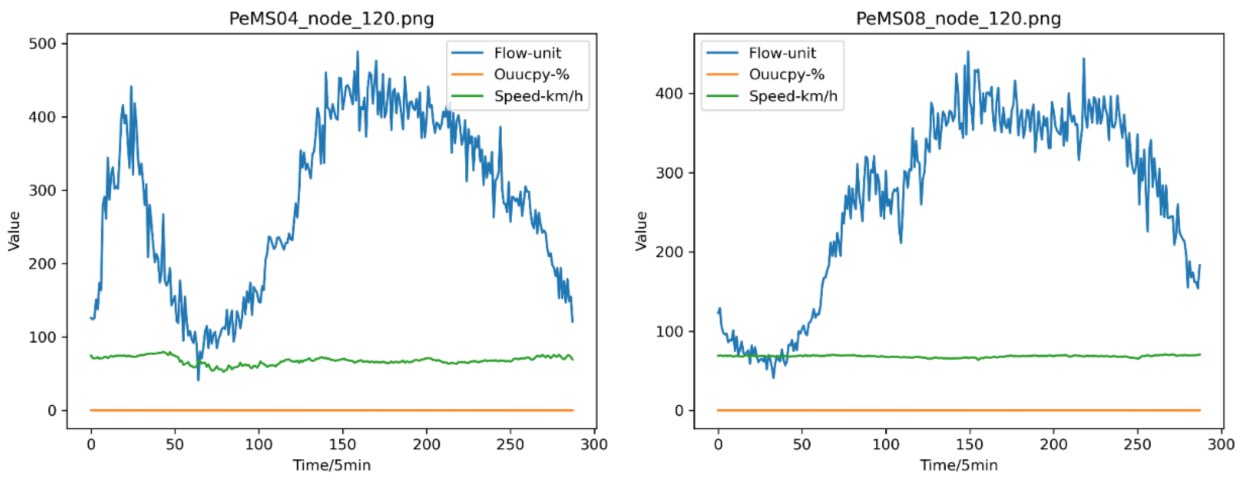


Figure 4. Visual comparison of data fluctuation of three elements of 24-h traffic flow at intersection 120 in the randomly selected data set.

4.2. Data Processing and Parameter Setting

We removed redundant detectors to ensure that each detector represented one junction and that the distance between detectors was not less than 3.5 miles. Finally, there were 307 detectors in the PeMS04 set and 170 detectors in PeMS08. The detectors aggregated the traffic flow data at junctions every 5 min, and so the data of 288 points were collected for each junction every day. The missing values were filled by linear interpolation. In addition, the data were processed by zero-mean normalization to bring the average to 0.

In the experiment, we tested our model by using the pytorch framework. We tested order $k \in [0, 1, 2]$ of the LPF convolution module and achieved satisfactory results. Experimentation revealed that the optimal results were obtained for $k = 2$. Meanwhile, the kernel size of the causal gated convolution was set to 3. In terms of data, we adopted the concept of "sliding window" to divide the data. In the LPF convolution module, the space axis was convoluted as per the time axis slice. The "sliding window" was employed to predict the traffic flow of the $\tau + 1$-th time point by using the traffic flow of the previous $0 \sim \tau$ time points. The $\tau + 1$ data constituted a training set sample, the τ data were input x and the $\tau + 1$ data were labelled $\hat{y}$. To obtain the next sample, we moved the "sliding window" to the next time point where the traffic flows of the $1 \sim \tau + 1$ time points constituted the new input data x, and the traffic flow of the $\tau + 2$-th time point became the new label $\hat{y}$. These data and labels formed the training set and the test set. The "sliding window" used the average value of multiple data for prediction, avoiding outliers and differences caused by single data. In this way, the impact on the prediction results was minimized and our data were more stable. In this study, the size of our "sliding window" was set to $\tau = 12$. For the prediction range, the list of continuous time nodes may be selected at will; the prediction ranges in this study were set to 5 min, 15 min, 30 min, and 45 min, including two time marker points for short-term prediction and long-term prediction, respectively.

