Graph Neural Network Based Spatial-Temporal Traffic Flow Prediction Approaches

Zhi Chen

A thesis submitted to

Auckland University of Technology

in partial fulfilment of the requirements for the degree

of

Master of Computer and Information Sciences (MCIS)

2022

School of Engineering, Computer and Mathematical Sciences

Table of Contents

List of Figures

List of Tables

Glossary

ASTGCN: Attention Based Spatial-Temporal Graph Convolutional Networks

ARMA: Autoregressive Moving Average

ARIMA: Autoregressive Integrated Moving Average

BP: Back Propagation

CNN: Convolutional Neural Network

DCRNN: Diffusion Convolutional Recurrent Neural Network

EEMD: Ensemble Empirical Mode Decomposition

EMD Empirical Model Decomposition

GAT: Graph Attention Network

GCN: Graph Convolutional Network

GLU: Gated Linear Units

GNN: Graph Neural Network

GRU: Gate Recurrent Unit

GSVM: Gray Support Vector Machine

HA: History Average

IoT: Internet of Things

ITS: Intelligent Transportation System

KNN: K-Nearest Neighbours

LSTM: Long-Short Term Memory

MAE: Mean Absolute Error

MAPE: Mean Absolute Percentage Error

ML: Machine Learning

MLP: Multilayer Perceptron

MRA-BGCN: Multi-Range Attention Two-component GCN

MSTGCN: Multi-Component Spatial-Temporal Graph Convolution Networks

MST-GCRN: Multiple Layers Spatial-temporal Aware Graph Convolutional Recurrent Network Model

MST-GCTN: Multiple Layers Spatial-temporal Aware Graph Convolutional and Temporal Convolutional Network Model

MST-AGCRN: Attention Mechanism based MST-AGCRN

MST-AGCTN: Attention Mechanism based MST-AGCTN

OGCRNN: Optimized Graph Convolution Recurrent Neural Network

PeMS: Performance Measurement System

POI: Point Of Interest

ResNet: Residual Network

RMSE: Root Mean Squared Error

RNN: Recurrent Neural Network

STDM: Spatiotemporal Data Mining

STSGCN: Spatial-Temporal Synchronization Graph Convolutional Network

SVM: Support Vector Machine

TCN: Temporal Convolutional Network

TGC: Traffic Graph Convolution

VAR: Vector Autoregressive

VDS: Vehicle Detection Station

Attestation of Authorship

"I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning."

Signature of Candidate: ……

Acknowledgements

One year of research time passed in a flash, and the postgraduate career was about to end in a blink of an eye. Looking back at the time spent at Auckland University of Technology and the sudden COVID-19 epidemic, I realized the hard-won learning opportunities and the joy of study. In general, this year is still very fulfilling, whether studying or living, and all have left a profound impact on my life.

In the process of completing this graduation thesis, first of all, I would like to thank my supervisor, Dr. William Liu. He gave me tremendous help and encouragement throughout the whole thesis, from the initial research direction selected to the construction of the simulation models and the final thesis writing. Every step of the work of my thesis is inseparable from Dr. Liu's patient guidance and help. His serious and rigorous attitude towards scientific research has set a good example for me. I also want to thank my wife, Lili Chen, for her continuous support. Her encouragement and company gave me much motivation and kept me improving. At the same time, I would like to thank all my classmates and friends for their help in my study and life.

Abstract

The problem of traffic flow prediction is an important topic in the research of Intelligent Transportation System (ITS). With the acceleration of urbanization, the pressure of traffic load has increased. These situations urgently require scientific management and scheduling. Therefore, the development of intelligent transportation systems is imperative. The maturity of machine learning technology and the continuous development of graph neural networks allow us to better understand the temporal and spatial dynamics features of traffic flow data hidden in complex traffic networks. However, this is very challenging because of the high degree of nonlinearity, complexity, and randomness of traffic flow. These factors make the traffic flow difficult to predict and lead to low prediction accuracy, which is difficult to meet the needs of application scenarios. Traditional traffic flow prediction models and methods lack the ability to extract periodic characteristics of traffic flow data, which makes it impossible to learn more powerful traffic flow feature data reasonably. Moreover, many existing machine learning models do not fully consider the correlation between the traffic flow sequence in the spatial dimension and the temporal dimension, which makes the general applicability of the models insufficient. In addition, most combined deep neural network models ignore the characteristics of the traffic network graph structure and cannot express the high-order correlation between different nodes.

In response to the above problems, this thesis proposes four Graph Neural Network-Based Spatial-Temporal Traffic Flow Prediction models to improve the accuracy of traffic flow prediction further. First of all, this thesis adopts reasonable data analysis and dimensionality reduction strategies to improve the reliability of input data and reduce its complexity. These methods improve the model's ability to extract traffic flow features in the data input stage. Secondly, based on the Graph Neural Network (GNN), our models improve the interpretability and accuracy of the models in the

temporal dimension through the advantages of Gate Recurrent Unit (GRU) and Temporal Convolutional Network (TCN) in the temporal dimension feature processing. Combined with the powerful extraction capability of the graph convolutional neural network module for spatial dimensional features, the models' general applicability and prediction accuracy are enhanced. On this basis, this thesis uses the self-attention mechanism to enable the models to capture the dynamic dependence of traffic flow data in temporal and spatial dimensions, thereby further improving the prediction accuracy of the models.

In this thesis, the models are tested on two real traffic flow data sets. The simulation results confirm that the models can be effectively used for traffic flow prediction, and the prediction accuracy is better than other similar methods. Especially when the prediction step is long, the models have more obvious advantages in prediction accuracy. Due to the advantages of the GRU module in sequence data processing and the ability of the attention mechanism to extract dynamic dependencies between nodes, the MST-AGCRN model has higher prediction accuracy than other models we proposed. At the same time, the MST-AGCTN model has higher complexity and more parameters, and its performance is lower than expected. It needs further research and exploration.

Chapter 1
Introduction

In recent years, with the continuous growth of car ownership, traffic problems have become increasingly prominent. Driven by this background, ITS (Intelligent Transport Systems) [1] has become a research hotspot in the field of transportation as a technical means and management thinking that can effectively solve traffic problems. Among them, traffic flow prediction is an important research topic in the field of intelligent transportation. Traffic flow prediction can provide rich decision-making information support for the construction of major ITS basic components and play an important role. Since the 1960s, many scholars have carried out relevant research work in the field of traffic flow prediction. Based on these studies and the differences in the methods used, the existing traffic flow prediction models can be divided into classical statistical theories and analytical models, traditional machine learning methods, and deep learning methods.

With deep learning, breakthroughs have been continuously made in learning tasks such as natural language processing and computer vision. In recent years, in terms of traffic flow prediction, deep learning methods have achieved better results than the other two methods[2]. For example, Lv et al. [3] proposed a new deep learning-based traffic flow prediction method in 2014, which considers the temporal and spatial correlation of traffic flow. Since then, based on the temporal and spatial correlation of traffic flow, the use of deep learning methods to study traffic flow prediction problems from the perspective of temporal and spatial feature mining has become an effective and promising research direction for deep learning in traffic prediction [4, 5]. Research believes that by mining the temporal and spatial patterns of traffic flow data, accurate prediction of the basic parameters of traffic flow can be achieved. To realize the effective operation of the intelligent transportation system. However, from the

perspective of machine learning algorithms, the complexity of spatiotemporal data poses a major challenge to existing machine learning algorithms. The complexity is caused by the irregularity and non-Euclidean structure of spatial-temporal data. From this perspective, the research direction further focuses on graph neural networks. GNN [6] (compared to the convolutional and recurrent neural network) can better process traffic flow data generated from non-Euclidean domains. Therefore, it has become a popular and cutting-edge research direction to establish a time-space model through graph neural networks to solve traffic flow prediction problems. Among the latest research results in this field, the application of GNN reduces the complexity of ML(machine learning) algorithms and improves the accuracy of predictions.

In summary, how to apply graph neural networks to the field of traffic flow prediction to capture the spatial and temporal dependence of road networks. How to enhance the learnability and universal applicability of predictive models. How to improve the prediction accuracy of the prediction model. These three questions will be the key research directions of this work.

1.1 Background

With the need for social development, people began to pursue more convenient and smooth transportation. Moreover, the acceleration of urbanization has led to the rapid growth of vehicles, which has led to increased traffic pressure. According to the New Zealand Ministry of Transport statistics, New Zealand had 4.4 million vehicles in 2019. Since 2012, the number of vehicles has increased at a faster rate, with an increase of 23% in 10 years [7]. As the number of motor vehicles grows, the problem of traffic congestion has become more and more obvious. Take Auckland as an example. Due to various demographic and economic factors, motor vehicles have increased significantly, and traffic congestion has become increasingly serious.

In Auckland, highway commuters lost an average of 85 hours due to congestion in 2018, compared with 79 hours in 2017[8]. Moreover, traffic congestion has become one of the most common problems in the transportation system all over the world. It is

closely related to the government's urban planning and construction, and people's life and travel [9]. Traffic congestion has caused serious economic losses and has a great negative impact on city management and personal health. In terms of the work and life of urban residents, traffic congestion affects passengers' itineraries, prolongs passengers' travel time, and causes many inconveniences. In terms of urban environmental construction, traffic congestion leads to vehicle fuel waste and air pollution and increases the risk of traffic accidents[10]. In terms of personal health, it will increase mental stress and cause anxiety. Anxiety about traffic jams can cause people to have bad psychological and physical reactions. Negative reactions such as increased blood pressure, irritability, and decreased tolerance [11].

Therefore, improving traffic conditions and alleviating traffic congestion has become an urgent problem in the development of major cities. ITS can solve these problems well. It can grasp the traffic situation on the road in time and even predict traffic flow information such as the traffic flow in the future time period. In that case, it can make diversion measures in advance, conduct reasonable dredging of road vehicles, and reduce or even avoid road traffic congestion.

Some cities have formulated a series of traffic management policies according to local conditions to alleviate the traffic pressure in cities and reduce traffic congestion. For example, Beijing, China, has implemented measures to restrict travel with license plate numbers, effectively controlled the total number of vehicles on the road, alleviated traffic congestion to a certain extent, and reduced vehicle exhaust pollution. Some cities control traffic in certain areas and restrict certain vehicles during working days, such as Hangzhou and Shanghai in China. This series of measures can effectively reduce traffic congestion and reduce the negative impact of traffic congestion. However, it also inconveniences residents' travel to a certain extent, so there are certain limitations [12].

In recent years, computer software and hardware technology have developed rapidly. Machine learning, deep learning and artificial intelligence theory and application

technology are gradually maturing. These factors provide strong support for researchers to study ITS from the field of artificial intelligence and apply intelligent information and communication technology to provide services for transportation and management [13].

Intelligent transportation systems are committed to using intelligent methods to improve traffic conditions. As the most basic reference for traffic control, traffic flow has become the focus of research in ITS. In response to this problem, many global companies and research institutions (such as Uber and Google and China's Didi Chuxing, Ali and Huawei) have conducted in-depth research based on the characteristics of their fields[14]. For example, Didi Chuxing collects the travel data of Didi drivers to mine the temporal and spatial characteristics of urban traffic system data to build a better route recommendation system and save users waiting time due to traffic jams. Another example is Google, which provides map services around the world. Its Map App can collect traffic conditions on traffic roads and promptly remind drivers of the congestion conditions of various road sections to make better choices.

In summary, in ITS, traffic flow is a basic and important indicator. Normal traffic flow is a key parameter of the smooth operation of the entire transportation system. When the traffic flow is too large, it will cause traffic congestion and environmental pollution. At the same time, this will also increase the risk of traffic accidents, and in severe cases, it will affect the normal operation of the city and even cause traffic paralysis. Therefore, if the traffic flow can be effectively and accurately predicted in advance, the flow can be adjusted and traffic controlled before the traffic congestion occurs. This is conducive to eliminating traffic safety hazards and providing accurate information support for passengers' travel. This will also help guide passengers to arrange travel reasonably and reduce travel costs. Therefore, accurate traffic flow prediction can help the development of intelligent transportation systems. It can fundamentally reduce various losses caused by traffic congestion and has great practical significance.

1.2 Motivation

This thesis aims to study how to effectively and efficiently use huge and complex traffic flow data to build a traffic flow prediction model and use the model to make a more accurate prediction at a certain spatial and temporal dimensions in the future. As mentioned above, accurate traffic flow prediction can provide information support for ITS, which is significant for solving traffic congestion problems and boosting the development of intelligent transportation. Moreover, traffic flow prediction can provide rich decision-making information support for the construction of major basic components of ITS, and play an important role in it [14].

Traffic flow is a nonlinear time series, and its changes are affected by many factors, such as weather conditions, date, time, emergencies, traffic congestion. Therefore, traffic flow prediction is very challenging research work. Due to the upgrading of computer hardware technology, the widespread application of data acquisition equipment, and the popularization of Internet of things (IoT) [15] applications, a large amount of data is generated in the transportation system. Including vehicle GPS position data, trajectory data, people travel record data, these data have laid a data basis for the development and research of ITS. It also provides necessary information input for traffic flow prediction and improves its feasibility and reliability.

Utilize the huge data set of actual traffic flow information and its outstanding spatial and temporal characteristics. Establish a traffic flow prediction model through deep learning-related technologies, and use this as a basis to predict the future traffic flow in different regions. Such an accurate traffic flow prediction model also plays an irreplaceable role in the urban ITS and can solve some of the existing traffic problems:

(1) Statistics and prediction of traffic flow data can provide an important basis for road network planning. So as to better improve the city's intelligent transportation system.

(2) Real-time prediction of traffic flow in different areas of the transportation network

can provide immediate data source support for the urban intelligent transportation system. In this way, the reasonable traffic flow distribution can be realized, and the prevailing traffic pressure can be alleviated.

(3) Through real-time prediction, it is possible to grasp the trend of changes in urban traffic flow, make arrangements in advance for upcoming traffic problems, and improve the ability to deal with emergencies that may exist in urban traffic.

1.3 Contribution

There are three main deficiencies in the existing traffic flow prediction research models. First, many research models cannot effectively consider the temporal and spatial correlation between the time series in the traffic flow data and the spatial nodes. There is a lack of comprehensive extraction of data features in spatial-temporal dimensions. Some models ignore the temporal characteristics of the data when extracting spatial features effectively. Vice versa. The second is that many studies have neglected the dynamic dependence of traffic network nodes and the periodic characteristics of traffic flow. When some research models use feature data, they ignore that the dependency relationship (influence weight) between nodes will dynamically change with time. Third, the existing models have the shortcoming of insufficient interpretability when extracting time series features. They ignore the causality in time or lack the extraction and use of the ‘past moment’ features.

This thesis focuses on the establishment of a traffic flow prediction model through the deep neural network method. To be precise, it is the design and establishment of a depth graph neural network model based on time and space. Traffic flow prediction is a typical spatial-temporal data prediction problem, and data has a strong correlation in both temporal and spatial dimensions. Moreover, this correlation will change dynamically with time, space and other factors. Therefore, this thesis proposes a traffic flow prediction method based on graph convolution/recurrent neural network and the Self-attention mechanism through the modeling idea of graph neural network to solve the spatial and temporal dependence in the prediction process. The attention

mechanism is used to strengthen the model's ability to predict dynamic traffic flow. The main research contributions of this work are summarized as follows:

(1) Divide the periodic dependence of traffic flow data and carry out quantitative analysis of periodic dependence. This work divides the periodic dependence of predict data into three types of time components: recent dependence, daily periodic dependence and weekly periodic dependence. To quantitatively analyze the influence of these three groups of temporal-dependent components on the predicted time period, this thesis selects the data in a certain time period from the Simulation data set for Pearson correlation analysis. This work thus quantitatively analyses the degree of periodic relevance of traffic flow data, rather than merely dividing periodic components based on experience. This has improved the credibility of the prediction. On this basis, this research adopts a reasonable data dimensionality reduction strategy, which reduces the complexity of input data and the complexity of model calculations.

(2) Through selecting the data set and the comparative study with the existing traffic flow prediction models, the feasibility of the modeling ideas in this work and the practicability and significance of the established models are clarified. Many researchers tend to use more complex neural network architecture to study traffic flow prediction problems in the existing research. However, these methods artificially increase the complexity of the single-layer neural network and invisibly increase the uncertainty of the prediction model and the difficulty of data processing. While they increase the computational cost, they may not necessarily enhance the accuracy of the prediction. This research uses the most basic and simple ideas of deep neural networks to reduce the complexity of each layer of neural networks. On this basis, the receptive field of the model is expanded by increasing the depth of the neural network model. Using the depth of the graph neural network module to capture the deeper hidden dependencies of the road network spatial-temporal data. Moreover, the use of residual connection makes the deepening of the network without gradient dispersion and network degradation problems. In this way, the richness and dimensionality of the eigenvalues of the input to the fully connected layer are increased. It is possible for

the models to capture the hidden relationship of the eigenvalues.

(3) A graph neural network model based on time and space for traffic flow prediction is established in this research. Based on the research of existing traffic flow prediction methods and models, this thesis proposes some modeling methods for graph neural network based spatial-temporal traffic flow prediction. This thesis guides the research and design of the model through two themes. One is to build an efficient neural network by weighing the calculation time cost of the prediction model and the prediction accuracy. The second is to enhance the learnability and universal applicability of the prediction model by introducing the self-attention mechanism. This thesis uses the self-attention mechanism of temporal and spatial separation to capture traffic flow data's temporal and spatial dynamics from a more detailed dimension. Furthermore, using this method to reduce the interference of other non-strongly correlated factors on the prediction results.

In summary, this study compares with other models by establishing a simpler but deeper graph neural network model. Through quantitative analysis of data, this work selects more important data features as the input of the model to reduce the input of redundant features. By reducing the dimensionality of the input data, the computational overhead is reduced. By selecting the same data set to compare with other models in terms of prediction accuracy, it proves the correctness and advancement of this thesis in feature extraction and modeling of traffic flow prediction model.

1.4 Thesis Structure

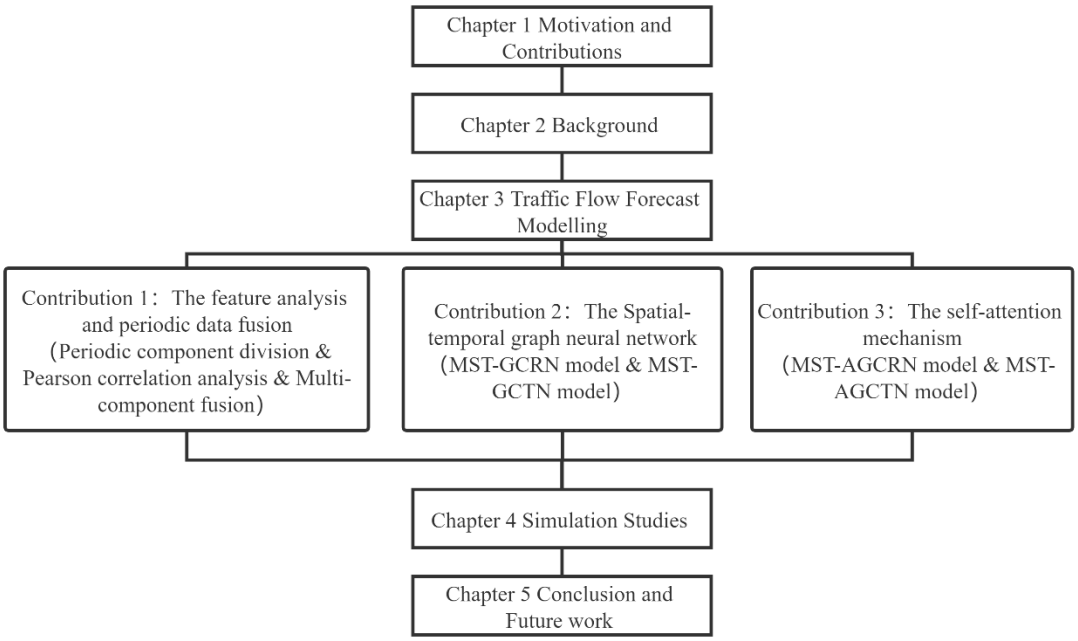


Figure 1.1 Thesis structure

The structure of the thesis is shown in Figure 1.1. This thesis mainly includes the introduction of relevant background knowledge, the design of graph neural network model for traffic flow prediction, the verification of simulation research and the comparison with the baseline method. The chapters of the thesis are arranged as follows:

Chapter 2 introduces the related basic theories used in the research process of this thesis. Mainly surveys related theories and models of traffic flow prediction research. Then, the related knowledge of neural networks is studied and discussed. Finally, three important theoretical knowledge points are further elaborated: graph convolutional neural network, self-attention mechanism and residual network.

Chapter 3 proposes three modules to solve the traffic flow prediction problem. These three modules are mainly used to solve the gaps in previous research and improve the model's prediction accuracy and universal applicability. The models we proposed are mainly composed of three modules. The first module is the feature analysis and periodic data fusion module. The second module is two models based on spatial-

temporal graph convolution. In this module, this thesis uses MST-GCRN (Multiple layers Spatial-temporal aware Graph Convolutional Recurrent Network model) and MST-GCTN (Multiple layers Spatial-temporal aware Graph Convolutional and Temporal Convolutional Network model) to extract high-dimensional spatiotemporal features. The third module is two improved models that have added a self-attention mechanism. MST-AGCRN and MST-AGCTN based on the self-attention mechanisms are used to extract more powerful high-dimensional spatial-temporal features of traffic flow data.

Chapter 4, we conducted extensive simulation research on the four proposed graph neural network-based models. The performance of the models proposed in this thesis is evaluated by comparing some baseline methods. By predicting and evaluating criteria, the performance of these models on the same data set is compared objectively with the models proposed by this thesis. At the same time, to verify the importance of spatial-temporal dual-module modeling, this chapter specifically demonstrates the influence of spatial correlation modeling and temporal correlation modeling on prediction accuracy through the model's actual performance in this aspect. Finally, the performance of the model in training is also mentioned. Through the different performances of the TCN and the GRU modules in terms of time performance, we discussed our basic modeling ideas, the rationality and feasibility of modeling, and the models' deficiencies.