Mean absolute error (MAE) and root mean square error (RMSE) were used as metrics and baselines for measuring and evaluating the performance of different methods in the experiment. We compared our method with the following typical methods: (1) historical average (HA); (2) auto-regressive integrated moving average (ARIMA); (3) Vector auto-regressive (VAR); (4) long short-term memory (LSTM) network; and (5) gated recurrent unit (GRU) network.

4.3. Analysis and Comparison of Results

Table 1 shows that our CGLGCN performed best for both datasets in terms of all metrics. It can be seen that the prediction results of conventional time series analysis methods were usually not ideal. This proves that the capabilities of such methods for model nonlinear and complex flow data are suboptimal. Generally, methods based on deep learning achieved better prediction results than conventional time series analysis methods. Our model considered temporal correlation and spatial correlation simultaneously, and so achieved better prediction results than LSTM and GRU models that considered only temporal correlation. Our model thus further reduced prediction errors.

Table 1. Performance comparison of different methods applied to PeMS04 and PeMS08.

Model	PeMS04 (5/15/30/45 min)		Model	PeMS08 (5/15/30/45 min)	
	MAE	RMSE		MAE	RMSE
HA	35.12/35.75/36.89/37.25	41.25/44.97/52.87/54.89	HA	26.98/27.77/28.45/29.12	32.87/35.31/42.08/47.29
ARIMA	30.06/30.56/31.95/32.63	33.12/43.98/59.87/65.73	ARIMA	21.43/22.02/22.98/23.87	22.47/29.15/39.21/48.34
VAR	30.58/31.72/32.69/33.52	46.29/49.92/52.87/54.89	VAR	19.19/19.56/20.35/21.12	22.12/26.93/31.45/35.91
LSTM	27.57/28.03/28.65/29.15	34.05/37.72/44.89/47.01	LSTM	21.41/21.85/22.45/23.21	21.32/27.73/36.78/42.51
GRU	26.41/27.08/28.23/29.12	34.12/37.59/45.12/46.85	GRU	19.52/19.98/20.82/22.46	21.39/26.69/37.83/43.28
CGGCN	23.15/24.08/25.23/26.12	32.12/33.49/35.15/37.90	CGGCN	17.52/17.98/18.84/20.46	21.15/22.49/24.15/25.85
CGLGCN	22.12/22.94/23.51/24.24	31.23/32.45/34.05/35.91	CGLGCN	16.41/170.1/17.74/18.84	20.28/21.87/23.29/24.89

Figures 5 and 6 show the varying performance of different methods when the prediction was carried out for a longer period. In general, the models' prediction precision became lower when the prediction spanned a longer period. The models which only considered the temporal correlation, including ARIMA, LSTM, and GRU, achieved almost the same quality of results as our model in the 5-min short-term traffic flow prediction, but their performance declined sharply when the prediction time became longer. The VAR model considering both temporal and spatial correlations performed stably. However, it showed significant differences for different sample scales, and its performance badly declined with the expansion of the road network scale. Although the VAR model considers temporal and spatial correlations, it is not applicable to traffic flow prediction for large-scale road networks, and real-world road networks are usually huge. Our CGLGCN model performed well in the test. It was shown to be competent for the prediction task no matter whether the road network was large or small, and always delivered adequately stable performance even when the prediction spanned a longer period.