Chapter 5 gives conclusions and future work. We discussed the in-depth understanding of this research field, some possible future research directions, and key links that require further research.

Chapter 2

Background

This chapter introduces the relevant basic theories used in the research process of this thesis. Firstly, it surveys the related theories and models of traffic flow prediction research. Including the classification of research methods, the existing modeling methods of traffic flow prediction, and the understanding of spatial-temporal data. Then, from the perspective of establishing the temporal and spatial model of traffic flow prediction, the related knowledge of neural networks is introduced, including convolutional neural network (CNN), recurrent neural network (RNN), graph neural network (GNN). Finally, based on the actual needs of this thesis, the three important theoretical knowledge points are further elaborated: graph convolutional neural network, attention mechanism and residual network.

2.1 Overview of Traffic Flow Prediction

Over the years, many scholars have carried out relevant research work in the field of traffic flow prediction. Their research includes a variety of traffic flow prediction scenarios, such as urban geographic area passenger flow forecast, urban subway station passenger volume forecast, and urban bus station passenger volume forecast [16, 17], high-speed rail station passenger volume forecast [13], expressway traffic flow predict, expressway traffic speed predict [18]. Among them, traffic flow prediction and traffic speed prediction are often unified as traffic flow prediction problems.

These studies respectively proposed various effective and novel models and methods for the prediction scenarios and achieved a series of fruitful research results. With the rapid development of computer and information technology in recent years, more and more different types of traffic flow prediction methods and related technologies have

been proposed [5]. For example, some algorithms related to parameter optimization, particle swarm optimization, genetic algorithm. have also been used by researchers to predict traffic flow. Furthermore, many artificial intelligence algorithms have achieved good results.

Based on these studies, according to the differences in the methods used in traffic flow prediction, the existing traffic flow prediction-related models can be divided into three categories: classical statistical theories and analytical models, traditional machine learning methods, and deep learning methods. Figure 2.1 shows the classification diagram of the traffic flow prediction models.

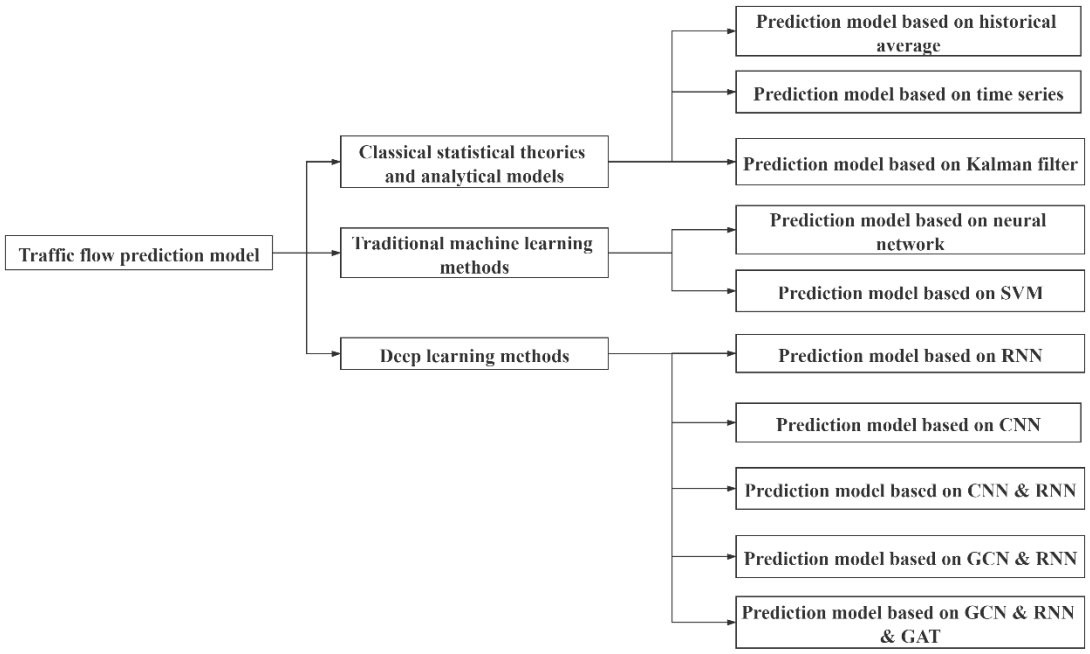


Figure 2.1 Classification of traffic flow prediction models

2.1.1 Classical statistical theories and analytical models

Among the classical statistical theories and analytical models, the simplest and most direct method is the History Average (HA), which calculates the average value of traffic flow during a certain period of history as the future traffic flow's predicted value. This method is simple, easy to understand, easy to implement, and has a certain reference value for future traffic predictions. However, it cannot consider the impact

of random and unexpected events on traffic flow, and its accuracy is low. Models based on statistical methods are earlier methods used for traffic flow prediction. Stephanedes et al. [19] first published the use of HA models for traffic control. Ahmed et al. [20] used time series related knowledge to establish a prediction model for this field. R.E.Kalman et al. [21] proposed a model based on the Kalman filtering algorithm. Okutani et al. [22] considered the impact between a road segment to be predicted and a road segment in a surrounding area. They then established a prediction model through the Kalman filter algorithm. At the same time, to improve the model parameters, they used a large number of prediction error corrections. Some scholars also use classic time series prediction models to predict traffic flow, such as vector autoregressive model (VAR) [23], autoregressive moving average model (ARMA) [24], autoregressive integrated moving average model (ARIMA) [25, 26] and its variant models, such as stationarity ARIMA and seasonal ARIMA [26]. These classical time series methods belong to Parametric Models, which assume that the value of the time series at time t depends only linearly on its historical observations and random noise. When realizing, forecasting is made by mining the law of the traffic flow in the time dimension from the historical time series of the traffic flow. This type of method generally achieves good results when the time series has a certain periodicity or regularity. However, the traffic flow sequence in the real traffic scene is affected by many factors and has strong randomness and uncertainty, so the prediction accuracy of this type of method is not good. In addition, this type of method is difficult to integrate other environmental data for traffic prediction.

2.1.2 Traditional machine learning methods

The rapid development of artificial intelligence and big data has performed well in various fields in recent years. More and more research teams are also applying traditional machine learning methods to traffic flow prediction. These models can continuously adjust the parameters to the optimal through adaptive learning to obtain more accurate calculation results. The addition of machine learning technology makes

it possible to use relatively little data to predict traffic flow models.

In traditional machine learning methods, some scholars use K-Nearest Neighbours (KNN) to predict traffic. For example, Zheng et al. [27] calculated geographic locations with similar traffic conditions through KNN. Then combine its flow to make short-term forecasts. The biggest disadvantage of this method is that it is difficult to determine the best value of the parameter K, that is, the number of nearest neighbors. Some scholars use integrated models to predict traffic flow. For example, Wei et al. [28] combined Empirical Model Decomposition (EMD) with BP (Back Propagation) neural network to predict subway pedestrian flow. In addition, Support Vector Machine (SVM) model is widely used in short-term traffic flow prediction. Its working principle is to establish a hyperplane in high-dimensional space to solve the non-linear classification problem, so as to achieve a more accurate classification effect. Jiang et al. [17] combined Ensemble Empirical Mode Decomposition (EEMD) and Gray Support Vector Machine (GSVM) to predict short-term high-speed rail traffic. Later, Sun et al. [29] put forward a wavelet support vector machine (Wavelet-SVM) for short-term traffic prediction in subway stations. These integration methods mainly model nonlinearity from the perspective of the time dimension of the sequence. However, it does not consider the direct correlation between traffic flow sequences in spatial dimensions. The model capacity is limited, and it is not easy to expand.

2.1.3 Deep learning methods

1. Deep learning based prediction methods

As deep learning has continuously made breakthroughs in learning tasks such as natural language processing and computer vision [29], scholars have begun to study how to apply deep learning technology to traffic flow prediction tasks. They combined classic time series prediction and traditional ML methods for traffic flow prediction to improve the accuracy of the forecast. Liu et al. [30] manually construct feature vectors of different factors and then input them into a deep feedforward neural network to predict traffic flow. This method assumes that the input elements are

independent of each other and requires a lot of feature engineering. To learn the relationship of traffic sequence in time, some scholars[31, 32]use RNN time series prediction models and its variant models (Long-Short Term Memory (LSTM)[33], GRU[34]) for traffic flow prediction. Some scholars who combine RNN models with traditional machine learning methods[35]. Although RNN models can capture the sequential connections of traffic flow in the time dimension and have good scalability, they cannot process sequences based on specific spatial relationships between sequences. To consider the temporal and spatial correlation of traffic flow at the same time, some scholars try to study traffic flow prediction from the perspective of temporal and spatial feature mining. For example, Zhang et al. [36] divided the urban area into grids of equal size to calculate the regional inflow and outflow, and designed a deep spatial-temporal residual network model ST-ResNet to predict the inflow and outflow in each region, and achieved good prediction results. This research also provides new ideas for other researchers. Inspired by the ST-ResNet model, Jin [37] constructed the STRCNs model, combined CNN and LSTM to capture regional traffic's temporal and spatial dependence, and achieved better prediction results than the ST-ResNet model. In addition, Yao et al. [38] simulated spatial-temporal dependent traffic prediction models by integrating CNN and RNN (LSTM) models and subsequently proposed spatiotemporal dynamic networks based on the similarity between dynamic learning locations [39]. Although these models can reasonably consider the spatial-temporal correlation between the flow of urban areas and extract rich spatial-temporal features, they can only process Euclidean structure data and are not suitable for non-Euclidean structure data.

Therefore, some scholars later studied how to apply graph convolution technology to spatial-temporal data mining. Seo et al. [40] proposed graph convolutional recurrent networks based on graph convolution and recurrent networks, but it is difficult to find the most suitable combination model to optimize the prediction task. Zhao et al. [41] embedded graph convolution into the gated recurrent network as a feature extraction unit to extract the traffic network's spatial features and apply it to the real traffic data

set to achieve good results. However, because it uses the early Graph Convolutional Network (GCN) model, the model space feature extraction capability is limited, and the model calculation speed is slow. Li et al. [42] fused graph convolution with GRU to capture the temporal and spatial dependence of traffic flow, designed a DCRNN (Diffusion Convolutional Recurrent Neural Network) model and performed long-term prediction based on the Encoder-Decoder framework. Yu et al. [43] designed a spatiotemporal graph convolution model STGCN based on GLU (Gated Linear Units) [44]. Using graph convolution and gated convolution to capture the spatiotemporal dependence of vehicle speed on each section of the highway. It works well, and the training time of this model is much shorter than that of the DCRNN model. Later, Diao et al. [45] improved their models based on Yu's work, designed a Laplacian matrix estimator and proposed a dynamic graph convolutional neural network model-DGCNN. Guo et al. [18] modeled the correlation between the targeted traffic to be predicted and its recent traffic, daily periodic traffic, and weekly periodic traffic, and introduced a spatiotemporal attention mechanism to capture the spatiotemporal correlation between nodes. These methods consider the spatial correlation between nodes from the perspective of the positional relationship between nodes. They are based on a static road network structure and capture the correlation between node traffic flows according to the low-order neighboring nodes of each node. And learn the spatial feature representation of the node according to the correlation. Later, some scholars used multiple graphs to solve spatial-temporal data mining tasks. For example, Chai et al. [46] applied multi-graph convolution to bicycle rental traffic prediction. Geng et al. [47] used urban areas as nodes. Based on POI information (point of interest) and road connectivity, multiple graphs are constructed. In addition, the combination of recurrent neural network and graph convolution is used to predict the demand for taxi rides by passengers in various regions. However, these multi-graph convolution models only use the graph structure to learn the embedding representation of the site, ignoring the dynamics of the relationship between the nodes, and do not consider the high-order correlation between the embedding representations

learned by the nodes on different graph structures.

2. The relationship between deep learning and spatial-temporal data mining

The main research direction of this thesis is the prediction of traffic flow. Traffic flow prediction is a typical spatiotemporal data mining problem. Therefore, this section will introduce the concepts and theoretical knowledge related to time and space involved in Deep learning research.

(1). Understanding of spatial-temporal data

Temporal data, also known as time series, is a data series formed by the same phenomenon at different times. Data in the real world is often related to time, and a series of observations obtained in chronological order is called time series data. Common ones are temperature changes, stock prices. There are many mature time series mining algorithms [48] to obtain the rich information contained in time series data.

Spatial data, data with spatial coordinates, is a special type of data that can quantitatively describe things or phenomena with positioning significance. For example, the geographic location and distribution characteristics of objects.

Spatial-temporal data is spatial data that has temporal elements and changes with time. For example, online car-hailing order data has time attributes when the order is created and contains spatial information [49]. It has obvious spatial distribution characteristics and the characteristics of huge amounts of data, nonlinearity, and time-varying.

With the advancement and development of data collection equipment and methods, relevant spatial-temporal data can be effectively collected for research content in various fields. It lays a data foundation for the development of spatiotemporal data mining (STDM) algorithms and models. For example, traffic flow prediction is a typical spatial-temporal data mining problem. To serve the final prediction task, researchers need to analyse relevant data from time and space dimensions to mine the

spatial-temporal information in each measurement data.

(2). Spatial-temporal data mining

In modern human social life, complex behaviors bring about the accumulation of temporal and spatial data. This kind of spatial-temporal data can be deeply excavated to reveal human social life. For example, it is possible to mine the user's travel trajectory and travel rules through mobile behavior and provide services such as location prediction and location recommendation [50, 51]. Social behaviors can be used to apply social spatiotemporal data for identity recognition and social relationship inference. The urban calculation of population flow can be carried out through migration behavior, and the interpretation and prediction of cluster behavior can be carried out through the analysis of cluster behavior. There are many kinds of spatial-temporal data in actual scenes, among which common spatial-temporal data can be divided into three categories: Event Data, Trajectory Data, and Raster Data.

1). Event Data

Generally speaking, any event can be represented by a point in time and location, where location and time respectively represent the location and time of the event. It is simply denoted as (l_i, t_j). In addition to the information of time and space dimensions, each event also contains other non-temporal information. For example, the event involves information about the type of events, such as the crowd and the nature of the event. There are three types of events A, B, and C, as shown in Figure 2.2. Since an event is not always a point object, other geometric figures can be used to describe it. Such as linear, polygonal. For example, a forest fire can be represented by a spatial polygon, which represents the affected area. Similarly, events generally do not exist at a certain instant in time but correspond to the start and end times. Therefore, the time in the spatial-temporal point generally represents a certain period of time corresponding to the event's occurrence.

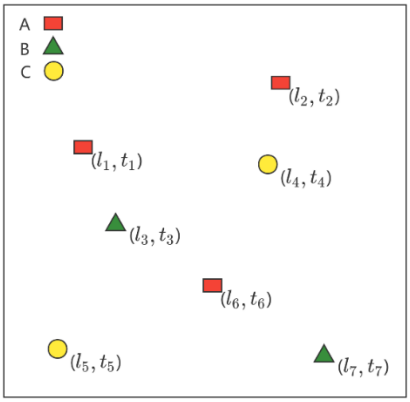


Figure 2.2 Three kinds of events [42]

2). Trajectory Data

The trajectory is the path left by the target moving in space over time. For example, the trajectory of the taxi from the place where passengers boarded to the place where they got off the vehicle, the trajectory of animal migration. Generally, trajectory data is collected by sensors installed on moving objects. These sensors regularly transmit location information of moving targets. For example, a taxi can obtain its driving trajectory through GPS positioning data. The trajectory data generally contain other types of information about the moving target in addition to the position information that changes over time. For example, the vehicle's speed information during the driving process, the heartbeat rate of the person during the running process, and other information. Trajectory data is often used in applications such as transportation and ecological science.

3). Raster Data

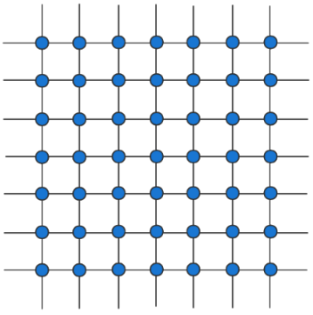


Figure 2.3 Regular space

In raster data, every observation in the spatial-temporal domain is recorded in a fixed unit in the spatial-temporal grid. Raster data generally corresponds to a set of fixed-position objects, denoted as $S = \{s_1, s_2, \dots s_m\}$. These location objects can be regularly distributed in space, and a constant distance between adjacent objects is maintained. It is similar to the distribution of elements in the image, as shown in Figure 2.3. It can also be distributed irregularly in space, such as an intersection sensor network, as shown in Figure 2.4. For each location object, the raster data records all its observations in a fixed set of time stamps $T = \{t_1, t_2, \dots t_n\}$. Adjacent time marks can be either a fixed time interval or an unfixed time interval. The Cartesian product of the location object collection, the timetable and the collection forms the spatial-temporal grid $S \times T$ corresponding to the raster data. Each value (s_i, t_j) in the network corresponds to a measurement value.

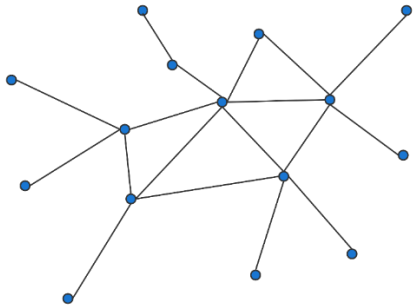


Figure 2.4 Irregular space

In addition, another characteristic of spatial-temporal data is that data has many characteristics. To solve problems of these kinds, a variety of information is needed. For example, predicting road traffic conditions, the available data features include many kinds, such as traffic flow, average lane occupancy rate, average road speed, and external factors such as weather and natural disasters. Therefore, the application of deep learning in spatiotemporal data mining is more difficult. It is very different from image recognition only based on image data, and voice recognition only needs to be based on voice data. The type and quality of data required by it vary, and it also requires preprocessing. Therefore, it is necessary to use a neural network capable of learning graph structure to mine complex data and perform deep learning tasks.

(3). Deep Learning based Spatial-temporal Data Mining

Through the overview, deep learning is currently the most advanced and effective method and model in the field of traffic flow prediction research. According to the previous analysis, the ability of deep learning in spatiotemporal data mining is significantly stronger than the other two models and methods. Compared with classical statistical models and traditional machine learning, deep learning can not only learn the characteristics and correlation of spatiotemporal data but also does not require the manual design of features. It can mine more complex features in spatial-temporal data. In addition, deep learning is better at dealing with complex spatial-temporal data problems. By deepening the network, extracting features from shallow information for analysis and integration, generating deeper features has great advantages for solving complex time-space mining problems in reality.

In the next sections of this chapter, this work focuses on the field of deep learning. It mainly introduces related concepts based on deep neural networks and related principles and technologies of traffic flow prediction based on graph neural networks. First, an overview of traditional neural networks, such as CNN and RNN. Then introduce the basic knowledge of GNN. Third, based on the focus of this thesis, we introduce the graph convolutional network (GCN) used to capture the spatial dependence of the traffic flow of the road network. Finally, a brief description of multilayer perceptrons, RNN, and common RNN variants (LSTM, GRU) captures temporal dependence in traffic flow prediction.

2.2 CNN based Traffic Flow Prediction Models

Convolutional Neural Network (CNN) is a highly efficient deep learning recognition method. It has developed rapidly in recent years and has attracted widespread attention. It is a deep feedforward neural network with the characteristics of local connection and weight sharing[52]. The first scene where researchers used convolutional neural networks was Yann Lecun's application in handwritten data recognition [53]. Then in the following years, CNN began to be widely used in

various fields. It is mainly used in various image and video analysis tasks, such as general image classification, face and object recognition. Its accuracy is generally far beyond that of other neural network models. Research shows that convolutional neural networks perform well in application scenarios such as image recognition, natural language processing, and speech recognition. Its characteristic is that it can automatically capture the characteristics of the image by using the convolutional layer. The model reduces the cost of manually extracting features, and at the same time, greatly improves the accuracy of image recognition. It reduces people's dependence on image recognition related knowledge and improves the application value of the model.

2.2.1 Structure of convolutional neural network

Generally speaking, CNN is a kind of feedforward neural network formed by the cross stacking of convolutional layer, convergence layer and fully connected layer. The structure of a CNN has three characteristics: local connection, weight sharing and sub-sampling. These three characteristics make the CNN has certain translation, scaling and rotation invariance capabilities. Compared with ordinary neural networks, CNN has fewer parameters, and the training process is completed by backpropagation algorithms.

Convolutional neural networks are very similar in structure to ordinary artificial neural networks, and both are composed of neurons that can learn weights and biases. First, each neuron will receive some input, and then it will do some dot product operations. The output of the CNN is the score corresponding to each category. In this process, some computational skills are the same as ordinary neural networks.