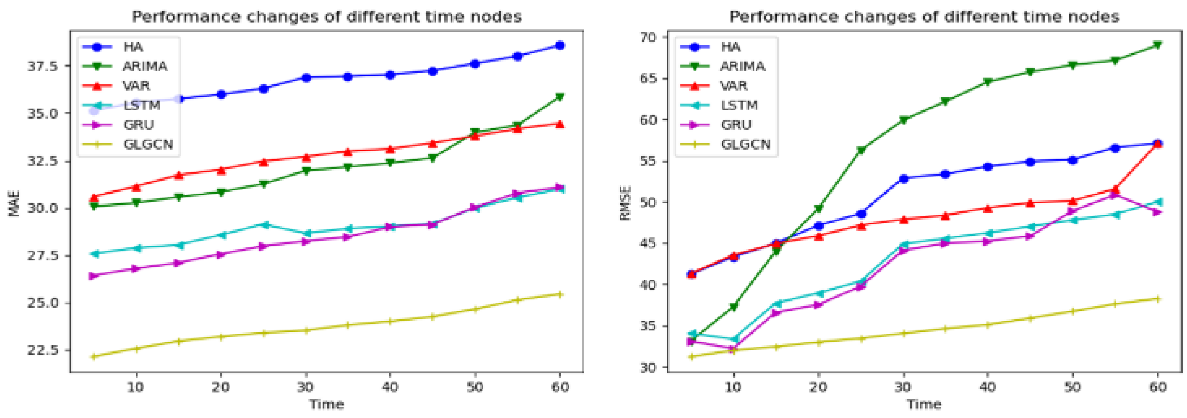


Figure 5. Performance comparison of different models over time for the PeMS04 dataset.

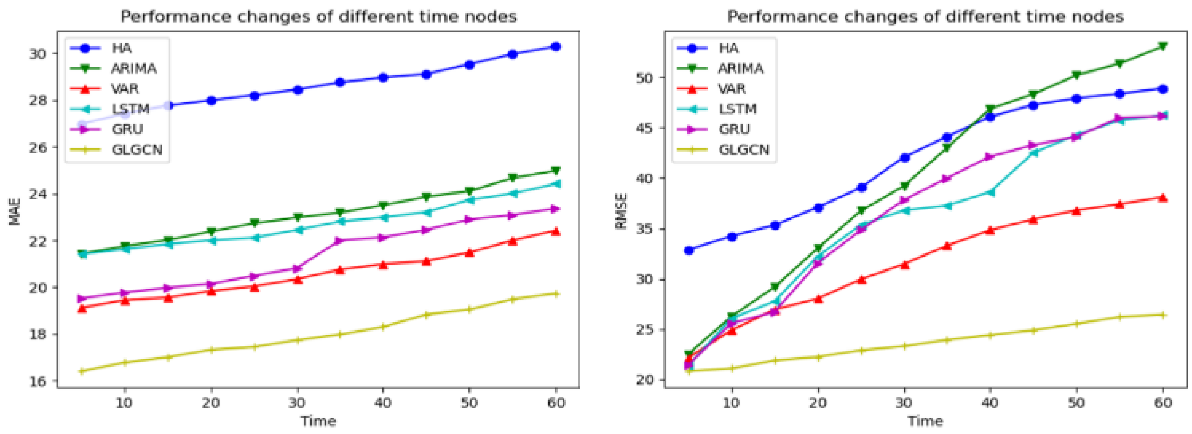


Figure 6. Performance comparison of different models over time for the PeMS08 dataset.

As shown in Figures 7 and 8, we performed ablation experiments on CGLGCN on pems04 and pems08 datasets, respectively. The model that does not use our low-pass filtering constraints is labelled CGGCN. Experiments demonstrated that our model using a low-pass filter constraint had better performance, which proves the feasibility of our low-pass filter constraint.

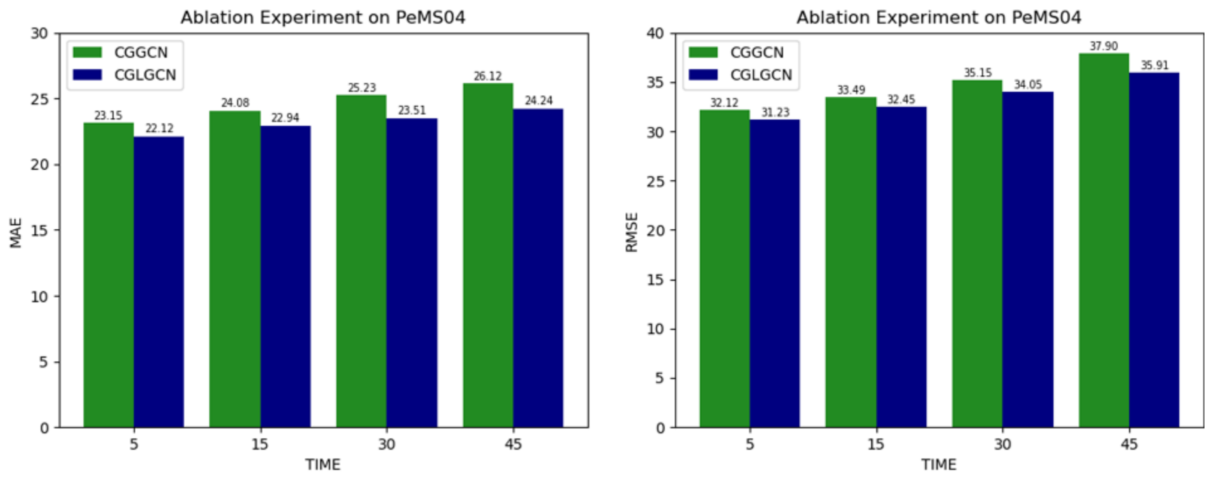


Figure 7. Performance comparison of ablation experiment for the pems04 dataset.

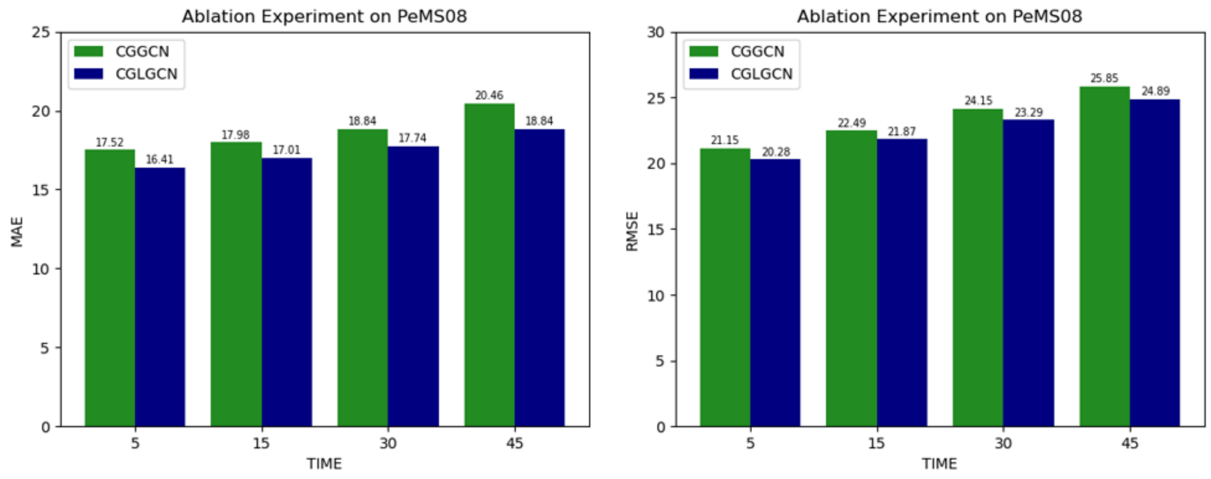


Figure 8. Performance comparison of ablation experiment for the pems08 dataset.

Our model abandons the LSTM and GRU models that use multiple parameters and involve considerable time consumption, and uses gated linear units to capture the features of time dimension, which greatly reduces the training time. As shown in Figure 9 and Table 2, our CGLGCN model took only 2579.79 s and 1279.83 s for 1000 rounds of training in pems04 and pems08 datasets, respectively, while the LSTM model took 21,844.54 s and 17,973.62 s, and the GRU model 10,000.15 s and 7589.26 s. These substantial increases in training speed (more than eight-fold and approximately seven-fold, respectively) were mainly due to the fact that recursive structures were not used. Our method can realize full parallel training rather than relying entirely on a chain structure.

Table 2. Comparison of training consumption time.

Dataset	**1000 Epoch Time Consumption (s)**		
	LSTM	**GRU**	**CGLGCN**
PeMS04	21,844.54	17,973.62	2579.79
PeMS08	10,000.15	7589.26	1279.83