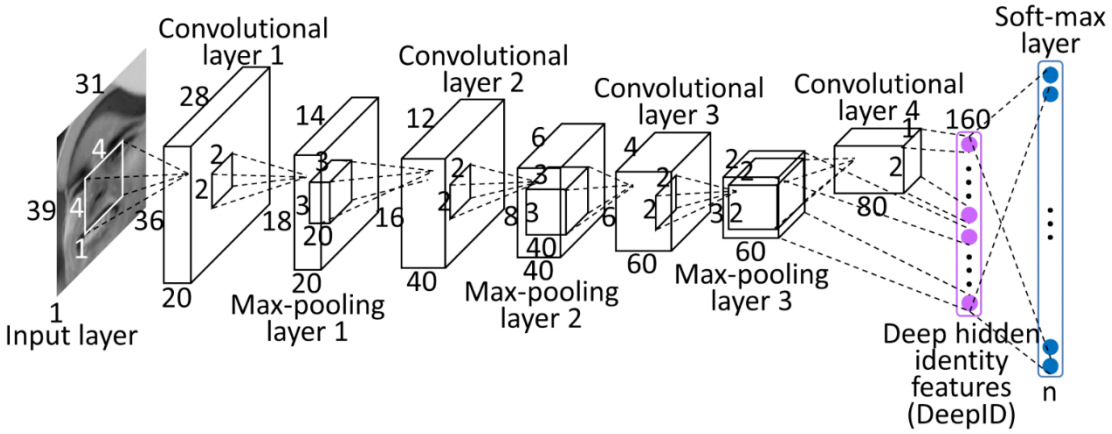


Figure 2.5 Convolutional neural network structure (image recognition) [54]

As shown in Figure 2.5. It is a convolutional neural network structure applied in the field of image recognition [54]. The structure consists of an input layer, four convolutional layers, three pooling layers, a fully connected layer and a SoftMax layer. As can be seen from the figure, CNN is a deep learning model with clear hierarchies. The input is raw data such as RGB images. The principle of CNN is to extract the deep information hidden in the original data from the input layer after a series of operations such as multiple convolutions, pooling, and activation function processing. The final result is the local characteristics of the original data. Generally speaking, when dealing with classification problems, to be able to judge the category of the image by the probability accurately, it will finally use a SoftMax operation. This operation can transform these local features into probability distributions. In convolutional neural networks, different operations are represented by "layers". For example, in a CNN, a convolutional layer is used to implement a convolution operation, and a pooling layer is used to complete the pooling operation. The convolutional layer and pooling layer are commonly used network layers in convolutional neural networks, and their characteristics are as follows:

1. Convolutional layer

The convolution operation is one of the most basic operations in convolutional neural networks. It is also a core point of difference between convolutional neural networks and ordinary artificial neural networks. Furthermore, convolution is an important

operation commonly used in analytical mathematics. Usually, one-dimensional convolution or two-dimensional convolution is used to process signals or images. In particular, two-dimensional convolution is often used for image processing. Because the structure of the image is two-dimensional, one-dimensional convolution needs to be extended. The essence of two-dimensional convolution is to use a convolution kernel matrix to slide on the image matrix. After each sliding, the product of the pixels in the overlapping part of the image and the matrix is added to obtain an output value. It finally got a new image[55]. Figure 2.6 shows an example of a convolution operation. Assuming that the input image A represents a $3X3$ matrix, the corresponding convolution kernel B represents a $2X2$ matrix. Assuming that every time a convolution operation is passed, the convolution kernel will slide one pixel, that is to say, the step size of the convolution is 1.

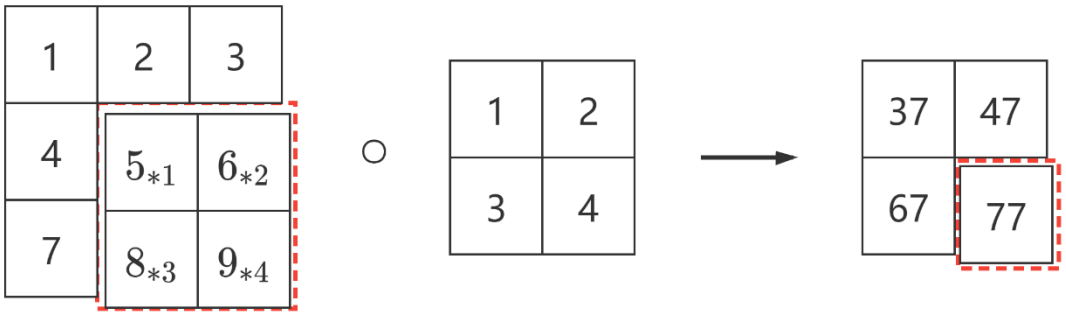


Figure 2.6 Convolution operation

As shown in Figure 2.6, the first convolution operation starts from the (0,0) position of the input image. First, the original image parameters and the corresponding convolution kernel parameters are multiplied bit by bit. Second, add the numbers obtained in the first step to get the result 37. Then, according to this calculation method, let the convolution kernel perform convolution operations on the original input image from left to right and from top to bottom according to the corresponding step size. The final output result is a $2X2$ output. And this output will be used as the input of the next layer of operation. When there are multiple convolution kernels, the output is a three-dimensional tensor. Among them, the depth of the three-dimensional tensor is exactly equal to the number of convolution kernels.

Generally speaking, natural images have fixed features, so this feature learned in a certain part of the image can be extended and applied to other parts of the image. The convolution operation is to operate through a certain number and size of convolution kernels in each position of the entire image to obtain the information of each part of the original image in turn. It is a local operation so that the convolution operation can be used for image processing well.

2. Pooling layer

After the original image undergoes a convolution operation, the characteristic image obtained will have certain static information. The pooling operation is to perform statistical operations on the feature images obtained by the convolution operation. Pooling, on the one hand, needs to reduce the size of the network feature image. On the other hand, it also needs to retain the important information in the feature image. The feature image obtained after a convolution operation of the original image and the size before the convolution operation does not change much. If the convolution operation continues during the training process, it will cause a huge amount of calculation.

Moreover, the difficulty of network training will increase as the depth of the network increases. By using the pooling layer, CNN can reduce the amount of calculation of the network without sacrificing the original image characteristic information. It can speed up the network training and reduce the resolution of the image after the convolution operation.

Pooling operations generally have two methods: max pooling and mean pooling [56]. Mean pooling refers to keeping the average of all pixels in each part divided by the feature image in each pooling operation. Max pooling refers to keeping the maximum value of each part's pixels divided by the feature image in each pooling operation. In addition to these two common pooling operations, another method is random pooling. Random pooling means randomly retaining the values of all pixels in each part of the feature image. Figure 2.7 shows a max-pooling operation.

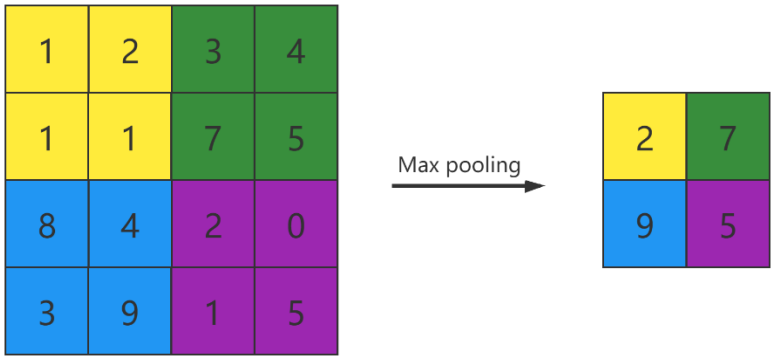


Figure 2.7 Max pooling operation

2.2.2 Features of convolutional neural network

The main difference between CNN and the traditional neural network is that CNN has the characteristics of local connectivity and weight sharing [57].

1. Local connectivity

Generally speaking, for external things, people's cognition is often from the partial to the whole. Regarding the spatial characteristics of an image, it is generally believed that the closer the parts of the image are, the higher the correlation. However, the correlation between parts that are far away will be relatively low. Based on this, it is not necessary for a neuron to perceive the entire image simultaneously when it perceives the image. It only needs to associate each part of the image separately. The idea of local connectivity comes from the structure of the visual system in biology. For neurons in the visual system, they will only respond to stimuli in certain specific areas. As an important feature of convolutional neural networks, the local connection makes the layers of convolutional neural networks not completely connected like traditional neural networks. The size of the receptive field of the convolutional neural network is set only to accept signals from a small area. This ensures that the neurons in the current network layer will only be connected to the pixels in the corresponding receptive area of the previous layer. Finally, by combining the local information of all neurons, all the upper layer information is obtained. This feature of local connectivity makes the convolutional neural network greatly reduce the number of parameters in

the network structure [58].

2. Weight sharing

Weight sharing means that although multiple convolution kernels can exist in each convolution layer, the parameters of the same convolution kernel in each part of the entire picture are shared. This feature does not change due to changes in the position of the picture. For an image, the characteristics of any position on the image may appear in another position of the image. Therefore, the same feature can be perceived in different positions of the image through weight sharing. For example, for face recognition, the parameters of the convolution kernel learned by the person's left eye through recognition can also be used to recognize the person's right eye. The weight-sharing feature of CNN can reduce the probability of overfitting, and it can also reduce the network model's complexity and improve its computational efficiency [59].

2.2.3 CNN based traffic flow prediction models

As mentioned earlier, CNN can effectively extract image features through convolution operations. Therefore, the feature extraction performance of CNN in Euclidean space is considered to be excellent. Due to the temporal and spatial two-dimensional properties of traffic flow. In previous research on traffic flow prediction, CNN was mainly used to extract the spatial correlation of traffic flow data. This is because the non-Euclidean characteristics of the transportation network make it difficult to be directly matrixed and applied to CNN. Therefore, researchers often convert traffic flows at different times into images of the traffic grid structure. Then, this kind of image is matrixed (grid). Different grids can be used to represent different traffic areas. In this way, CNN can extract and recognize the spatial data characteristics of different traffic areas. Different application scenarios, model improvements, and different understanding of the division of traffic networks or traffic flow data sets are the three main differences in the application of CNN in the field of traffic flow prediction.

The research of Davis et al. [60] focused on the forecast of taxi supply and demand in ITS. The study uses fixed-area rectangular cells or variable-area polygonal cells to grid the transportation network of the entire urban area. On this basis, CNN was used to extract spatial features separately, and the two models' prediction accuracy was compared. There are also similar studies on CNN in the field of taxi supply and demand forecasting [61],[62]and so on. According to the actual needs of the task, the research has different ideas in the selection and optimization of the input data set.

Guo et al.[63] proposed a new end-to-end ST-3DNet model to extract traffic raster data. In their research, the researchers added time data to the matrixed two-dimensional space data to form a new three-dimensional space-time data. Input such three-dimensional data into CNN and use 3D convolution to capture traffic data's temporal and spatial correlation. Research on the optimization and improvement of the CNN model can also be seen in studies such as [64]. Jiang et al. [65]focused on the prediction research of crowd flow. They built a system called Deep-Urban-Event, which converts the ever-changing crowd dynamic data in urban areas into a series of thermal images of traffic. CNN extracts the spatial characteristics of the heat map to predict the crowd flow trend in each area. Similar research directions can also be seen in the research of [64]. Lee et al. [66] applied CNN to the demand forecasting field of online car-hailing. The research proposes an efficient model architecture with a fully convolutional network and time-guided embedding to learn complex spatial-temporal features. The research uses this model to predict the future demand for taxis. The model is mainly optimized and improved on the basic CNN model, using average pooling and 1-dimensional convolution to meet the actual needs of research.

It can be seen from the above research that CNN has advantages in processing rasterized network data. Therefore, many researchers apply it in processing spatial data of transportation networks. However, in terms of specific applications, reasonably defining the input data in Euclidean space is a difficult point for research. Therefore, the application of the CNN model in the field of traffic flow prediction has limitations. Many models and methods perform well in specific scenarios. However,

most CNN-based models are not universal. A model that performs well in one task has a huge drop in performance when applied to other tasks.

2.3 RNN based Traffic Flow Prediction Models

In the feedforward neural network, the transmission of information is one-way. Although this method of information transmission makes the network easier to learn, it also weakens the neural network's learning ability. In biological neural networks, the connection between each neuron is more complicated. The feedforward neural network can be regarded as a complex function. Each input in the network is independent. That is to say, the network's output only depends on the current input of the network. However, in many practical applications, the input of a neural network is not only related to the input at the current moment but also has a certain relationship with the output of the network in the past period of time. For example, for a finite state automaton, its state at the next moment (corresponding to the output of the network) is not only related to the input at the current moment but also related to the current state (corresponding to the output of the previous moment in the network). Therefore, for a static network such as a feedforward neural network, the dimensions of its input and output are fixed. It cannot handle the situation where the network output depends on the output at the previous moment. For example, data such as text, voice, and video. Therefore, a neural network model with stronger learning ability is needed when dealing with general sequence problems, namely recurrent neural network (RNN). It can process sequence data of any length because its neurons have the characteristics of self-feedback regulation. Currently, recurrent neural networks have been widely used in speech recognition, language models, prediction based on sequence data, and natural language generation[67-69].

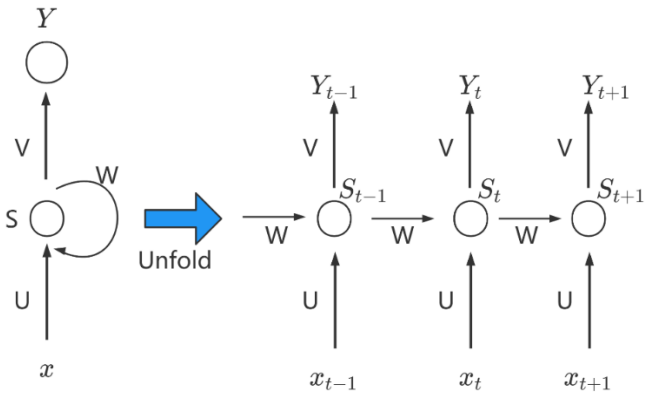


Figure 2.8 Structure of RNN[70]

The overall structure of the recurrent neural network is shown in Figure 2.8 [70]. The structure on the left is foldable and can be expanded on the time step as shown on the right. Where X_t represents the input vector at the t-th time step. Y_t represents the output vector of the t-th time step. S_t represents the hidden layer vector at the t-th time step. U, V, and W represent the mapping parameter matrix and are shared in each time step. The hidden vector S_t of each time step in the RNN contains the information of the historical time step. The calculation process of the single hidden layer RNN model is shown in formulas (2-1) and (2-2), where f represents a non-linear activation function, which can be functions such as tanh and ReLU.

$$S_t = f(UX_t + WS_{t-1} + b_s) \quad (2\text{-}1)$$

$$Y_t = VS_t + b_y \quad (2\text{-}2)$$

Although the hidden vector of RNN can retain part of the historical time step information, the RNN model has shortcomings such as insufficient long-term memory, easy gradient disappearance, or gradient explosion due to its simple structure. Therefore, some scholars have proposed a variant model - LSTM to improve RNN.

2.3.1 LSTM neural network structure

To solve the inability of RNN to retain long-term memory, Schmidhuber et al. [71] proposed the LSTM model. On this basis, many researchers have made certain improvements to the model from the practical application level [72-74]. As shown in Figure 2.8: LSTM adds three "gates" to control the input, output and hidden layer

state information in the RNN. They correspond to input gate, output gate and forget gate, respectively. Moreover, for the problem of RNN's gradient disappearance, LSTM also adds a memory neuron to alleviate this problem. The four parts cooperate with each other to determine memory information or forget information [75].

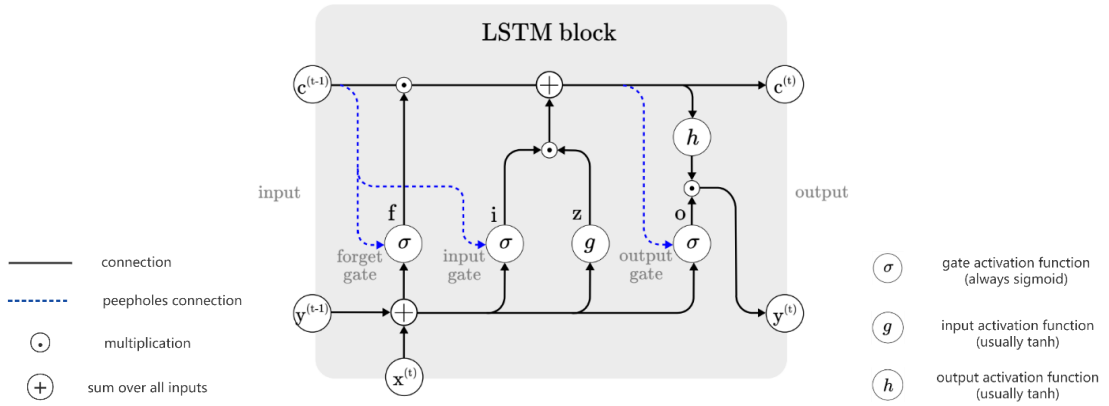


Figure 2.9 Architecture of a typical vanilla LSTM block[76]

Specifically, suppose $x_t \in \mathbb{R}^{n\times c}$ is the current n input sequence data with feature dimension c, and $H_{t-1} \in \mathbb{R}^{n\times h}$ is the hidden layer of the previous time step state. Then the input gate $I_t \in \mathbb{R}^{n\times h}$, the forget gate $F_t \in \mathbb{R}^{n\times h}$ and the output gate $O_t \in \mathbb{R}^{n\times h}$ are calculated as follows:

$$I_t = \sigma(W_{in}x_t + W_{ih}H_{t-1} + b_i) \quad (2\text{-}3)$$

$$F_t = \sigma(W_{fn}x_t + W_{fh}H_{t-1} + b_f) \quad (2\text{-}4)$$

$$O_t = \sigma(W_{on}x_t + W_{oh}H_{t-1} + b_o) \quad (2\text{-}5)$$

Among them, W and b are learnable weight parameters and bias terms, respectively. And the forget gate and input gate control the memory neuron $C_t \in \mathbb{R}^{n\times h}$ by formula (2-6):

$$C_t = F_t \odot C_{t-1} + I_t \odot \widetilde{C}_t \quad (2\text{-}6)$$

Where $\odot$ represents the Hamada multiplication in which the element corresponds to the multiplication. $\widetilde{C}_t$ represents a candidate memory neuron. Use $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ as the activation function:

$$\widetilde{C}_t = \tanh\left(Wx + W_h H_{t-1} + b\right) \quad (2\text{-}7)$$

Among them, W and b are learnable weights and bias items.

The information in the memory neuron C_t will be incorporated into the hidden layer state H_t by the output gate O_t:

$$H_t = O_t \odot \tanh(C_t) \quad (2\text{-}8)$$

Among them, $\odot$ represents the Hadamard multiplication of the element corresponding to the multiplication.

In short, it can be obtained from formula (2-6) that the forgetting gate F_t can control whether the information of the memory neuron at the previous time step needs to be integrated into the memory neuron at the current time step. The input gate I_t controls whether the input information of the current time step needs to be incorporated into the memory neuron. For example, when $F_t = 1$ and $I_t = 0$, then $C_t = C_{t-1}$, which means that the information of the historical time step is passed on consistently. Therefore, LSTM can well capture the long-term dependencies in the sequence, and the problem of gradient disappearance in RNN will not occur. However, the LSTM network architecture is too complicated, and there are too many parameters, which affects the computational efficiency of the model.

2.3.2 GRU neural network architecture

GRU (Gate Recurrent Unit) [77] is a recurrent neural network structure. The main idea of GRU is the same as LSTM. However, to alleviate the overcomplication of the LSTM structure, the GRU streamlined it. It simplifies the three "gates" in LSTM into two "gates": reset gate $R_t \in \mathbb{R}^{n\times h}$ and update gate $U_t \in \mathbb{R}^{n\times h}$, the calculation formula is as follows:

$$R_t = \sigma(W_{rn}x_t + W_{rh}H_{t-1} + b_r) \quad (2\text{-}9)$$

$$U_t = \sigma(W_{un}x_t + W_{uh}H_{t-1} + b_u) \quad (2\text{-}10)$$

Among them, W and b are learnable weights and bias items.

GRU abandons the memory neuron, and directly uses the update gate U_t to linearly combine the hidden layer state H_{t-1} of the previous time step and the current candidate hidden layer state $\widetilde{H}_t$:

$$H_t = U_t \odot \mathrm{H_{t-1}} + (1 - U_t) \odot \widetilde{H}_t \quad (2\text{-}11)$$

The candidate hidden layer state $\widetilde{H}_t$ is controlled by resetting the gate:

$$\widetilde{H}_t = \tanh\left(W_{hn} x_t + R_t \odot W_{hn} H_{t-1} + b_h\right) \quad (2\text{-}12)$$

Among them, W and b are learnable weight parameters and bias terms, and $\odot$ represents the Hamada multiplication of the element corresponding to the multiplication.

In short, only the hidden layer state of the previous time step contains historical information. When the reset gate $R_t = 0$, the hidden layer state at the last moment will be discarded. Therefore, the reset gate can capture the short-term dependence of the network. When the update gate $U_t = 1$, $H_t = H_{t-1}$ means that all historical information is retained so that the update gate can control the long-term dependence of the network.

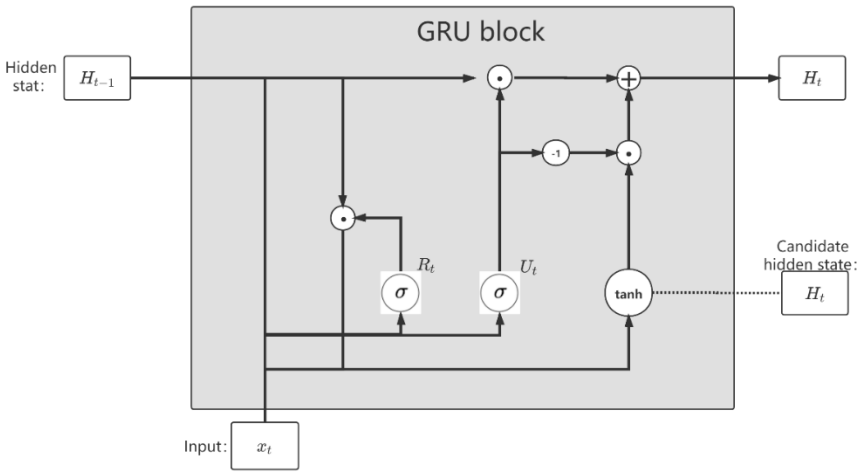


Figure 2.10 Architecture of GRU block[77]

The basic unit structure of GRU is shown in Figure 2.10. In the figure, R_t represents the reset gate, and U_t represents the update gate. The reset gate can control the hidden state at the previous moment and control how much information is input into the current state.

However, whether it is RNN, LSTM or GRU, they are all recurrent structures. Therefore, when calculating the current output, they must wait for the previous neuron to complete the calculation and pass the hidden layer state before moving forward.

Therefore, the network model of the structure cannot be calculated in parallel on a large scale like the CNN architecture model.

2.3.3 Temporal convolutional network(TCN) architecture

From the discussion of RNN in the previous section, it can be seen that LSTM, GRU and other recurrent neural networks can indeed model sequences very well. However, the recurrent neural network itself has the problems of gradient dispersion and gradient explosion. Variants such as LSTM and GRU only alleviate the gradient dispersion problem. At the same time, the recurrent neural network is difficult to obtain high training efficiency due to its unique calculation method. Because the output of the RNN at each moment is produced by performing the same operation on the output at the previous moment. The calculation at the current moment depends on the calculation result at the previous moment. This inherent characteristic makes RNN can only calculate serially one by one, which is difficult to parallelize. It is even more difficult to have a better acceleration effect on the GPU. Therefore, in terms of traffic flow prediction, some of these shortcomings cause the performance of RNN in all aspects of this field to fail to achieve the desired effect.

Compared with RNN, CNN allows parallel convolution calculations, making CNN have a good acceleration effect on GPU. However, traditional CNN has big flaws in dealing with time series problems. On the one hand, the size of the convolution kernel is limited. On the other hand, it integrates the past and future time feature information into the features of the node indiscriminately. To solve this problem, Bai et al.[78] proposed a new architecture-TCN in 2018, hoping to defeat RNN in a variety of mainstream RNN applications such as Polyphonic Music Modeling, Word-and Character-Level Language Modeling. In simulations, comparing it with a variety of RNN structures, it is found that TCN can reach or even exceed the RNN model on a variety of tasks. Although RNNs can theoretically have infinite "memory", they are difficult to train and parallel. Compared with RNN, TCN inherits the advantages of CNN and has the characteristics of longer "memory" and easier parallelism. Therefore,

TCN can take advantage of the parallel computing features of GPU to greatly speed up the model training process.

Theoretically, TCN[79] discards the value after t time in each step of convolution, so that the output at time t depends only on the input before time t. Therefore, in TCN, the function of the convolution kernel is to remove part of the information in the input, leaving only valid information. When the convolution operation does not depend on future information, it is TCN.

Moreover, it should be noted that TCN is not a single network structure but a type of neural network that is improved based on convolutional neural networks and used to solve sequence problems. It involves the idea of causal convolution, expansion convolution and one-dimensional convolution. Among them, the causal convolution can be understood as: the time series prediction requires that the prediction y_t at time t can only be judged by the input x_1 to x_{t-1} before time t. Similar to the idea of the Markov chain. Compared with traditional convolutional neural networks, causal convolution can only "see" the past data but cannot "see" the future data. Therefore, the information leakage is solved well. One-dimensional causal convolution is generally implemented by Padding, and the front end of the sequence is filled with zeros of the corresponding number of bits. In addition, no padding is performed at the end of the sequence.

The expansion convolution is to add weights with a value of zero to the convolution kernel to broaden the receptive field under the premise of the same amount of calculation. Compared with the traditional convolutional neural network, although the actual size of the convolution kernel remains the same, the size of the receptive field is enlarged. Therefore, the advantage of expanded convolution is that it enlarges the field of perception without loss of information by pooling. It allows each convolution output to contain a larger range of information.

Therefore, through TCN, it is possible for the network to capture the long-term memory of the traffic flow, and it also ensures that only the data of the past moment is

used when predicting the future traffic flow. It avoids the information leakage problem of traditional CNN in this respect.

2.3.4 RNN based traffic flow prediction models

In the scenario of traffic flow prediction, RNN and its variants (LSTM or GRU) can well complete the processing task of sequence data, so they are often used to capture the correlation of traffic flow data in the time dimension.

Chen et al. [80] proposed a new traffic prediction framework based on multiple residual recurrent graph neural networks. In this framework, residual neural networks and GNN are used to solve the problem of gradient explosion and disappearance, as well as the extraction of spatial features. On this basis, the model uses GRU further to extract the temporal dynamic features of high-dimensional feature data. Through the combined application of multiple neural networks, the task of traffic flow prediction is completed. In the study of Cui et al. [81], LSTM was used as a neural unit for temporal feature extraction. The data features as input to LSTM are extracted by Traffic Graph Convolution (TGC) in the original traffic flow data set. Guo et al. [82] proposed an Optimized Graph Convolution Recurrent Neural Network (OGCRNN) model, which captures traffic flow's temporal and spatial characteristics through the combination of GCN and GRU modules. The difference from the previous study is that this study optimizes the traffic flow data periodically during the input phase. The Seq2Seq + Spatial Relation model proposed by Liao et al. [83] still uses the combination of GCN+LSTM. Its characteristic is that according to the characteristics of the actual road network nodes, the data in the data set is filtered and eliminated.

There are many pieces of research in this area, and most of them use LSTM or GRU to extract the temporal dimension features of traffic flow. This work will not repeat them one by one here. In summary, through the above examples, it can be seen that RNN and its variants cannot be applied to prediction tasks alone in the field of traffic flow prediction. Researches use the excellent performance of RNN for time series feature extraction. The temporal features of traffic flow are extracted based on the

high-dimensional feature data captured by other models in advance. Through the form of the combined model, the entire flow forecasting task is completed.

2.4 GCN based Traffic Flow Prediction Models

CNN has been successfully applied to image classification [84], semantic segmentation [85], machine translation [86] and other fields. The underlying data representations in these fields are all grid-like. However, the data involved in many interesting tasks cannot be represented in a grid-like structure. They are located in an irregular domain. This is the case with three-dimensional grids, social networks, telecommunications networks, biological networks, or brain connections. These data can usually be represented in the form of graphs.

Graph Neural Network (GNN) [6, 87], as a generalization of cyclic neural networks, can directly process more general graphs, such as cyclic, directed, and undirected graphs. The graph neural network is composed of an iterative process. First, the iterative process propagates the state of the nodes to the equilibrium state. Then there is a neural network, which generates the final output based on the state of each node. In recent years, graph neural networks have been widely used in various fields such as social networks, knowledge graphs, recommendation systems, and even life sciences. The graph neural network has powerful functions that can model the dependency relationship between graph nodes, so this makes a major breakthrough in the research field related to graph analysis.

2.4.1 Graph theory

Graph theory is the basis of graph neural networks. Before studying the graph neural network, it is of great significance for this work to clarify the related concepts of the graph to be studied. Graphs are one of the most powerful frameworks in data structures and algorithms. Almost all structures or systems such as transportation networks, chess games, and interpersonal interaction networks can be represented by graphs. Generally speaking, a graph can be regarded as an abstract network composed

of "vertices", and each vertex in the network is connected to each other through "edges". That is, the existence of an edge between two points means that the two vertices are related [88].

2.4.1.1 The concept of graph theory

In graph theory, a graph is often written as $G = (V, E)$. That is, a graph is an ordered two-tuple $< V, E >$, marked as G.

(1) $V = \{v_1, v_2, v_3 \dots v_n\}$ is the vertex set of graph G. It is a finite non-empty set whose elements are called vertices or nodes.

(2) $E = \{e_1, e_2, e_3 \dots e_m\}$ is the edge set. It is a finite set, and each element in E has a pair of nodes in V corresponding to it, called an edge.

According to whether the edges are directed or not, graphs can be divided into two categories. The most basic graph is usually defined as an "undirected graph". The edges in the graph are all undirected, and the undirected edge e corresponds to the unordered vertex pair $< u, v >$. u and v are called the two end points of e. In an undirected graph, the degree of a vertex is the number of edges (or arcs) adjacent to the vertex. The corresponding undirected graph is called "directed graph". The edges in the figure are all directed. The directed edge e corresponds to the ordered pair of vertices $< u, v >$. At this time, u is called the starting point of e, and v is the ending point of e. In a directed graph, degrees are divided into "in-degrees" and "out-degrees" according to the direction of the edges. The degree of the vertex is the sum of the in-degree and the out-degree. The number of edges that end at a vertex is called the in-degree of the vertex. The number of edges starting from a vertex is called the vertex's out-degree.

2.4.1.2 Matrix representation of the graph

A graph can be described by definition, or it can be represented graphically. In addition, it can also be represented by a matrix, like a binary relationship. Using a

matrix to represent a graph, it is possible to understand some properties and construction algorithms of the graph through matrix operations. This is also easier for computer processing.

The elements in the degree matrix are the vertex degrees. Except for the main diagonal, all other values are 0. In an undirected graph, the value of the main diagonal of the degree matrix is the degree of the vertex. In a directed graph, only one of the in-degree or out-degree needs to be considered. The value of the main diagonal of the matrix is the in-degree (or out-degree) of the vertex.

The adjacency matrix is a matrix that represents the adjacency relationship between vertices. Similarly, use $G = (V, E)$ to represent the graph. If $< u, v > \in E$, then u and v are called adjacent nodes. If the two points are not adjacent, the relationship between the two points in the adjacency matrix is 0, and the relationship between the two points is 1. Using the adjacency matrix to represent the graph, it is easy to determine whether any two vertices in the graph are connected by edges[89].

Laplace matrix [90] is an important matrix often used in graph theory. It is defined as $L = D - A$, where D is the degree matrix of the graph, and A is the adjacency matrix of the graph. Figure 2.11 shows a simple graph, and Figure 2.12 shows its degree matrix, adjacency matrix and Laplacian matrix. The Laplacian matrix has the following properties:

(1) The Laplace matrix is a positive semi-definite matrix.

(2) The minimum eigenvalue is 0 because the sum of each row of the Laplacian matrix is 0.

(3) The smallest non-zero eigenvalue of the Laplacian matrix is the algebraic connectivity of the graph.

(4) The number of feature values of 0 is the number of connected regions in the graph.

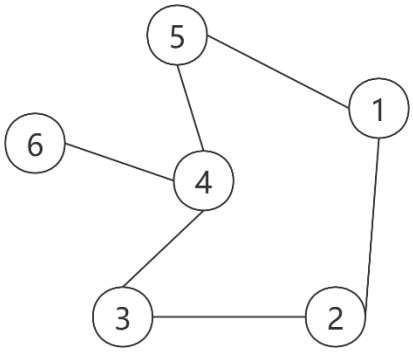


Figure 2.11 Example graph

$$\begin{pmatrix} 2&0&0&0&0&0\\ 0&3&0&0&0&0\\ 0&0&2&0&0&0\\ 0&0&0&3&0&0\\ 0&0&0&0&3&0\\ 0&0&0&0&0&1 \end{pmatrix} \quad \begin{pmatrix} 0&1&0&0&1&0\\ 1&0&1&0&1&0\\ 0&1&0&1&0&0\\ 0&0&1&0&1&1\\ 1&1&0&1&0&0\\ 0&0&0&1&0&0 \end{pmatrix} \quad \begin{pmatrix} 2&-1&0&0&-1&0\\ -1&3&-1&0&-1&0\\ 0&-1&2&-1&0&0\\ 0&0&-1&3&-1&-1\\ -1&-1&0&-1&3&0\\ 0&0&0&-1&0&1 \end{pmatrix}$$

Figure 2.12 The degree matrix, adjacency matrix and Laplacian matrix of Figure 2.11

2.4.2 Graph neural network analysis

Through the overview of graph theory in the above section, the concept of the graph is clarified. According to the concept of graph theory, in computer science, a graph is a data structure composed of two parts: vertices and edges. There are many types of graphs, including undirected graphs, undirected graphs with weights, directed graphs, directed graphs with weights, cyclic graphs. The graph G can be described by the vertex set V and the edges E it contains, that is, $G = (V, E)$. Depending on whether there is a direct dependency between the vertices, the edges can be directed or undirected. In addition, it should be noted that vertices are also called nodes, and these two terms can be interchanged in this work. Figure 2.13 lists several types of graphs.

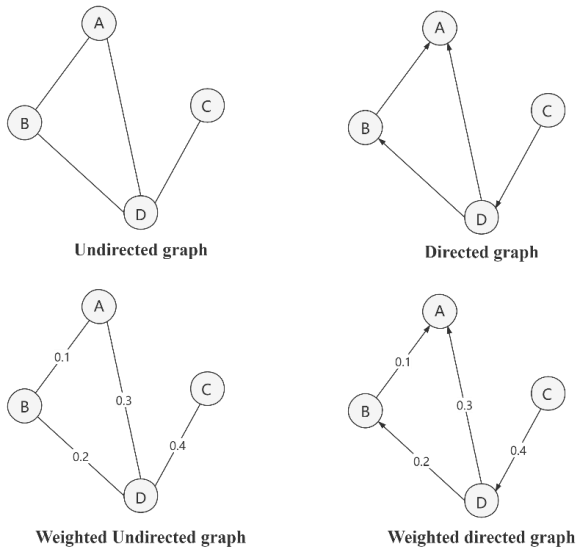


Figure 2.13 Four types of graphs[6]

Graph neural network (GNN) is a kind of neural network that runs directly on graph structure. A typical application of GNN is node classification. GNN is the first algorithm proposed by F. Scarselli et al. in the paper [6], so it is usually considered as the foundation and the beginning of research for GNN. In the node classification problem setting, the feature v_x of each node v is associated with a true label t_v. Given a partially labeled graph G, the task goal is to use these labeled nodes to predict the labels of unlabelled nodes. It learns each node represented by a d-dimensional vector h_v containing neighbourhood information :

$$h_v = f(x_v, x_{co[v]}, h_{ne[v]}, x_{ne[v]}) \quad (2\text{-}13)$$

Where $x_{co[v]}$ represents the feature of the edge connected to v. $h_{ne[v]}$ represents the embedding of adjacent nodes of v. $x_{ne[v]}$ represents the characteristics of the adjacent nodes of v. The function f is a transition function that maps these inputs to a d-dimensional space. To find the unique solution of h_v, the algorithm applies Banach fixed point theorem to rewrite the above equation as an iterative update process.

$$H^{t+1} = F(H^t, X) \quad (2\text{-}14)$$

H and X represent the concatenation of all h and x respectively. The output of the GNN is calculated by passing the state h_v and the characteristic x_v to the output

function g.

$$O_v = g(h_v, X_v) \quad (2\text{-}15)$$

Both f and g here can be interpreted as a feedforward fully connected neural network. L_1 loss can be directly expressed as:

$$loss = \sum_{i-1}^{p}(t_i - o_i) \quad (2\text{-}16)$$

It can be optimized by gradient descent.

However, the original GNN has three main limitations:

(1) If the assumption of "fixed point" is relaxed, then a more stable representation can be learned by using a multilayer perceptron, and the iterative update process can be eliminated. This is because, in the original paper, different iterations use the same parameters of the transfer function f, and different parameters in different layers of MLP allow hierarchical feature extraction.

(2) It cannot handle edge information. For example, different edges in the knowledge graph may represent different relationships between nodes.

(3) Fixed points will hinder the diversity of node distribution and are not suitable for learning to represent nodes.

To solve the above problems, researchers have proposed a graph convolutional neural network (GCN), a variant of GNN. GCN is used to make up for the limitations of GNN.

2.4.3 Overview of GCN

At present, convolutional neural networks have achieved good results in computer vision, natural language processing and other fields. Data that can be gridded, such as images and voices, can be called Euclidean style data. For example, traditional network models (LSTM) or CNN convolutional neural networks can efficiently process this grid-based data and extract features from the data. However, there are a

lot of non-Euclidean data in the real world, such as social network data, biological gene protein data, traffic data. A graph structure can well represent this kind of data. However, machine learning on graphs is a very difficult task because graphs are a very complex but informative data structure.

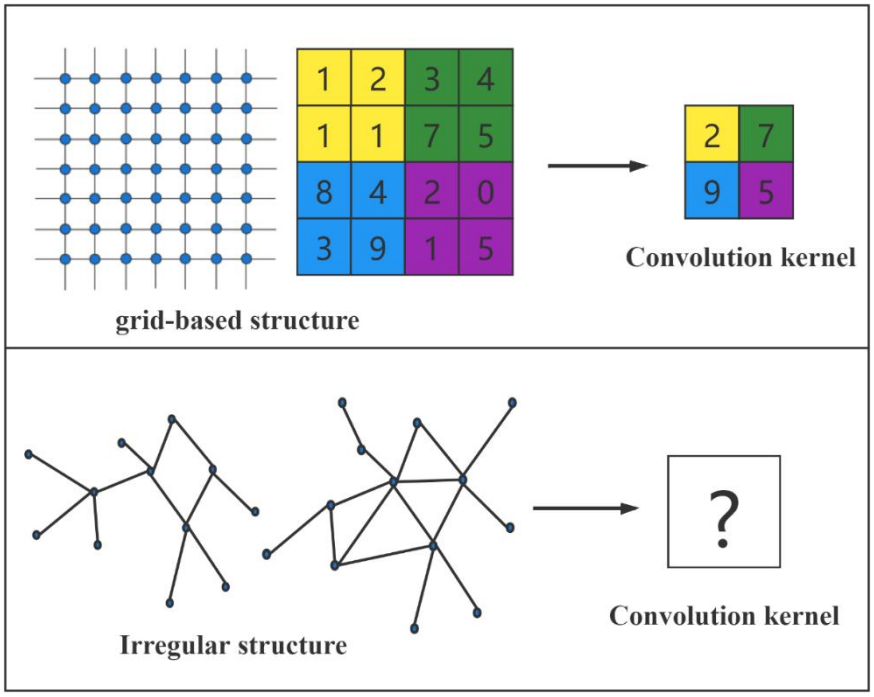


Figure 2.14 The difference between standard convolution and graph convolution

The basic idea of the convolutional neural network is as described in the previous section. It uses the receptive field of the grid data and the parameter matrix of the convolution kernel to perform an inner product operation to obtain the convolution output value of the position, as shown in Figure 2.5. However, when CNN processes the graph structure data shown in Figure 2.14, the number of neighbor nodes of different nodes is not equal. Most of the graphs are heterogeneous, so it is difficult to determine the parameter dimension of the convolution kernel. Arranging the relative position relationship between the weight matrix and the nodes in the neighborhood for effective inner product calculation has become a big problem.

The graph convolutional neural network is designed to be applied to graph data and use graph structure information directly. GCN [91] is a neural network structure that

can effectively mine the features of graph data. For example, social network [92], protein molecular structure [93], communication network [94] and so on. GCN can dig deep into their characteristic laws, and the scope of application is gradually expanded to various fields.

Many scholars have explored the above problems and achieved some results. The mainstream graph convolution methods include spatial methods and spectrogram methods.

The idea of the spatial method is to directly apply the convolution kernel to the nodes and their neighborhoods on the graph. The core of this method is how to properly select the neighborhood of nodes for heterogeneous graph data. Niepert et al. [95] proposed the PATCHY-SAN method, first selecting the candidate node sequence as the central node. Then heuristically select the neighborhood for the central node linearly, map it to a vector and then perform the traditional convolution operation. It has achieved good results in social network tasks. Li et al. [96] introduced graph convolution operations in human action recognition tasks. A variety of division strategies are proposed to divide the neighborhood of nodes into different subsets. By controlling the number of subsets, it is ensured that different nodes can share the weight of the convolution kernel. Cui et al. [97] proposed a high-order graph convolutional recurrent neural network model, which considered the high-order neighbor information of the spatial dimension to learn and predict traffic volume.

The spectrogram method extends the convolution operation on the grid data to the graph structure data through the graph Laplacian matrix. Bruna et al. [98] proposed a general graph convolution framework in 2014 to transform the eigenvectors of the Laplacian matrix into the spectral domain. Then it is approximated by the method of spline interpolation. But this method does not solve the problem of convolution kernel sharing. Subsequently, Defferrard et al. [99] optimized the method, replacing the spline interpolation with the K-order truncated approximate solution of the Chebyshev polynomial. This method realizes parameter sharing in the entire network and proves

that the scope of the convolution kernel is strictly limited to the K-order neighbors of the central node, and at the same time, reduces the complexity of the model.

Many people have tried to use graph convolution methods to solve practical problems in recent years and have achieved some research results. However, the theory is not yet complete. The feature description and specific analysis of graph structure data in specific problems are still being explored.

2.4.4 Principle of GCN

Same as the convolutional neural network. For the feature extraction of graphs, a multilayer neural network structure can also be used. For each layer, the following mapping function can be used to calculate:

$$H^{(l+1)} = f(H^{(l)}, A) \quad (2\text{-}17)$$

Among them, $H^{(l)} \in \mathbb{R}^{N \times d^{(l)}}$ represents the node-level expression of the l-th layer graph. It is an $N \times d^{(l)}$-dimensional matrix. N represents the number of nodes, and $d^{(l)}$ represents the dimension expressed by the nodes of the lth layer (the dimension expressed by the nodes can be different in each layer, which is determined by f and can be flexibly set). $H^{(0)} = X$ represents the initial node expression matrix of the 0-th layer. Assuming that there are a total of L-layer networks, $H^{(L)} = Z$ represents the node expression matrix output by the last layer.

Similar to CNN, graph convolution also uses shared weights. However, unlike CNN, the weight of each kernel is a regular matrix, which is assigned according to the corresponding position. The weight in the graph convolution is usually a set. When calculating the aggregate feature value for a node, all points participating in the aggregation are allocated to multiple different subsets according to a certain rule. The nodes in the same subset adopt the same weight to realize weight sharing.

In a nutshell, the graph convolution operation weighs each node's features and the features of its neighbor nodes and then propagates to the next layer. This kind of

graph convolution operation is called graph convolution in the spatial domain, and it has the following characteristics:

(1) As the number of layers deepens, the farther features each node can aggregate.