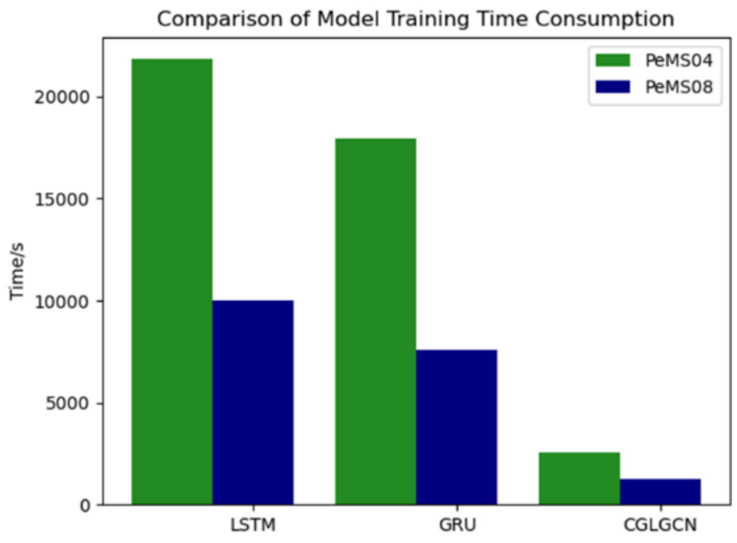


Figure 9. Visualization of training time comparison.

Figures 5 and 6 show that our model performed stably in short-term and long-term predictions. To validate the stability of our model, we tried to predict 24-h traffic flow. The scale of the x-coordinate in Figure 10 represents the flow prediction for a total of 288 time points collected every 5 min from 0:00 to 24:00 on a single day. It can be seen from the figure that the prediction generated by our model for traffic flow changes throughout the day well matched the trend of traffic flow changes. Our model was able to predict accurately the traffic flow in each period at each junction. This fully demonstrates the stability of our model.

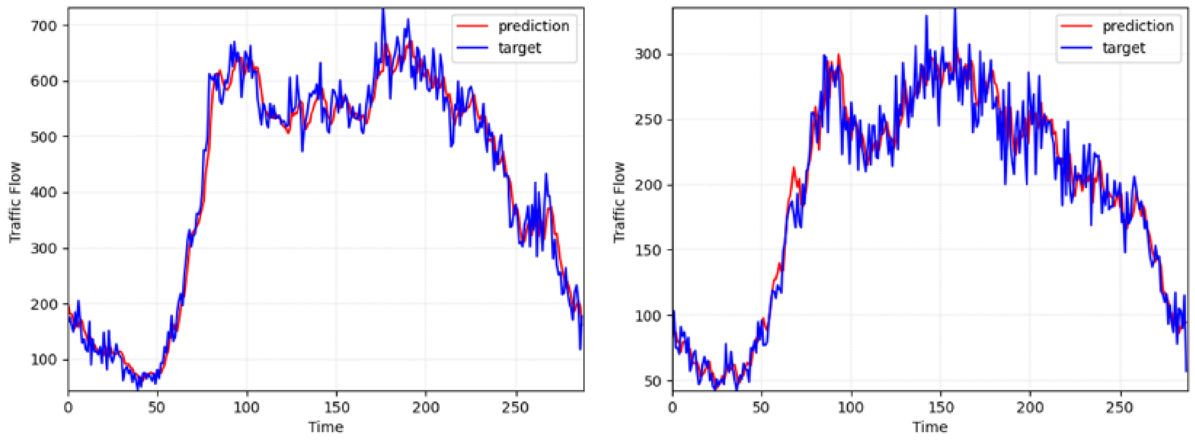


Figure 10. (**left**) the fitting map of traffic flow prediction from 00:00 a.m. to 24:00 p.m. for detector 120 in the PeMS04 dataset; (**right**) is the fitting map of traffic flow prediction from 00:00 a.m. to 24:00 p.m. for detector 120 in the PeMS08 dataset.

5. Conclusions and Outlook

This paper proposes a causal gating low-pass graph convolution neural network for traffic flow prediction, called CGLGCN, and has demonstrated that the causal gating linear unit has a good effect in capturing time features. On the premise that the prediction accuracy is not greatly impacted, the method reduces the amount of parameters and time of operation. Because the traffic network graph is irregularly distributed, low-pass graph convolution can better capture the spatial correlation of non-Euclidean data, which is consistent with the definition of the graph and can reduce the impact of high-frequency noise. High-frequency noise would shift the focus points during network model training and reduce the capacity to learn the spatial features of traffic data. Experiments on two real-world data sets show that our model performed better than five existing models.