(2) The weight is shared and will not be specific to each node, which is the same as traditional CNN. Intuitively, if the weight is different from node to node. Then once the graph structure changes, the weight will immediately become invalid.

(3) The number of neighbor nodes of each vertex may be different, which leads to more significant eigenvalues of vertices with more neighbor nodes.

(4) When calculating the adjacency matrix, the characteristics of the node itself cannot be included in the aggregated eigenvalues.

In addition, to overcome the shortcomings of spatial graph convolution, scholars proposed graph convolution in the spectral domain [91, 99]. The idea is to use the Laplacian matrix and Fourier transform of the graph to perform convolution operations. This method is also the method used in this research to extract spatial features, and the details will be described in the next chapter.

2.4.5 GCN based traffic flow prediction models

There is a great correlation between traffic conditions and the road traffic network, so recently, researchers formulated the traffic flow prediction model as a graphical modeling problem. Because of the spatial characteristics of road networks, GCN is more suitable for modeling non-Euclidean spatial structure data than CNN. It is also more suitable for extracting spatial feature data from the structure of the traffic road network. Through spatial graph convolution or spectral graph convolution, GCN can aggregate the neighbor node information of each road network node. Combine this information with its own node information and update it to high-dimensional node feature information. The upgraded feature data contains richer node features, enabling the neural network to capture deeper feature rules.

Diao et al. [45] used a dynamic Laplacian matrix estimator to decompose real-time traffic data into global components that are stable and depend on long-term temporal and spatial traffic relationships. This method is used to extract spatial features that change over time. Thus, the characteristics of the time dimension are obtained through GCN in the space dimension. Extracting traffic flow data features in temporal and spatial dimensions respectively is also an important idea for applying deep learning models in Wu et al. [100]. They proposed integrating WaveNet into GCN for spatial-temporal modeling to form a Graph WaveNet model, which can handle long sequences. Song et al. [101] proposed a spatial-temporal synchronization graph convolutional network (STSGCN) model. This model mainly divides the data into different time periods in the time dimension. The feature information extracted by GCN in different time periods is spliced together in the time dimension, and then further spatial-temporal correlation extraction is performed through the neural network. Seo et al. [40] proposed Graph Convolutional Recurrent Neural Network (GCRN), a model that combines recurrent network and graph convolution operations. Subsequently, Li et al. [42] proposed a DCRNN (Diffusion Convolutional Recurrent Neural Network) model that successfully used GRU and graph convolution for long-term traffic prediction. Yu et al. [43] proposed a GCN with a gating mechanism and applied it to the traffic volume prediction problem. Guo et al. [18] used the GCN spectrogram method to extract spatial features from the original data. Then input it into the CNN model to further extract temporal features. On this basis, through the stacking of spatiotemporal modules, higher-dimensional spatiotemporal feature information can be obtained.

In summary, the application of graph neural network in the field of traffic flow prediction needs to fully consider the time dimension or time series. In other words, the application of graph neural networks to traffic prediction problems must fully consider the complex time-space correlation in a road traffic network. At the same time, in the modeling process, the relationship between various model combinations must be considered. Special attention is needed to avoid the problem of error

propagation between steps brought about by the step-by-step generation of prediction results.

2.5 Enhancement Mechanism of Models

2.5.1 Deep residual network

The residual network was proposed in 2015, and the core is to solve the side effects caused by increasing the depth of the network. Once it was born, it has achieved good results in image classification, detection, and positioning based on ImageNet data [102]. Through in-depth research on the residual network. Researchers found that it is not that the deeper the network, the better the effect of the neural network. Researchers found through simulations that as the number of network layers continues to increase, the model's accuracy will continue to improve. But when the level of the network increases to a certain number, the training accuracy and testing accuracy begin to decline rapidly. This shows that when the network becomes very deep, it will become more difficult to train it.

In theory, regardless of the depth of the network model, it is possible to approximate the intrinsic relationship and essential characteristics of the data through functions. But when solving complex problems in real situations, the number of computing units required increases exponentially. Shallow networks often have the problem of insufficient function expression capabilities, while deep networks may only require fewer computing units. However, the network hierarchy is not as deep as possible. With the increase of network levels, on the one hand, the excessive number of layers will lead to the wasteful occupation of video memory and "eating" computing power. On the other hand, problems such as overfitting, gradient dispersion, and network degradation will also occur. However, the deep residual network can well solve the degradation problem caused by increasing the network depth.

There are two main designs for the residual network: shortcut connection and identity mapping. The shortcut connection makes the residual possible, and the identity

mapping makes the network deeper. The structure of the residual block is shown in Figure 2.15.

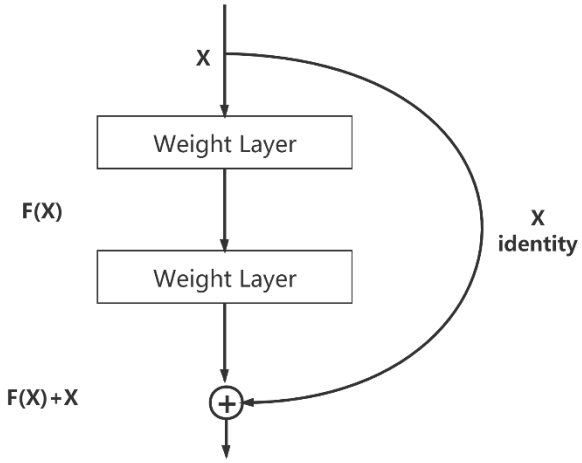


Figure 2.15 Residual learning unit

One or more layers of networks can be skipped using this quick connection method. The input result before the jumped layer is directly used as the output result of the stacked layer. The shortcut connection method neither adds additional parameters nor increases the computational complexity. Its existence can still use backpropagation stochastic gradient descent to train the entire network end-to-end. Moreover, it is easy to use public libraries to implement without modifying any content [103].

2.5.2 Attention mechanism

The attention mechanism is a deep learning technique that imitates humans paying more attention to important areas when observing things [104]. The attention mechanism was first applied to the image classification task by the Google DeepMind team, which can accurately identify multiple objects in the image, reducing the error rate of the MINIST classification task by 4% [105]. The effectiveness of the attention mechanism in image classification tasks is demonstrated. Later, Bahdanau et al. [106] applied the attention mechanism to machine translation tasks for the first time based on previous work. The Google machine translation team proposed a sequence network model Transformer [107] based on the attention mechanism, which attracted widespread attention. The attention mechanism has become one of the commonly

used techniques in deep learning. In deep learning, different weights are assigned to different parts of interest so that the model can effectively learn important information. That is beneficial to the task and filter noise information.

2.5.2.1 Definition of attention mechanism

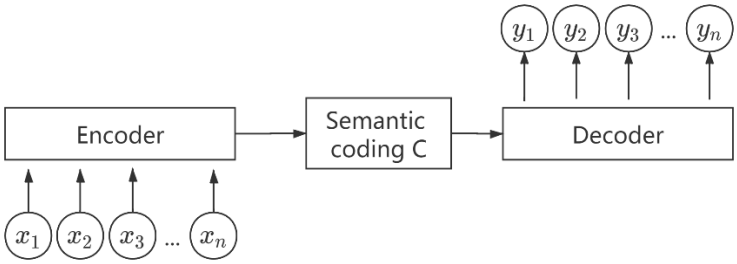


Figure 2.16 Encoder-Decoder abstract framework diagram[43]

In general, most attention mechanisms are implemented based on the Encoder-Decoder framework structure. The abstract framework diagram of Encoder-Decoder is shown in Figure 2.16. The model maps an input sequence of variable length $X = (x_1, x_2, \dots, x_n)$ to another output sequence of variable length $Y = (y_1, y_2, \dots, y_m)$. Among them, the encoder compresses the information of the entire input sequence into an intermediate semantic vector C. The decoder generates y_i through the intermediate semantic vector C and the generated output sequence $y_1, y_2, \dots, y_{i-1}$, that is, $y_i = f(y_1, y_2, \dots, y_{i-1}, C)$. f represents the decoding function. In the entire decoding process, the intermediate semantic code C is a constant, which is shared in the decoding process at each time step. This results in the output sequence y_i having no discrimination for each x_i in the input sequence. To solve this problem, Bahdanau introduced an attention mechanism in the Encoder-Decoder structure [106], the structure diagram is shown in Figure 2.17. Among them, s_{t-1} is the hidden vector at step $t-1$ in the decoding process, $s_t = f(s_{t-1}, y_{t-1}, C_t)$. C_t represents the semantic vector corresponding to step $t-1$, $C_t = a_{t,1}h_1 + a_{t,2}h_2 + \cdots + a_{t,T}h_T$. Where T is the length of the input sequence, h_j represents the hidden vector of the j-th step in the encoding process, and $a_{t,j}$ represents the degree of C_t's attention to the hidden vectors of different time steps. The calculation method of $a_{t,j}$ is shown in formulas (2-18) and

(2-19).

$$a_{t,j} = a(s_{t-1}, h_i) \quad (2\text{-}18)$$

$$a_{t,j} = \exp(a_{t,j}) / \sum_{j=1}^{T} \exp(a_{t,j}) \quad (2\text{-}19)$$

Where a represents the alignment model, which is used to calculate the degree of alignment or influence of h_j on s_{t-1}. Then $a_{t,j}$ is normalized by SoftMax. Under normal circumstances, parameterize a and participate in training together in the neural network. The four commonly used alignment methods [108] are shown below.

(1) Additive Attention

The alignment shown in formula (2-20) is often used in additive attention, where w and w_a are both parameter matrices.

$$a(s_{t-1}, h_j) = w^T \tanh\,(W_a[s_{t-1}: h_j]) \quad (2\text{-}20)$$

(2) Multiplicative Attention

Multiplicative attention often uses the alignment shown in formulas (2-21) and (2-22), where w_a is the alignment matrix.

$$a(s_{t-1}, h_j) = s_{t-1} W_a h_j \quad (2\text{-}21)$$

$$a(s_{t-1}, h_j) = s_{t-1} h_j \quad (2\text{-}22)$$

(3) Multi-layer perceptron attention (MLP) Attention

The attention of MLP often adopts the alignment shown in formula (2-23).

$$a(s_{t-1}, h_j) = \sigma(w^T \tanh(W_a[s_{t-1}: h_j] + b_1) + b_2) \quad (2\text{-}23)$$

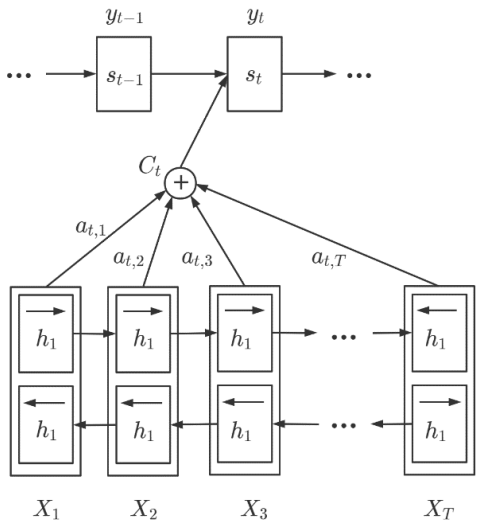


Figure 2.17 Attention mechanism structure diagram

2.5.2.2 Types of attention mechanism

Nowadays, the idea of the attention mechanism has been widely used in various deep learning tasks. Such as machine translation [49], image caption generation [109], question answering system [110], speech recognition [111]. The idea of the attention mechanism has multiple implementation methods in different application scenarios. According to its implementation, its structure category can be roughly divided into two categories. One is the basic attention mechanism structure, and the other is the combined attention mechanism structure. In the basic attention mechanism structure, there are mainly soft attention mechanism, hard attention mechanism, and self-attention mechanism. Some scholars further proposed local attention mechanism and global attention mechanism based on soft attention and hard attention [112]. In the structure of the combined attention mechanism, there are mainly Co-Attention [110], Attention-Over-Attention [97], Multi-Head Attention [107].

2.5.2.3 Overview of graph attention (GAT) mechanism

Graph Attention Network (GAT) [113] is a new type of neural network structure, which is very important in GNN. The introduction of the attention mechanism based on the graph convolutional network GCN is very practical. It operates on the graph structure data and uses the attention layer to solve the previous shortcomings based on

graph convolution. By stacking layers, nodes can pay attention to their neighborhood characteristics. Implicitly assign different weights to different nodes in a neighborhood. Moreover, it does not require any expensive matrix operations, such as inversion, and does not require prior knowledge of the graph's structure. In this way, GAT simultaneously solves several key challenges of spectrum-based graph neural networks and makes the model easy to apply to induction and conversion problems.

GAT introduces an attention-based architecture, which can handle node classification tasks where the data structure is a graph. Its idea is to update the vector representation of each node by paying attention to the neighbor node characteristics of each node. The attention structure has the following outstanding characteristics: (1) The operation is efficient because it can be parallelized across pairs of nodes. (2) By assigning arbitrary weights to adjacent nodes, it can be applied to graph nodes of different degrees. (3) The model is directly applicable to inductive learning problems. The model can be extended to graph tasks where the structure is completely invisible.

2.5.2.4 Enhancement through the self-attention mechanism

In many sequence-based tasks, the attention mechanism has almost become the de facto standard [106, 114]. One of the benefits of the attention mechanism is that it allows the processing of variable size input. Focus on the most relevant part of the input to make a decision. When an attention mechanism is used to calculate the representation of a single sequence, it is usually called self-attention or internal attention. With RNN or CNN, self-attention is useful in tasks such as machine reading [72] and learning sentence representation [115]. Vaswani et al. [107] showed that not only self-attention can improve methods based on RNN or CNN, but it is also sufficient to build a powerful model to obtain the most advanced performance in machine translation tasks.

In recent years, the attention mechanism has also been applied in the field of traffic flow prediction. Chen et al. [116] designed a multi-range attention two-component GCN (MRA-BGCN) model. This model introduces a multi-range attention

mechanism to aggregate information in different adjacent areas and automatically learn the importance of different areas. On this basis, the weighted traffic information of adjacent areas is used as the input of the graph convolutional neural module. Guo et al. [18] also used graph attention mechanisms in the task of traffic flow prediction. In their proposed Attention Based Spatial-Temporal Graph Convolutional Networks (ASTGCN) model, the temporal attention mechanism and the spatial attention mechanism are respectively applied before the temporal and the spatial module to capture the dynamic dependence of time and space, respectively. In addition, in the field of traffic flow prediction, Geng et al. [117] and Li et al. [118] respectively introduced the Masked Multi-head self-attention mechanism in their research. Furthermore, in the research of Li et al. [119] and Park et al. [120], the GAT mechanism has also been fully discussed and applied.

In general, the attention mechanism can effectively mine the dynamic spatial-temporal patterns of traffic flow data, which is very suitable for the application scenarios of traffic flow prediction problems. In the studies mentioned above, the models with the attention mechanism often show better performance than those without this mechanism. Moreover, the computational complexity of the attention mechanism is low and will not cause additional computational overhead to the model. Therefore, it has received more and more attention from researchers.

2.6 Discussion

Through the analysis of different types of prediction methods, some research gaps can be found. Due to the random, dynamic and non-linear characteristics of traffic flow data and insufficient flexibility of the models, part of the existing models lack a reasonable understanding and interpretation of the data, resulting in low prediction accuracy. Due to the lack of consideration of the periodic characteristics of traffic flow data, some deep learning models cannot reasonably learn the periodic characteristics of traffic flow data. Most existing traffic flow prediction methods modeled from the temporal dimension do not fully consider the direct correlation

between the traffic flow sequences in the spatial dimension. The model capacity is limited, and it is not easy to expand. The combined deep learning models ignore the high-order correlation between the feature representations of different nodes on the graph structure. In addition, the spatial-temporal correlation feature extracted based on the relationship between nodes lacks consideration of the dynamics between them.

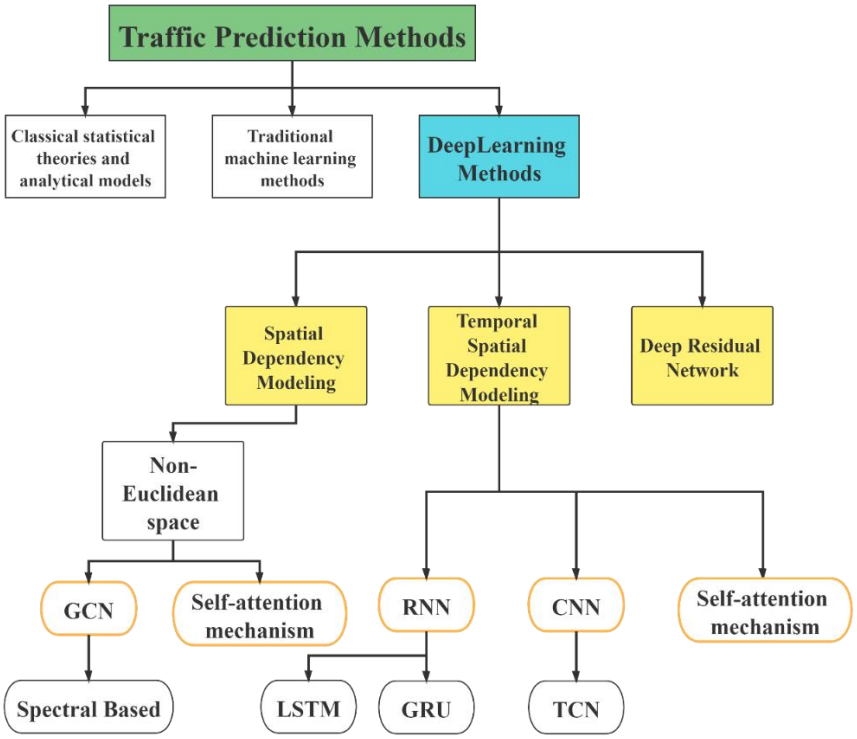


Figure 2.18 Category comparison of traffic flow prediction research

According to the survey in this chapter, this work draws a category comparison of traffic flow prediction research in Figure 2.18 to describe the latest research focus in the field. These latest methods are indicated to be the most effective way to fill these gaps. It can be seen from the figure that in recent years, research in the field of traffic flow forecasting has mainly focused on three directions. Deep learning is considered a better prediction method than Classical statistical models and traditional machine learning methods. In the deep learning model, the spectral method of GCN is demonstrated to have the best performance in terms of spatial dependence extraction. RNN variants (GRU, LSTM) are reported to have the best performance in terms of time-dependent extraction. CNN's TCN model is also proved to have certain

advantages in capturing time features. In addition, researchers are also concerned with the attention mechanism and deep residual network because they have very good effects on the optimization of the model and the improvement of calculation accuracy.

In summary, the research on the deep learning traffic flow prediction model is still in the stage of theoretical research and preliminary testing. Especially based on the research of graph neural networks. Moreover, most models focus on research in a certain aspect, and there are relatively few comprehensive research models. By comparison, Guo et al. [18]'s research is more advanced in many prediction models, and its performance is also more prominent. Therefore, this work plan uses its research as a comparison and reference. On this basis, this thesis plans to improve the prediction accuracy by improving the spatial-temporal prediction model and data processing method in the modeling chapter.

2.7 Summary

This chapter details the basic theories and research results related to the research content of traffic flow forecasting. Explains the understanding of spatial-temporal data, including basic concepts and the mining of temporal and spatial data. In addition, it deeply explores the related theoretical knowledge of deep neural networks and the research of these theories and models in the field of traffic flow forecasting. Including the discussion on the application of convolutional neural networks, recurrent neural networks, graph convolutional neural networks in this field. On this basis, this study also explored the feasibility and performance of deep residual networks and attention mechanisms as optimization methods for predictive models in this field. Finally, in the discussion session, some advanced traffic flow prediction methods were compared. Through analysis, this thesis found the research gap and the direction that can be improved in this field. The traffic flow data processing and improved traffic flow prediction model we proposed will be presented and discussed in detail in the next chapter.

Chapter 3

Traffic Flow Prediction Modeling

In Chapter 2, through the analysis of different prediction methods, some research gaps can be found. First, the existing models have insufficient flexibility and interpretability of traffic flow data with random, dynamic, and nonlinear characteristics, resulting in low prediction accuracy. Second, the models and methods lack the extraction of periodic features of traffic flow data, which leads to the inability to learn more powerful feature data of traffic flow data reasonably. The third is that the existing traffic flow prediction methods that model from the time dimension do not fully consider the correlation between the traffic flow sequences in the space dimension, which makes the general applicability of the model insufficient. Fourth, the combined deep neural network model ignores the characteristics of the graph structure of the traffic road network and cannot express the high-order correlation between different nodes. Finally, although the latest traffic flow prediction application models have improved feature capture and scalability, they often have insufficient capabilities in capturing the dynamic correlation between nodes.

In this chapter, we propose two types of traffic flow prediction models. These two types of models are mainly used to solve the last research gaps and improve the prediction accuracy of the models. These two types of models are mainly based on the GCN+RNN models and the GCN+CNN models. In addition, the attention mechanism and some effective machine learning optimization methods have also been applied to improve the model's performance. In the next few subsections of this chapter, the research problem is first explained. Second, we use a framework graph to explain our proposed model's detailed structure, some traffic flow data processing methods, and model optimization methods. Finally, according to the framework structure, the details of our proposed models are elaborated.

3.1 Problem Statement

The purpose of traffic flow prediction is to predict the future traffic flow based on the traffic flow observed by N related sensors on the road network. The use of our deep learning models for traffic flow prediction requires a large amount of road traffic data support in the real world. The sensors on the road network can be represented as a figure 3.1, or as shown in formula (3-1):

$$G = (V, E) \qquad (3\text{-}1)$$

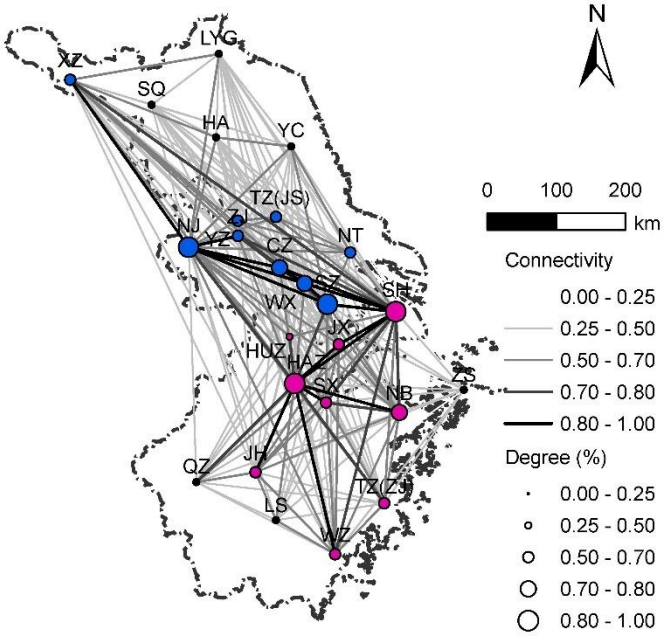


Figure 3.1 Inter-city connectivity in the composite infrastructure network[3]

Among them, V is the node set $|V| = N$, and E is the edge set. The relationship between the sensors can be embodied by different graph neural network message transmission methods according to actual needs. Express the traffic flow observed on graph G as a graphical signal $X \in \mathbb{R}^{N \times P}$, where P represents the number of features of each node, and let $X^{(t)}$ represent the graphical signal observed at time t. Then the goal of the traffic flow prediction problem is to learn a function $h(\cdot)$ to map the graphical signal at the historical time $T'$to the graphical signal at the future time T. Given the graph G, the traffic flow prediction problem can be represented by Figure 3.2.

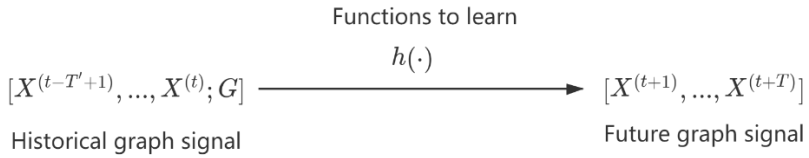


Figure 3.2 Illustration of traffic flow prediction problem[12]

Furthermore, the input signal X should include two important features. In the spatial dimension, it is expressed as a collection of basic parameters of traffic flow at a certain moment. In the temporal dimension, it is expressed as a collection of basic parameter data of traffic flow in a time sequence. We use three important parameters (traffic flow, traffic flow density, and average vehicle speed) to describe the traffic operation status and traffic flow's temporal and spatial features. This means that P in $X \in \mathbb{R}^{N\times P}$ is 3. The definitions of these three parameters are as follows:

Definition 1: Traffic flow. The total number of vehicles passing through a certain section in a unit time. Its expression is shown in formula (3-2).

$$Q = KV \quad (3\text{-}2)$$

Among them, Q represents the flow rate. V represents speed. K represents density.

Definition 2: Traffic flow density. Traffic flow density generally refers to the density of motor vehicles in a lane. Its expression is shown in formula (3-3).

$$K = \frac{N}{L} \quad (3\text{-}3)$$

Among them, N represents the number of vehicles in the road segment. L represents the length of the road section.

Definition 3: Average vehicle speed. The ratio of the length of a certain road segment to the average travel time of all vehicles passing through the road segment. Its expression is shown in formula (3-4).

$$\bar{V}_s = \frac{S}{\frac{1}{n}\sum_{i=1}^{n} t_i} = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} \frac{1}{v_i}} \quad (3\text{-}4)$$

Among them, $\bar{V}_s$ represents the interval average speed. S represents the length of the road section. t_i represents the travel time of the i-th vehicle. n represents the number of vehicles observed. v_i represents the travel speed of the i-th vehicle.

3.2 Models Framework

In sections 2.6 and 3.1, the existing research gaps and traffic flow prediction problem statements are discussed separately. It can be seen from the analysis and discussion that our proposed model needs to realize the task of traffic flow prediction and improve the prediction accuracy. That is, the traffic flow value for a period of time T after the time t:($X^{(t+1)}, \dots, X^{(t+T)}$) is predicted by inputting the traffic flow data for a period of time T'before the time t:($X^{(t-T'+1)}, \dots, X^{(t)}$). From a model point of view, it is to solve several shortcomings of the existing models. One is to improve the interpretability of input data. The second is to extract the periodic characteristics of traffic flow data effectively. The third is to improve the deep learning model's ability to capture high-order correlations between space and time. The fourth is to effectively extract the dynamic associations between nodes in the traffic road network structure. To solve the above problems, the model framework we proposed is shown in Figure 3.3.

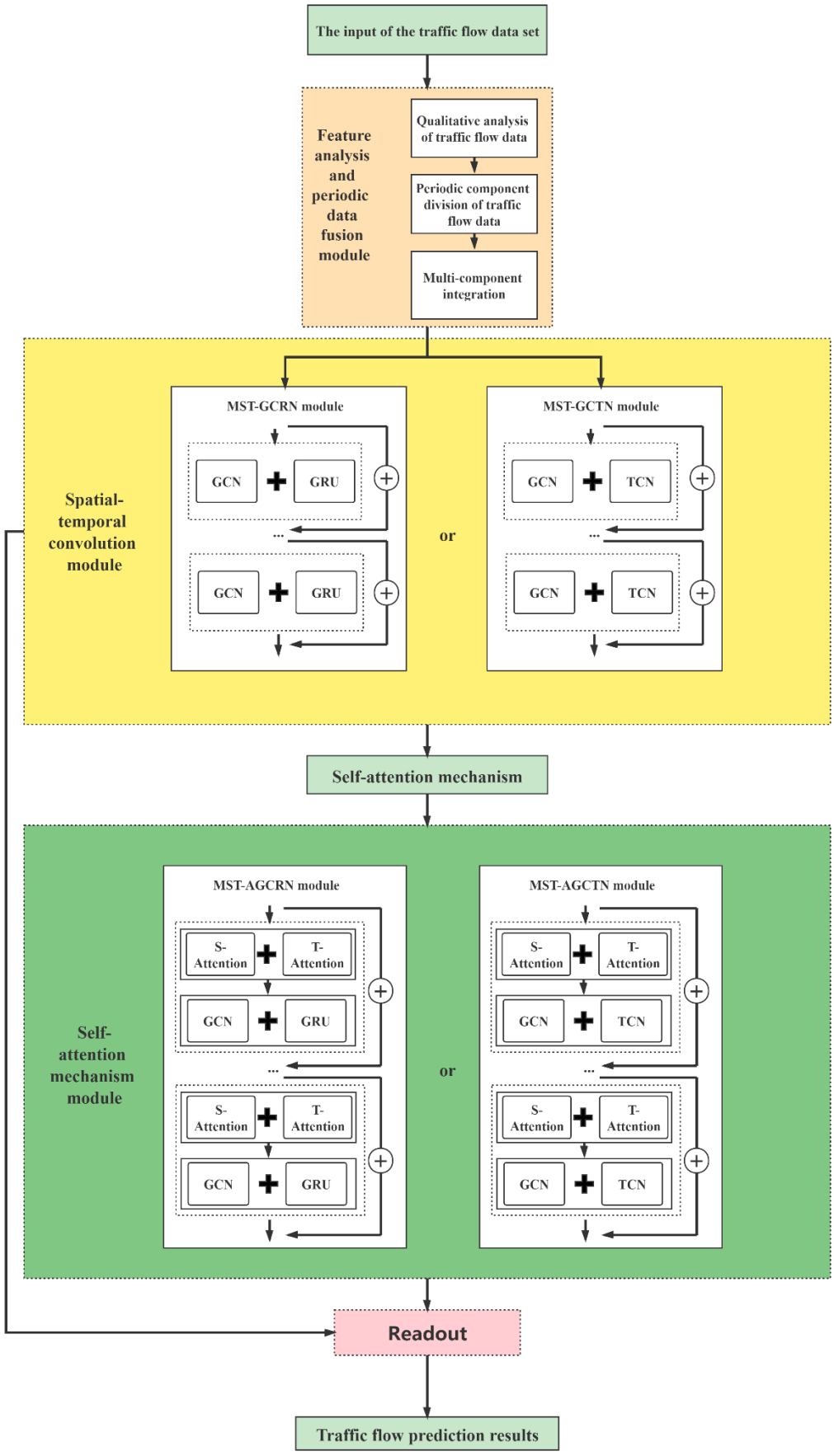


Figure 3.3 The framework of Models proposed

The models we proposed are mainly composed of three modules. The first module is the feature analysis and periodic data fusion module. Through the first and second parts of this module, the composition of the characteristic data input by the model is determined based on Pearson correlation analysis and periodic analysis of the data. The multi-dimensional time series data is mapped on a time axis through the third part of the data fusion method. The second module is two models based on spatial-temporal graph convolution. In this module, this thesis uses MST-GCRN (Multiple layers Spatial-temporal aware Graph Convolutional Recurrent Network model) and MST-GCTN (Multiple layers Spatial-temporal aware Graph Convolutional and Temporal Convolutional Network model) to extract high-dimensional spatiotemporal features of the data processed by the previous module. This is also the embedding process of feature data. The third module is two improved models that have added a self-attention mechanism. MST-AGCRN and MST-AGCTN based on the self-attention mechanism are used to extract more powerful high-dimensional spatial-temporal features of traffic flow data. The main principle is to capture the dynamic spatial-temporal correlation between nodes by adding a spatiotemporal self-attention module between the input data and the machine learning module. Finally, all four models we proposed can input the extracted feature data into the Readout module for traditional machine learning training and prediction tasks.

Through these three main modules, the problems raised in section 3.1 can be effectively solved. The differences between the models proposed in this study and previous research models are: 1. Data qualitative analysis and fusion technology improve the quality and reliability of input data. This enables the model to fuse feature data of different dimensions into a time dimension during the input stage. This simplifies the calculation process, enhances the connection between components, and makes it more scalable. 2. The models use GRU and TCN to capture data features in the temporal dimension. Through the advantages of GRU and TCN in sequence data processing, the interpretability and accuracy of the model in the time dimension are improved. 3. By adding a two-dimensional self-attention block of time and space to

the model, the weight of the node can be more reasonably distributed. Thereby, the dynamic spatial-temporal correlation on the transportation network can be captured more accurately. The following sections of this chapter will explain each module in detail.

3.3 Traffic Flow Data Feature Analysis and Periodic Spatiotemporal Data Fusion

3.3.1 Periodic quantitative analysis of traffic flow

This section uses Pearson correlation analysis to prove the periodicity of traffic flow data quantitatively. Pearson correlation analysis can intuitively demonstrate which traffic flow data in the past ($X^{(t-T'+1)}, \dots, X^{(t)}$) plays a key role in predicting future data ($X^{(t+1)}, \dots, X^{(t+T)}$).

In this work, according to the periodic division method commonly used in traffic flow prediction, the traffic flow input data is divided into three components according to the periodicity. The three periodic components are the recent time segment, the daily periodic time segment, and the weekly periodic time segment in sequence. Recency periodicity describes the similarity of traffic flow at a specific time during the day. Daily periodicity describes the similarity of traffic flows at specific times of the day. Weekly periodicity describes the similarities in traffic flow at specific times of the week. To further confirm that the traffic flow data set we used meets the three periodic characteristics. This work conducts a Pearson correlation analysis based on real data. The value obtained by Pearson correlation analysis can be represented by a heat map. Therefore, it is possible to intuitively prove the periodic characteristics of the traffic flow data and the closeness of each other.

Pearson correlation analysis refers to the analysis of two or more correlated variables to measure the closeness of the correlation between the variables. Many indicators characterize the degree of correlation between variables, and the Pearson correlation

coefficient is currently more commonly used.

In terms of traffic flow prediction, the Pearson correlation coefficient r between historical data X and forecast data P can be calculated as shown in formula (3-5):

$$r = \frac{\sum_{i=1}^{T}(X_i-\bar{X})(P_i-\bar{P})}{\sqrt{\sum_{i=1}^{T}(X_i-\bar{X})^2(P_i-\bar{P})^2}} \qquad (3\text{-}5)$$

Among them, T indicates the prediction duration, P indicates the time series data of the traffic flow to be predicted, and X indicates the time series data of the historical traffic flow. Moreover, $\bar{X}$ represents the mean value of the random variable X, and $\bar{P}$ represents the mean value of the random variable P. Then the value of the Pearson correlation coefficient r indicates the degree of correlation between the sequence X and the sequence P, and the value range is between -1 and 1. The closer the absolute value of the correlation coefficient is to 1, the stronger the correlation between the variables; conversely, the closer the correlation coefficient is to 0, the weaker the correlation. The specific correlation degree division is shown in Table 3-1.

Table 3-1 Correspondence table of Pearson coefficient and correlation degree

PEARSON CORRELATION COEFFICIENT	CORRELATION DEGREE
|R|≥0.8	Highly correlated
0.5≤|R|<0.8	Moderately related
0.3≤|R|<0.5	Low correlation
|R|<0.3	Basically irrelevant

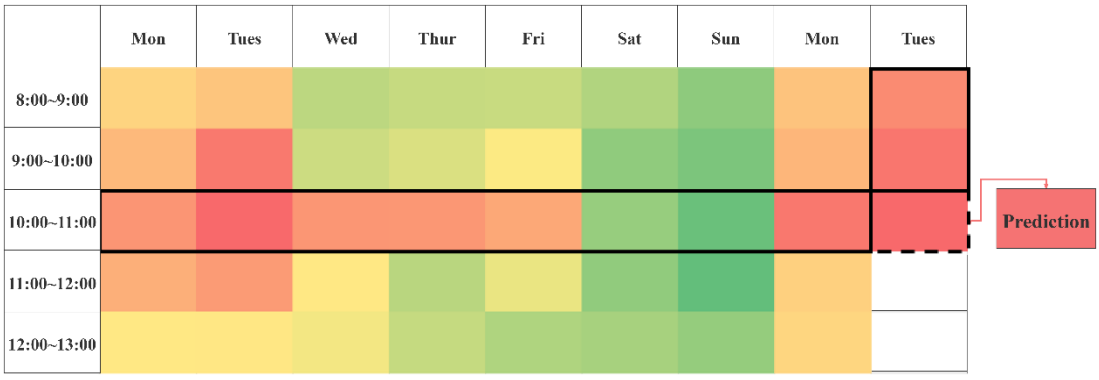


Figure 3.4 Periodic dependence degree heatmap

Figure 3.4 is a heat map of the Pearson correlation coefficient between the predicted

target in a certain period and all other periods in the representative real data set that we obtained through calculation. The deeper the red, the greater the correlation value. Each square in the figure represents 60 minutes. It can be seen that, for the selected control forecast data, the two closest periods (data 120 minutes before the forecast node) are the most relevant to it. The characteristics are very obvious, and the closer to the forecast period, the greater the value of the Pearson correlation coefficient. If the time period is more detailed, its gradual characteristics will be more obvious. The contrast is also very obvious for daily periodicity and weekly periodicity (horizontal black frame part). The characteristic of daily periodicity is that its Pearson correlation coefficient value is the largest in the same time period of the previous day. This value gradually changes gradually according to the distance according to the distance in the selected period of the previous day. For a week, since city residents work from Monday to Friday and rest on weekends, people's travel patterns during the week will be similar during the working days from Monday to Friday. The travel patterns of non-working days, such as Saturdays and Sundays, are more similar. Therefore, the Pearson correlation coefficient values of the same period of working days and non-working days are significantly different from the selected time period. The corresponding value will gradually change in the horizontal direction on different dates according to the distance from the selected date. It should be noted that the selected period is a working day period. Therefore, according to the difference between working days and rest days, the difference can be clearly shown in the heat map.

The results obtained through Pearson correlation analysis are used as the objective basis for this research to confirm our division of periodic components. At the same time, it also verifies the correctness and universal applicability of the data set we have selected.

3.3.2 Periodic component division of traffic flow data

This thesis divides the input traffic flow data along the horizontal time axis based on

Pearson correlation analysis. The periodic characteristics of the input data are shown in Figure 3.4. Assuming that the collection frequency of the data collection equipment deployed on the transportation network is q times/day, that is, each node collects q data values every day. At this node, set the current time to t_0 time, and then specify the length of the future time segment that needs to be predicted as T. That is to say, the prediction duration T is used as the time window unit. Then this study can divide the input flow data x along the horizontal time axis according to the recent periodicity, daily periodicity and weekly periodicity. The method is to intercept three time segments with a length of an integer multiple of T respectively along the horizontal time axis as the input of the three periodic components. The three input time segments can be represented as T_h, T_d, and T_w, respectively. Then the input data can be expressed as x_h, x_d and x_w. Among them, the subscript h represents the recent periodic hour. The subscript d represents the daily periodicity-day. The subscript w represents the weekly periodicity-week.

The specific representations of the three time series segments are as follows:

(1) The recent components

As shown in the Pearson correlation analysis section. At a certain node, the traffic flow in the predicted time segment T_0 will inevitably be affected by the traffic flow at the previous moment of the day or the previous stage of the day. The most obvious example is the influence of different periods of the working day on the traffic flow in the next period. Such as traffic accidents, weather changes. And it can be clearly seen that such an impact will gradually weaken or increase with the distance from the prediction period. Its expression is shown in formula (3-6).

$$X_h = (X_{t_0-T_h+1}, X_{t_0-T_h+2}, \dots, X_{t_0}) \quad (3\text{-}6)$$

(2) The daily periodic components

The daily periodicity component can be expressed as the correlation between the data in the predicted time period and the traffic flow data in the same time period in the

previous few days. Because there is a strong similarity between the traffic flow data of the previous few days and the data of the forecast period. Therefore, the data in this time segment in the previous few days can be used as the basis for predicting the data in the T_0 time period. For example, during the working day, subject to the cyclical changes of morning and evening peak traffic flow, the changing trend of its traffic flow data will show a trend of convergence. The daily periodic component models the traffic flow data in units of days. Its expression is shown in formula (3-7).

$$X_d = (X_{t_0-(T_d/T)*q+1}, \cdots, X_{t_0-(T_d/T)*q+T}, X_{t_0-(T_d/T-1)*q+1}, \cdots,$$

$$X_{t_0-(T_d/T-1)*q+T}, \cdots, X_{t_0-1*q+1}, \cdots, X_{t_0-1*q+T}) \quad (3\text{-}7)$$

(3) The weekly periodic components

The weekly periodicity component can be expressed as the correlation between the data in the predicted time period and the traffic flow data in the same time period in the previous few weeks. Because there is a strong similarity between the traffic flow data of the previous few weeks and the data of the forecast period. Therefore, the data in this time segment in the previous few weeks can be used as the basis for predicting the data in the T_0 time period. For example, during the working day of this week, the cyclical change of the morning and evening peak traffic flow is similar to the morning and evening peak traffic flow data of the previous week or earlier. The weekly periodic component models the traffic flow data on a weekly basis. Its expression is shown in formula (3-8).

$$X_w = (X_{t_0-7*(T_w/T)*q+1}, \cdots, X_{t_0-7*(T_w/T)*q+T}, X_{t_0-7*(T_w/T-1)*q+1}, \cdots,$$

$$X_{t_0-7*(T_w/T-1)*q+T}, \cdots, X_{t_0-7*1*q+1}, \cdots, X_{t_0-7*1*q+T}) \quad (3\text{-}8)$$

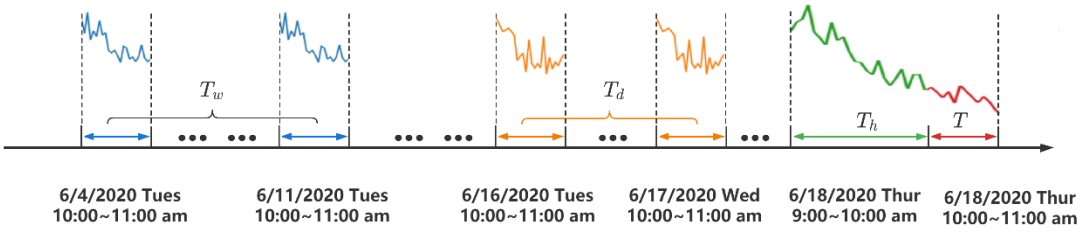


Figure 3.5 An example of periodic factor input

Take the real data set used in this thesis as an example. It can be seen from Figure 3.5 that the green traffic flow data corresponds to the recent component T_h. The orange traffic flow data corresponds to the daily periodic component T_d. The blue traffic flow data corresponds to the weekly periodic component T_w. The prediction of traffic flow data from 10:00 to 11:00 on June 18, 2020, requires the input of data corresponding to 09:00 to 10:00 on June 18, 2020, in the recent component. In addition, the daily periodic component is required to input data corresponding to 10:00 to 11:00 on June 16, 2020, and 10:00 to 11:00 on June 17, 2020. The weekly periodic component is also required to input data corresponding to 10:00~11:00 on June 4, 2020, and 10:00~11:00 on June 11, 2020.

3.3.3 Multi-component fusion

In this part, this work integrates periodic data through a multi-component fusion method. In summary, to simplify the model's calculation process, enhance the connection between components, and make the model more scalable. Using the idea of data dimensionality reduction or data compression, this thesis redesigned a feature data fusion method. Specifically, it is to compress three groups of periodic data and reduce the dimensionality to one dimension. That is, three groups of periodic components are projected onto the same time coordinate axis. Then, a set of inputs can be comprehensively trained through a model. This can reduce the computational complexity and improve the integration of the model in terms of periodicity.