Our model does not consider various unexpected events, such as bad weather or traffic accidents, and their effects on real traffic flow prediction. Meanwhile, no weight was assigned to the importance of different junctions. The weights of junctions in core areas and more remote areas should not be equal, and even junctions that are not directly connected may have hidden correlations. In the future, more factors will be considered to further improve our model. In addition, we may also consider using our model in relation to 5G traffic distribution, to reduce the energy loss of 5G base stations.

Author Contributions: Writing—review and editing, H.M.; supervision, Y.Z.; project administration, X.L.; funding acquisition, X.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key Research and Development Program of China (No. 2019YFE0126100), the Key Research and Development Program in Zhejiang Province of China (No. 2019C54005).

Data Availability Statement: The data will be released on https://github.com/peakdemo/CGLGCN (accessed on 1 August 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Williams, B.M.; Hoel, L.A. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *J. Transp. Eng.* **2003**, *129*, 664–672. [CrossRef]
2. Okutani, I.; Stephanedes, Y.J. Dynamic prediction of traffic volume through Kalman filtering theory. *Transp. Res. Part B Methodol.* **1984**, *18*, 1–11. [CrossRef]
3. Kuchipudi, C.M.; Chien, S.I.J. Development of a hybrid model for dynamic travel-time prediction. *Transp. Res. Rec.* **2003**, *1855*, 22–31. [CrossRef]
4. Sun, H.; Liu, H.; Xiao, H.; He, R.R.; Ran, B. Use of Local Linear Regression Model for Short-Term Traffic Forecasting. *Transp. Res. Rec. J. Transp. Res. Board* **2003**, *1836*, 143–150. [CrossRef]
5. Zheng, Y.; Lai, W. High speed short-term traffic flow prediction based on support vector machine. *Eng. Constr.* **2020**, *34*, 201–204.
6. Wu, C.H.; Ho, J.M.; Lee, D.T. Travel-time prediction with support vector regression. *IEEE Trans. Intell. Transp. Syst.* **2004**, *5*, 276–281. [CrossRef]
7. Zhenhua, M.; Kepeng, L.; Dongfu, S. Short-term traffic flow prediction based on wavelet denoising and Bayesian neural network joint model. *J. Sci. Technol. Eng.* **2020**, *20*, 13881–13886.
8. Xu, X.; Bai, Y.; Xu, L. Traffic flow prediction based on Stochastic Forest algorithm in bad weather. *J. Shaanxi Norm. Univ. Nat. Sci. Ed.* **2020**, *48*, 25–31.
9. Ding, C.; Wang, D.; Ma, X.; Li, H. Predicting Short-Term Subway Ridership and Prioritizing Its Influential Factors Using Gradient Boosting Decision Trees. *Sustainability* **2016**, *8*, 1100. [CrossRef]
10. Wang, D.; Zhang, Q.; Wu, S.; Li, X.; Wang, R. Traffic flow forecast with urban transport network. In Proceedings of the 2016 IEEE International Conference on Intelligent Transportation Engineering (ICITE), Singapore, 20–22 August 2016; pp. 139–143.
11. Park, D.; Rilett, L.R. Forecasting freeway link travel times with a multilayer feedforward neural network. *Comput.-Aided Civ. Infrastruct. Eng.* **1999**, *14*, 357–367. [CrossRef]
12. Dia, H. An object-oriented neural network approach to short-term traffic forecasting. *Eur. J. Oper. Res.* **2001**, *131*, 253–261. [CrossRef]
13. Wu, Y.; Tan, H. Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *arXiv* **2016**, arXiv:1612.01022.
14. Pan, Z.; Liang, Y.; Wang, W.; Yu, Y.; Zheng, Y.; Zhang, J. Urban traffic prediction from spatio-temporal data using deep meta learning. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data, Anchorage, AK, USA, 4–8 August 2019; pp. 1720–1730.
15. Pu, Y.; Wang, W.; Zhu, Q.; Chen, P. Urban short-term traffic flow prediction algorithm based on CNN RESNET LSTM model. *J. Beijing Univ. Posts Telecommun.* **2020**, *43*, 9.
16. Du, S.; Li, T.; Gong, X.; Yang, Y.; Horng, S.J. Traffic flow forecasting based on hybrid deep learning framework. In Proceedings of the 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), Nanjing, China, 24–26 November 2017; pp. 1–6.
17. Yao, H.; Wu, F.; Ke, J.; Tang, X.; Jia, Y.; Lu, S.; Gong, P.; Ye, J.; Li, Z. Deep multi-view spatial-temporal network for taxi demand prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; p. 32.