In the periodic analysis section, this work demonstrates the influence of the three different dimensions of the space-time components contained in the model on the

prediction results through periodic quantitative analysis. And in the periodic dependency component division, the recent dependency, daily periodic dependency and weekly periodic dependency of traffic flow data are specifically divided. According to existing research, most of the research in this field has adopted multi-component fusion technology [4, 18, 41, 43]. The idea of multi-component fusion is first to train each component separately and then fuse the output results obtained from the training to obtain the final prediction result Y, as shown in formula (3-9):

$$Y = W_H \odot Y_H + W_D \odot Y_D + W_W \odot Y_W \quad (3\text{-}9)$$

In the formula: $\odot$ means Hadamard product. Y_H, Y_D, Y_W correspond to the output of adjacent components, daily periodic components and weekly periodic components, respectively. W_H represents the weight matrix of adjacent components, W_D represents the weight matrix of daily periodic components, and W_W represents the weight matrix of weekly components.

It can be seen from the above formula (3-9) that the traffic flow prediction model based on multi-component fusion learns the degree of influence of different components on different nodes by learning from historical data. It can maximize the role of multiple components at different nodes. However, it can also be seen that the model needs to implement three relatively independent prediction models of short-term, daily-period, and weekly-period components. This invisibly increases the volume of the model, which means an increase in the number of overall parameters of the model and an increase in computational complexity. As a result, it will also cause a decrease in the degree of parallelization of the model during the training process and a decrease in the overall computing speed. In addition, the three periodic component models only fuse feature data together in the last layer. Such an approach actually weakens the mutual influence of the three components during the training process.

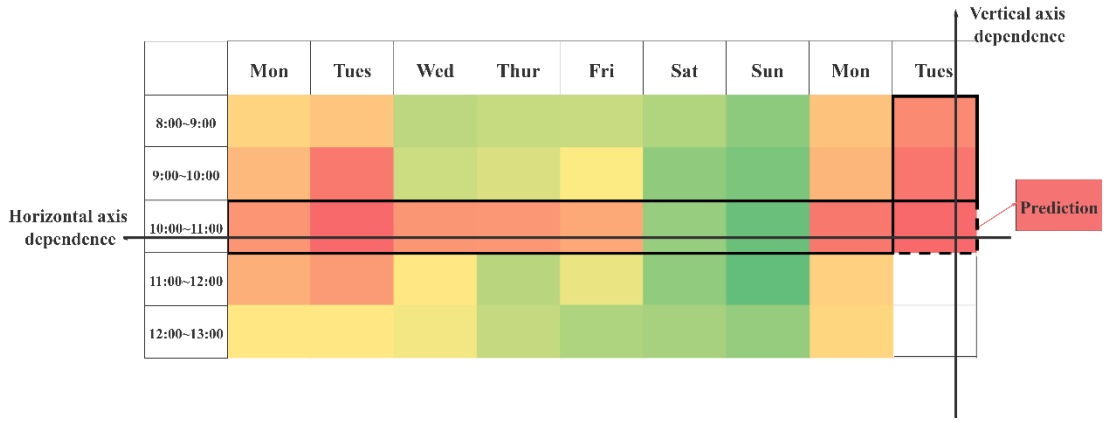


Figure 3.6 Schematic diagram of feature data dependency

As shown in Figure 3.6. According to the quantitative analysis of the Pearson correlation coefficient, it can be seen intuitively that the influence of the three components on the value to be predicted can be divided into two parts: the vertical axis and the horizontal axis. Therefore, all periodic dependent components can be projected on a new coordinate axis in terms of data mapping, as shown in Figure 3.7.

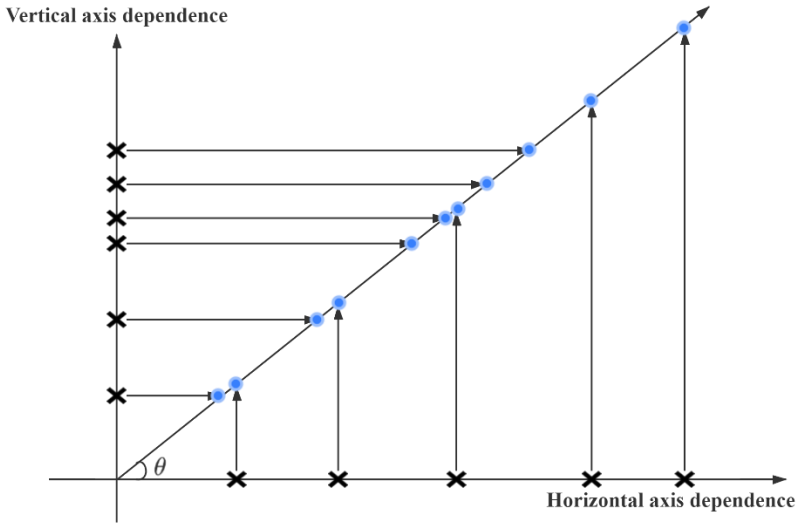


Figure 3.7 Feature data fusion

As a result, the model can merge the three components into one time dimension in the early stage of feature data input. So as to avoid the shortcomings of fusion in the later stage mentioned above. On this basis, the specific formula of the fusion method we proposed is as follows:

$$X = CONCAT(\frac{X_h}{cos\theta}, \frac{X_w}{sin\theta}, \frac{X_d}{sin\theta}) \quad (3\text{-}10)$$

Among them, $CONCAT(\cdot)$ means to concatenate all three sets of feature data in the same dimension. $X \in \mathbb{R}^{N \times C \times L}$. When N and C remain unchanged, the length L of the feature data input increases, and its length is the sum of the three time component dimensions. Therefore, the model simplifies the calculation process through this method, strengthens the connection between components, and makes it more scalable.

3.4 MST-GCRN and MST-GCTN Models

The second module of the model framework comprises two parts: MST-GCRN (Multiple layers Spatial-temporal aware Graph Convolutional Recurrent Network model) and MST-GCTN (Multiple layers Spatial-temporal aware Graph Convolutional and Temporal Convolutional Network model). Both neural network modules have the ability to extract the temporal and spatial characteristics of traffic flow data. Moreover, these two neural network models work independently. According to the current research status of traffic flow prediction in the literature review part and the discussion in the relevant sections of graph neural network, this research divides the modeling into three parts.

The first part of the two neural network models is GCN modules. The GCN module designed in our simulation can be called a spatial graph convolution module. This module is used to extract the spatial characteristics of the traffic flow. Using the idea of graph convolutional neural network, the spatial characteristics of the road network traffic flow at each node are used for message passing. So as to realize the feature aggregation and update in the spatial structure. This thesis demonstrates that GCN is superior in its ability to aggregate structural feature information, which can be seen from the general use of GCN to aggregate spatial structure features in many types of research on deep learning-based traffic flow prediction problems.

In the second part, there are recurrent neural network (GRU) module and temporal convolution neural network (TCN) module, respectively. In this part, we designed

these two modules to extract the temporal features of traffic flow data. These two methods respectively perform convolution/recurrent neural network operations on the features of each node of the road network traffic flow in the temporal dimension. Therefore, the features of the entire model in the temporal dimension are updated.

The third part is the prediction part of traffic flow. Through T iterations of the model in the temporal and spatial dimension, the characteristics of the flow data are fully learned. Therefore, the models can obtain sufficient and reliable high-dimensional features information on the temporal and spatial structure of road network traffic. Then, input this high-dimensional information into a standard neural network with a fully connected layer. Next, the machine learning training and model parameter adjustment is carried out through the neural network's forward and backward transmission process. Finally, through multi-epoch training, the models can obtain a prediction of traffic flow at a certain node and a certain time segment.

3.4.1 Spatial convolution model for traffic flow prediction

This work takes the extraction of the spatial structure of the road network within a certain period of time as the first step. In general, this thesis comprehensively weighs the pros and cons of multiple methods. GCN was chosen as the spatial convolution model for traffic flow prediction. Through the superposition of multiple space-time modules, the depth of the GCN layer is deepened. This can provide richer and more powerful features data for the prediction models. The accuracy and reliability of the prediction models are further improved.

As mentioned in the previous chapter, convolutional neural networks have been successfully applied to extract meaningful patterns and features from large-scale, high-dimensional data sets. The traffic flow data containing hidden local features are well suited for retrieving the correlation between its location and neighbors through CNN. However, due to the inability to process data with a non-European structure, standard CNN cannot solve complex road network problems. Therefore, GCN based on graph structure has become a more feasible alternative method, effectively

extracting spatial features from such data structure (topological graph) for machine learning. Specifically, graph convolution can effectively extract spatial information on sparse graphs, and only a few trainable parameters are needed. Moreover, the operation of graph convolution can be seen as applying a strictly localized filter to traverse the graph. Therefore, the information between adjacent nodes can be grouped and distributed through graph convolution.

3.4.1.1 Spatial GCN modeling for traffic flow prediction

In this part, this thesis will prove why GCN can well capture the spatial features of traffic flow data. In other words, this work aggregates the spatial features of adjacent nodes ($k = 1$) and higher-order neighbor nodes ($k = 2, \ldots, n$) for the target node through the message passing of multi-layer spatial modules. Figure 3.8 shows a simplified diagram of the adjacency relationship between nodes in the road network structure.

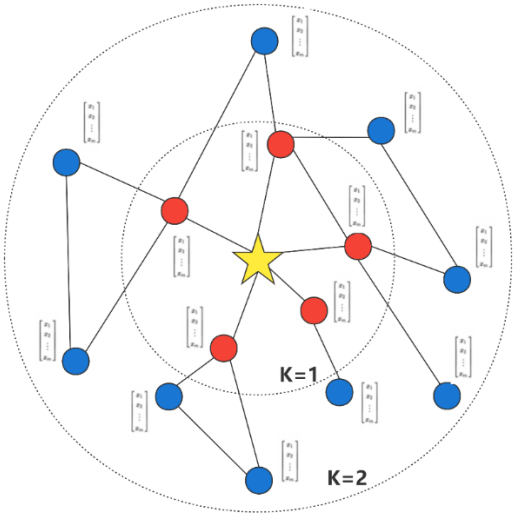


Figure 3.8 Message passing approach of GCN

As shown in Figure 3.8, in the process of a single message passing, GCN only transfers and aggregates the characteristic values of the surrounding red nodes and their own characteristic values. Therefore, it is considered to have the problem of insufficient scalability. However, through the multiple message transmission process of the multi-layer spatial GCN module, the star node can aggregate the influence of surrounding nodes and higher-order neighbor nodes. At the same time, it also includes

the adjacency relationships that may exist between neighboring nodes. It should be noted that GCN uses a full-graph training method, which means that the nodes of the full-graph must be updated in each iteration. Therefore, the complexity of GCN is $O(n^3)$. When the scale of the graph is large, this training method is undoubtedly time-consuming or even impossible to update. However, according to the data set and node scale in the field of traffic flow prediction, the size of n is moderate and acceptable. Taking the PeMSD data set as an example, there are about 100-300 data nodes after processing. Therefore, the complexity of using GCN in this research field is quite low compared to the tasks of social networks or knowledge networks. Therefore, such characteristic data is sufficient and feasible for us to predict future traffic flow.

3.4.1.2 Mathematical modeling of spatial GCN module

We use the graph convolution operation based on the spectrogram theory to process the traffic flow data directly and use the data correlation on the traffic network to extract the high-order features of the nodes in the spatial dimension.

First, only consider the spatial graph G on a certain time slice, and use this as an entry point to understanding the process of modeling spatial features. In the model proposed in this thesis, the spectrogram method is used to extend the convolution operation to graph structure data. Treat the data as signals on the graph, and then process the graph signals directly on the graph to capture meaningful data patterns and features in the space.

In spectrogram convolution, the feature of each node is regarded as a signal on the graph, expressed as $x \in \mathbb{R}^N$. A represents the adjacency matrix of the graph, and the corresponding Laplacian matrix of the graph is $L = D - A$. $D \in \mathbb{R}^{N \times N}$ is the diagonal matrix of the graph, $D_{ii} = \sum_j A_{ij}$. The standardized format of L is $L = I_N - D^{-\frac{1}{2}} A D^{\frac{1}{2}}$. $L \in \mathbb{R}^{N \times N}$, I_N is the identity matrix. The eigenvalue decomposition of L has $L = U \Lambda U^T$, where $\Lambda \in \mathbb{R}^{N \times N}$ is a diagonal matrix composed of the eigenvalues of L. U is the Fourier basis, the eigenvector matrix of L. Use $*_G$ to denote the convolution

operation on the graph, and then use the convolution kernel Θ to convolve the signal on the graph, which can be expressed as shown in formula (3-11). Its meaning is to first map the image signal x to the spectral domain through the Fourier transform of the image. Then use the convolution kernel $\Theta(\Lambda)$ to convolve the image signal x in the spectral domain and then perform the inverse Fourier transform. Finally, the result of graph convolution is obtained. Since formula (3-11) exists in the Fourier basis product operation of the graph, the computational complexity of formula (3-11) is $O(n^2)$ [99]. n is the number of nodes. To reduce the computational cost of the graph convolution operation, the Chebyshev polynomial or the first-order approximation can be used to approximate the equation (3-11).

$$\Theta *_{\mathcal{G}} x = \Theta(L)x = \Theta(U\Lambda U^T)x = U\Theta(\Lambda)U^T x \quad (3\text{-}11)$$

(1) Chebyshev graph convolution

To allow each filtering operation to be performed in the local space of the graph node and to reduce the filter parameters as much as possible, a polynomial filter can be used, and the convolution operation is shown in formula (3-12):

$$\Theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k \quad (3\text{-}12)$$

Where $\theta \in \mathbb{R}^k$ is a vector of polynomial coefficients. K is the size of the graph convolution kernel, which represents the maximum radius of the center node of the convolution operation. It can be approximated by Chebyshev polynomial to $\Theta(\Lambda)$, as shown in formulas (3-13) and (3-14):

$$\Theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{\Lambda}) \quad (3\text{-}13)$$

$$\widetilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - I_n \quad (3\text{-}14)$$

Where λ_{max} is the maximum eigenvalue of the Laplacian matrix L. Therefore, the graph convolution in formula (3-11) can be expressed by formulas (3-15) and (3-16):

$$\Theta *_{\mathcal{G}} x = \Theta(L)x \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x \quad (3\text{-}15)$$

$$\tilde{L} = \frac{2L}{\lambda_{max}} - I_n \quad (3\text{-}16)$$

The recursive definition of Chebyshev polynomial is $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, $T_0(x) = 1, T_1(x) = x$. An approximate solution based on Chebyshev polynomial can reduce the computational complexity of equation (3-11) to $O(K\varepsilon)$, where ε represents the number of edges in the graph.

The above-defined Chebyshev graph convolution operation defined in the single-dimensional graph signal data $x \in \mathbb{R}^N$ can be extended to multi-dimensional data. For two-dimensional data $x \in \mathbb{R}^{N \times C_i}$, C_i represents the feature dimension of the node. Let $y \in \mathbb{R}^{N \times C_o}$ denote the output of X after graph convolution. C_o represents the feature dimension of each node after graph convolution. Then the Chebyshev graph convolution operation on X is shown in formula (3-17), where $\Theta \in \mathbb{R}^{K \times C_i \times C_o}$ is the convolution kernel parameter to be learned.

$$Y = \Theta *_{\mathcal{G}} X = \Theta(L)X = \sum_{k=0}^{K-1} T_k(\tilde{L}) X_k \Theta_k \quad (3\text{-}17)$$

(2) First-order approximate graph convolution

The neural network model based on graph convolution can be realized by stacking multiple graph convolution layers as shown in equation (3-15), and a nonlinear layer needs to be connected after each layer of graph convolution. The K in formula (3-15) is set to 2, which can indicate the convolution of the graph with the first-order neighbors in the convolution range. A larger local area graph convolutional neural network can be realized by stacking multiple layers of first-order neighbor graph convolution plus a nonlinear layer [91]. When K takes 2, formula (3-15) is as shown in formula (2-18). Since the Chebyshev graph convolution network uses a normalized Laplacian matrix, it can be assumed that the normalized Laplacian matrix has $\lambda_{max} \approx 2$, then the formula (3-18) can be written as the formula (3-19).

$$\Theta *_{\mathcal{G}} x \approx \theta_0 x + \theta_1 (\frac{2L}{\lambda_{max}} - I_n) x \quad (3\text{-}18)$$

$$\Theta *_{\mathcal{G}} x = \theta_0 x - \theta_1 (D^{-\frac{1}{2}} A D^{\frac{1}{2}}) x \quad (3\text{-}19)$$

Among them, θ_0 and θ_1 are filter parameters, which are shared across the entire graph. In order to limit the number of parameters of the model and avoid overfitting. It can be assumed that $\theta = \theta_0 = -\theta_1$, so the first-order approximate graph convolution can be expressed by the formula (3-20):

$$\Theta *_{\mathcal{G}} x = \theta \left(I_n + D^{-\frac{1}{2}} A D^{\frac{1}{2}} \right) x = \theta (\widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{\frac{1}{2}}) x \quad (3\text{-}20)$$

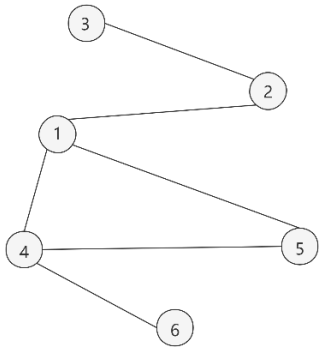
Where $\tilde{A} = A + I_n, \widetilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.

The first-order approximate graph convolution operation defined above on the one-dimensional graph signal data $x \in \mathbb{R}^N$ can be extended to the multi-dimensional graph signal data. For the two-dimensional graph signal data $x \in \mathbb{R}^{N \times C_i}$, C_i represents the characteristic dimension of the node. Let $Z \in \mathbb{R}^{N \times C_o}$ represent the output of X after the graph convolution operation, and the first-order approximate graph convolution operation of X is shown in formula (3-21). Where $\Theta \in \mathbb{R}^{C_i \times C_o}$ represents the parameter matrix of the convolution kernel.

$$Z = \widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{\frac{1}{2}} X \Theta \quad (3\text{-}21)$$

Figure 3.9 can be used to briefly show the feature extraction process of the 0-1 order neighbor information of the spatial node in the graph convolution process. Figure 3.9(a) is a simple topology diagram of the spatial road network structure. Figure 3.9(b) is the adjacency matrix and degree matrix calculated from the road network in Figure 3.9(a). The matrix representation of $\widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{\frac{1}{2}}$ and input data X is shown in Figure 3.9(c). For a more intuitive representation, assume that the current input data X has only one-dimensional features and the length of the time dimension is 1. From this, the result of a graph convolution operation performed by the convolution kernel on the data X is calculated, as shown in Figure 3.9(d). Among them, the value of space node 1 at time t is updated by itself and the information of the three nodes 2, 4, and 5. It changed from the original x_t^1 to $\Theta_1(0.25x_t^1 + 0.29x_t^2 + 0.25x_t^4 + 0.29x_t^5)$. That is, the input data is updated by the information of its 0-1 order neighbors. In the same

way, a graph convolution operation is performed on the entire input data X to obtain $\Theta *_{\mathcal{G}} X$, and the value of each node is updated by the information of the node's 0- $K-1$ order neighbors. When expanding to multi-dimensional data, the operation remains the same.



3.9(a)

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

3.9(b)

$$\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2} = \begin{pmatrix} 0.25 & 0.29 & 0 & 0.25 & 0.29 & 0 \\ 0.29 & 0.29 & 0.41 & 0 & 0 & 0 \\ 0 & 0.41 & 0.50 & 0 & 0 & 0 \\ 0.25 & 0 & 0 & 0.25 & 0.29 & 0.35 \\ 0.29 & 0 & 0 & 0.29 & 0.33 & 0 \\ 0 & 0 & 0 & 0.35 & 0 & 0.50 \end{pmatrix} \in \mathbb{R}^{6\times 6} \qquad X = \begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \\ \dots \\ x_t^6 \end{pmatrix} \in \mathbb{R}^{6\times 1\times 1}$$

3.9(c)

$$\Theta_1 *_{\mathcal{G}} X = \begin{pmatrix} \Theta_1(0.25x_t^1 + 0.29x_t^2 + 0.25x_t^4 + 0.29x_t^5) \\ \Theta_1(0.29x_t^1 + 0.29x_t^2 + 0.41x_t^3) \\ \Theta_1(0.41x_t^2 + 0.50x_t^3) \\ \Theta_1(0.25x_t^1 + 0.25x_t^4 + 0.29x_t^5 + 0.35x_t^6) \\ \Theta_1(0.29x_t^1 + 0.29x_t^4 + 0.33x_t^5) \\ \Theta_1(0.35x_t^4 + 0.50x_t^6) \end{pmatrix}$$

3.9(d)

Figure 3.9 GCN matrix representation

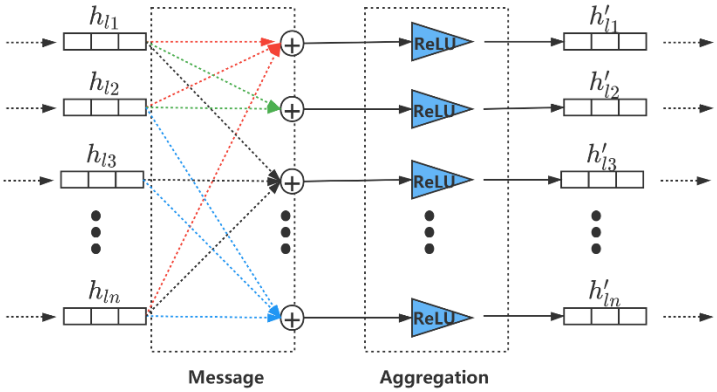


Figure 3.10 Spatially dependent feature aggregation at time t

As shown in Figure 3.10, h_l represents the feature input of the l-th neural network module at time t. h_{l1} represents the feature input of node 1 at time t. Through the message sending process, each node on the graph structure obtains the feature value set of the neighbor node (for example, the neighbor node of node 1 is h_{l2}, h_{ln}) and the current state of its own node (h_{l1}). Through the aggregation operation, the node is updated to the high-dimensional node feature (h'_{l1}) in the spatial dimension and used as the input for the next round of spatiotemporal feature extraction. Through the above method, the value of a certain sensor node A in the road network space at time t is updated by the information of itself and its n-th order neighbors. Extending to the whole world, this method also performs graph convolution operations on all nodes of the input traffic flow data in batches. Thus, the feature data information update based on the entire graph structure is obtained.