18. Yu, R.; Li, Y.; Shahabi, C.; Demiryurek, U. Deep learning: A generic approach for extreme condition traffic forecasting. In Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, TX, USA, 27–29 April 2017; pp. 777–785.
19. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.-Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; p. 28.
20. Yuan, Z.; Zhou, X.; Yang, T. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 984–992.
21. Niepert, M.; Ahmed, M.; Kutzkov, K. Learning convolutional neural networks for graphs. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 2014–2023.
22. Bruna, J.; Zaremba, W.; Szlam, A.; LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv* **2013**, arXiv:1312.6203.
23. Seo, Y.; Defferrard, M.; Vandergheynst, P.; Bresson, X. Structured sequence modeling with graph convolutional recurrent networks. In Proceedings of the International Conference on Neural Information Processing, Siem Reap, Cambodia, 13–16 December 2018; Springer: Cham, Switzerland, 2018; pp. 362–373.
24. Qi, X.; Liao, R.; Jia, J.; Fidler, S.; Urtasun, R. 3d graph neural networks for rgbd semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5199–5208.
25. Liu, Y.; Fan, B.; Xiang, S.; Pan, C. Relation-shape convolutional neural network for point cloud analysis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 8895–8904.
26. Yin, R.; Li, K.; Zhang, G.; Lu, J. A deeper graph neural network for recommender systems. *Knowl.-Based Syst.* **2019**, *185*, 105020. [CrossRef]
27. Hell, F.; Taha, Y.; Hinz, G.; Heibei, S.; Müller, H.; Knoll, A. Graph Convolutional Neural Network for a Pharmacy Cross-Selling Recommender System. *Information* **2020**, *11*, 525. [CrossRef]
28. Li, Y.; Yu, R.; Shahabi, C.; Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv* **2017**, arXiv:1707.01926.
29. Yu, B.; Yin, H.; Zhu, Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv* **2018**, arXiv:1709.04875.
30. Guo, S.; Lin, Y.; Feng, N.; Song, C.; Wan, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 922–929.
31. Geng, X.; Li, Y.; Wang, L.; Yang, Q.; Ye, J.; Liu, Y. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 3656–3663.
32. Zhong, W.; Suo, Q.; Jia, X.; Zhang, A.; Su, L. Heterogeneous Spatio-Temporal Graph Convolution Network for Traffic Forecasting with Missing Values. In Proceedings of the 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), Washington, DC, USA, 7–10 July 2021; pp. 707–717.
33. Siddiqi, M.D.; Jiang, B.; Asadi, R.; Regan, A. Hyperparameter Tuning to Optimize Implementations of Denoising Autoencoders for Imputation of Missing Spatio-temporal Data. *Procedia Comput. Sci.* **2021**, *184*, 107–114. [CrossRef]
34. Joelianto, E.; Fathurrahman, M.F.; Sutarto, H.Y.; Semanjski, I.; Putri, A.; Gautama, S. Analysis of Spatiotemporal Data Imputation Methods for Traffic Flow Data in Urban Networks. *ISPRS Int. J. Geo-Inf.* **2022**, *11*, 310. [CrossRef]
35. Liu, D.; Xu, X.; Xu, W.; Zhu, B. Graph Convolutional Network: Traffic Speed Prediction Fused with Traffic Flow Data. *Sensors* **2021**, *21*, 6402. [CrossRef] [PubMed]
36. Henaff, M.; Bruna, J.; LeCun, Y. Deep convolutional networks on graph-structured data. *arXiv* **2015**, arXiv:1506.05163.
37. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional sequence to sequence learning. In Proceedings of the International Conference on Machine Learning, Ningbo, China, 9–12 July 2017; pp. 1243–1252.