3.4.2 GRU based temporal feature extraction module

Through the GCN module, spatial feature dependence is effectively extracted. On this basis, our spatial-temporal convolution module uses GRU or TCN to extract the temporal feature dependence of traffic flow data. In this part, this work uses a simple and powerful variant of the recurrent neural network gated recursive unit GRU to capture the temporal dependence of the road network traffic flow. As mentioned earlier, the RNN model can effectively capture the temporal dependence of the road network traffic flow in the traffic flow prediction. The special structure inside the

RNN can store and memorize the context information of the sequence and use the stored information in future operations. This shows that RNN has a strong time series learning ability.

In the machine learning research of time series data, the Recurrent Neural Network is most commonly used to model the time dependence relationship, so as to extract the temporal dependence relationship from the series data. Some articles [121-123] believe that for the time series of sequence data, its distinguishing feature is that the context of the sequence is highly relevant. The special structure inside the RNN can store and memorize the contextual information of the sequence and use the stored information in future operations. For RNN, the output of its hidden layer not only enters the output end but also enters the next hidden layer. Thereby it can have an impact on the weight on the next time step. The internal memory unit of RNN can be used to process any sequence of input data， so that RNN has the ability of time sequence learning. Moreover, to solve the problem of gradient explosion and disappearance when the input sequence is relatively long, these problems can be solved by using an improved recurrent neural network LSTM or GRU.

3.4.2.1 Mathematical modeling of GRU module

This thesis uses GRU for temporal dependence modeling. Use x^t to represent the input signal of the current node. Use h^{t-1} to represent the hidden state passed down from the previous node. Then the GRU uses x^t and h^{t-1} to obtain two gate control states, and the formulas for the reset gate and update gate are as follows:

$$r = \sigma(W^r) \cdot [h^{t-1}, x^t] \quad (3\text{-}22)$$

$$z = \sigma(W^z) \cdot [h^{t-1}, x^t] \quad (3\text{-}23)$$

Among them, W is the weight matrix that needs to be trained in the model. After getting the gating signal, first, use the reset gate to get the reset data $h^{(t-1)'} = r^t \odot h^{t-1}$, and then $h^{(t-1)'}$ and x^t are spliced together. Then use a $tanh$ activation function to shrink the data to the range of $(-1,1)$, and get h':

$$h' = \tanh\left(W^{h'}[h^{t-1} \odot r, x^t]\right) \quad (3\text{-}24)$$

$\odot$ stands for multiplication of matrix elements. h' mainly contains the currently input x^t data. A targeted pair h' is added to the current hidden state, which is equivalent to "memorizing the state at the current moment". The final update memory stage uses both forgetting and memory steps, using the previously obtained update gate z, and the update expression is as follows:

$$h^t = z \odot h^{t-1} + (1 - z) \odot h' \quad (3\text{-}25)$$

r and z denote the reset gate and update gate of the GRU, respectively. h' represents the output of the network at time t. The input of each layer of GRU considers the output of the previous layer of GRU, thereby capturing the timing relationship of the road network traffic flow.

3.4.3 TCN based temporal feature extraction module

In this work, in the temporal-dimensional modeling process, not only from the perspective of RNN but also from the perspective of CNN to perform temporal-dependent modeling. From the comparison between RNN and CNN, we can more clearly confirm their respective advantages and disadvantages, and which model is more suitable for traffic flow prediction tasks. By discussing the advantages and disadvantages of the GCN+CNN model and the GCN+RNN model in the field of traffic flow prediction, a more accurate method of predicting traffic flow can be reflected. To model the temporal dimension from the perspective of CNN, we mainly have the following considerations.

First of all, this can strengthen the degree of integration of the spatial-temporal neural network layer in the spatial-temporal module. Secondly, for a relatively complex network with many graph nodes, the relatively complex single-layer neural network module used by RNN will cause problems such as large time overhead and slow response to dynamic changes of data. Compared with GRU, CNN has the advantage of fewer non-linear operations. It can effectively reduce the phenomenon of gradient

dispersion, making model convergence and training easier. In addition, in GRU, the model's output at the next moment depends on the state of the hidden layer at the previous moment. Therefore, the model cannot be parallelized. However, CNN does not need this kind of dependence and can be easily parallelized, thereby achieving an increase in computing speed.

3.4.3.1 Temporal CNN modeling for traffic flow prediction

According to the discussion of RNN and GNN in the previous chapter, and compared with the prediction model based on the recurrent structure, CNN has better feature value extraction capabilities and can achieve better prediction results. Therefore, in recent years, many scholars have used the CNN architecture in time series processing tasks, and the same is true in the field of traffic flow prediction. Such as ST-ResNet [37], MSTGCN [18] and so on. The CNN architecture shows strong learning ability in the field of image processing and has a better prediction accuracy than the recurrent structure on timing tasks in some scenarios. However, it is not supported by a complete basic theory like the recurrent structure. Therefore, the traditional CNN architecture does not have good interpretability in time series data processing.

Figure 3.11 shows the difference between the recurrent structure and the convolution structure in processing time series data. The recurrent structure adds connection operations in chronological order through neurons in the hidden layer. It transfers historical feature information to the next neuron in the form of a hidden layer state so that the network maintains the ability of historical memory. The convolutional structure does not join the connection operation in the hidden layer, so it can only extract features in a local range and treat any neuron indiscriminately. Therefore, traditional convolution will converge the characteristic information of the "future" time step to the current time step, so it is not interpretable.

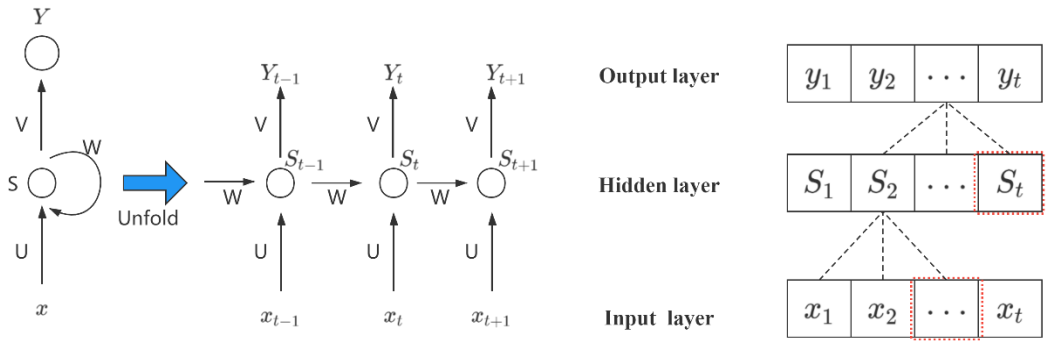


Figure 3.11 RNN structure and CNN structure

3.4.3.2 TCN based temporal dependence modeling

The traffic flow has a certain trend and correlation in the adjacent time interval. Therefore, after the space-dimensional convolution module, this work uses the temporal convolutional network (TCN) to perform convolution operations on the features of different time intervals of the station along the direction of the temporal dimension. This method can capture the temporal feature of the site. It can be seen from section 2.3.3 that TCN not only has the learning ability of CNN architecture but also has better interpretability than traditional convolution. From the perspective of the time dimension, causal convolution can solve the problem of information leakage very well.

For sequence tasks such as traffic flow prediction, it is necessary to model the traffic flow in the previous period of time instead of relying solely on the traffic flow at the previous moment. According to section 2.2 Convolutional Neural Networks, convolutional neural networks can form "memory" through convolution calculations. However, the biggest problem of using traditional convolutional neural networks in sequence prediction tasks is how to obtain the long-term memory of the sequence and how to deal with the problem of information leakage. Information leakage refers to sequence processing problems such as traffic flow forecasting. It needs to ensure that the model cannot reverse the sequence order. When the model predicts time t, future time data such as $t + 1$, $t + 2$ cannot be used. Therefore, this thesis uses the temporal convolutional network (TCN) model [124] to solve these problems. It is a

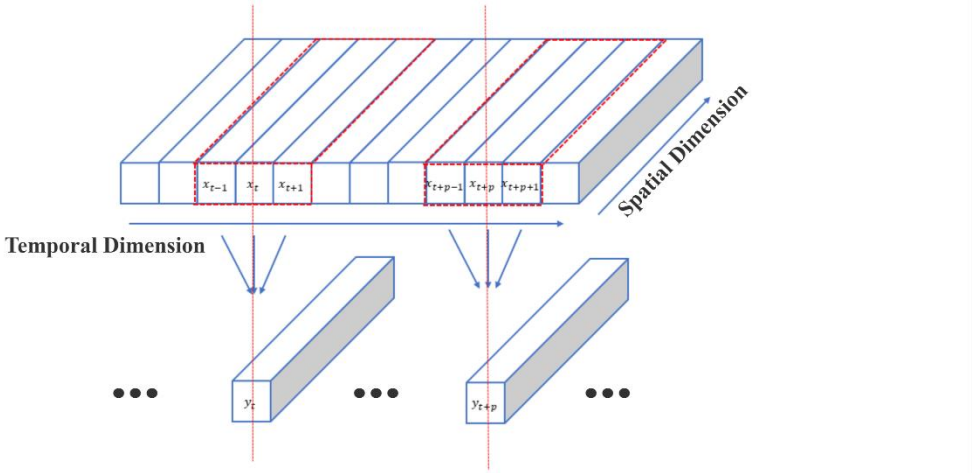
CNN architecture that uses causal convolution and dilated convolution instead of traditional convolution. The calculation formula of causal convolution is as follows:

$$F(s) = (x * f)(s) = \sum_{i=0}^{K-1} f(i) x_{s-i} \quad （3\text{-}26）$$

Among them, K is the size of the causal convolution kernel. $f = \{f_1, f_2, \dots, f_i\}$ is the convolution kernel. x is the feature vector of $s(t,n)$ (n node at time t). $F(s)$ represents the causal convolution at s. However, in practical applications, the convolution kernel of the causal convolution is generally set to a fixed value. It causes the limited range of the receptive field of causal convolution, and the limited memory capacity makes it unable to preserve long-term historical memory. Therefore, TCN uses dilated convolutions to expand the range of receptive fields. Dilated convolution extends the memory capacity of the network by expanding the convolution window. The calculation formula is as follows:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{K-1} f(i) x_{s-d\times i} \quad (3\text{-}27)$$

Among them, d is the dilation factor, that is, the size of the evaluation convolution window. When d is 1, the dilated convolution degenerates into ordinary convolution. By controlling the size of d, the receptive field is widened, and the memory capacity of the model is prolonged under the premise of the same amount of calculation.



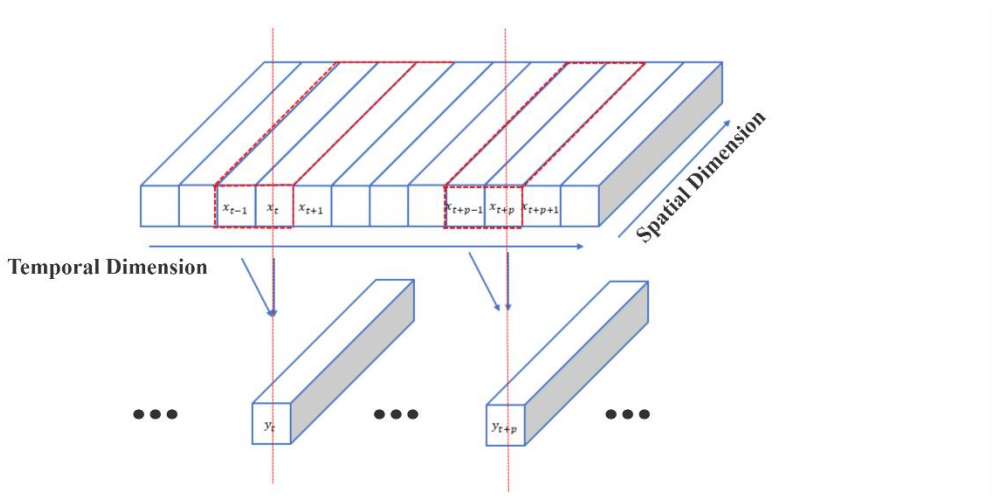


Figure 3.12 The difference between CNN (up) and TCN (down)

Figure 3.12 shows the structure and difference between traditional convolution and TCN. It can be seen that, compared to traditional convolution, TCN will not merge the feature information of the "future" time step into the current time step. That is, for the value at time t of the previous layer, it only depends on the value at time t and before the next layer. The difference from traditional CNN is that TCN cannot "see" future data. It has a one-way structure, not a two-way structure. That is to say, only the first cause can have the latter result, which is a strict time constraint model. Therefore, the TCN model has better interpretability and long-term memory ability than the CNN model.

The advantages of TCN are:

1. Parallelism. TCN can process feature data in parallel, without the need for sequential processing like RNN.

2. Flexible receptive field. The receptive field of TCN is determined by multiple factors (such as convolution kernel and dilation factor), and it can be flexibly customized according to different requirements and characteristics.

3. Stable gradient. RNN often has the problem of gradient disappearance and gradient explosion, which is mainly caused by the sharing of parameters in different time periods. Like traditional CNN, TCN has almost no gradient disappearance and

explosion problems.

4. Lower memory requirements. The convolution kernel of TCN is shared in one layer, and the memory usage is lower, which is quite different from RNN.

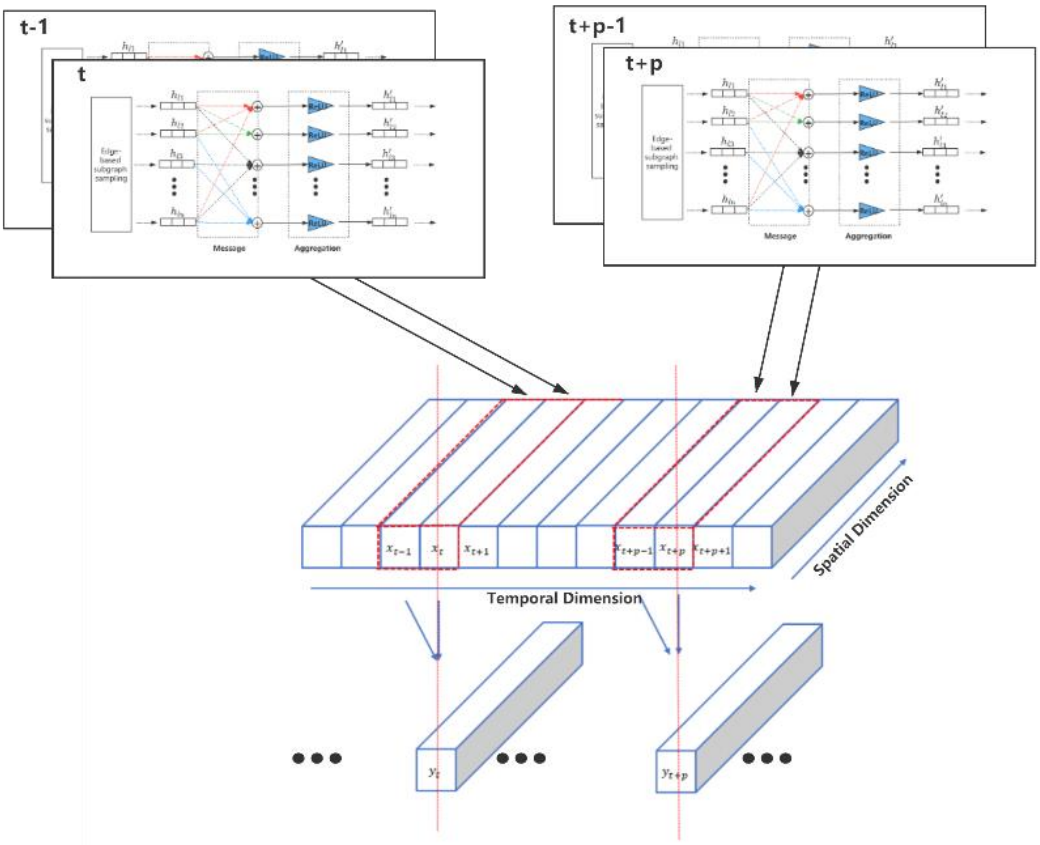


Figure 3.13 Temporal dependent feature aggregation

As shown in Figure 3.13, the GCN+TCN model obtains higher-dimensional feature values through the message passing process of neighbor nodes. The model updates the node in the time dimension to higher-dimensional node features through the aggregation operation and uses it as the input for the next spatial and temporal feature extraction round.

In summary, the input of the temporal-dimensional GRU/TCN module is the output of the spatial feature learning unit. Through the superposition of the spatial-temporal modules, the traffic flow features of the nodes already contain rich spatial feature information. Therefore, using the GRU/TCN module to superimpose the spatial module can effectively capture the temporal and spatial features of the traffic flow of

the stations.

3.4.4 Readout phase

For this thesis, the task of the Readout phase is to input the high-level traffic flow spatiotemporal feature information obtained through the learning of multiple spatiotemporal graph neural networks into the standard supervised machine learning neural network training task. Find the hidden relationship between the predicted value and the label to achieve the purpose of learning and prediction by adjusting various parameters in the neural network. This process is also the process of building a standard neural network to solve linear regression problems. In this process, two main tasks need to be solved. One is to determine the depth of the neural network, and the other is how to solve the problem of neural network overfitting.

For the first question, we increase the depth and feature richness of neural networks by stacking spatial-temporal modules. Therefore, the model has obtained sufficiently high-order and powerful features before the Readout phase. The model does not need too many hidden layers to learn from the data in the Readout phase. Therefore, only a neural network containing two hidden layers is required to complete the machine learning task at this stage. This setting can be verified in the next simulation stage. In the field of machine learning, the formula part of the multivariate linear regression model has been very mature, and we will not elaborate too much here.

For the second problem, over-fitting is a very critical problem that affects neural network learning. The way to solve the over-fitting problem is to adopt a regularization method and add a dropout layer in the process of machine learning. The learning effect of the combined use of these two methods is ideal. Therefore, in this work, first, use the $L1$ or $L2$ regularization method to assist in solving the problem of overfitting. For the neural network structure, by inserting a two-layer dropout layer in the neural network to prevent over-fitting.

3.4.5 Proposed MST-GCRN and MST-GCTN models

A fully functional traffic flow prediction model was built through the feature analysis and periodic data fusion module, spatial-temporal convolution module, and readout module. The overall model is shown in Figure 3.14.

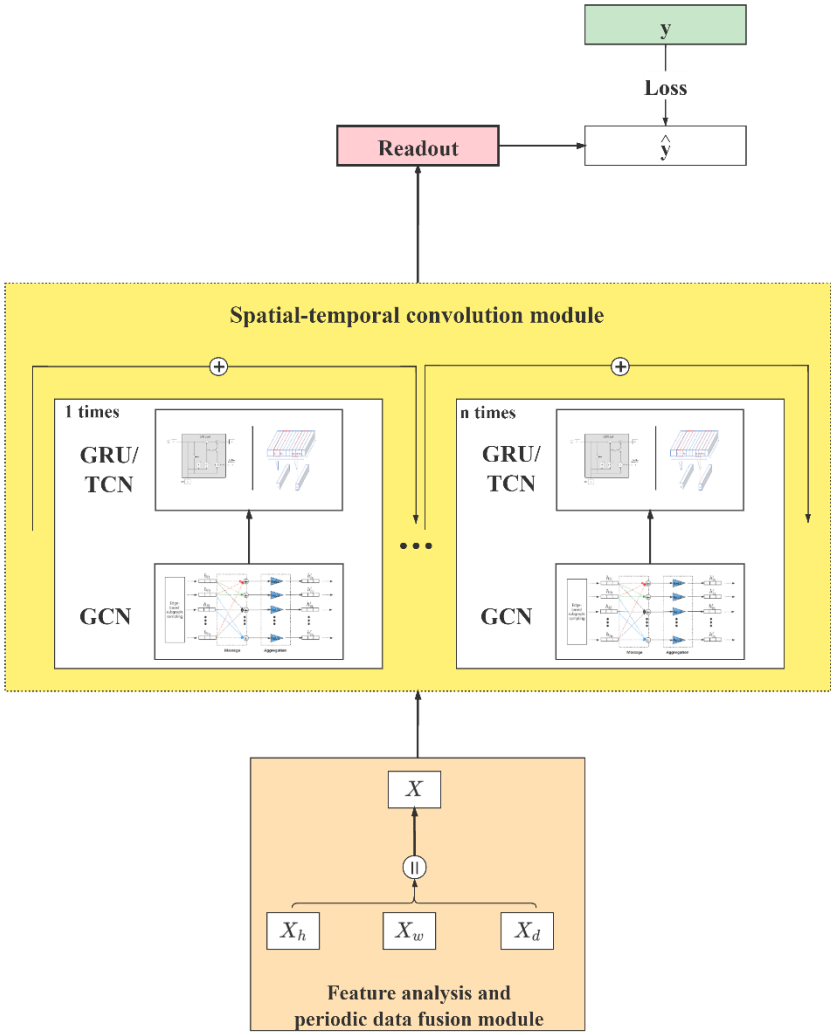


Figure 3.14 Spatial-temporal graph neural networks model

In the feature analysis and periodic data fusion module, the periodic spatial-temporal data is concatenated to the same temporal dimension according to the quantitative analysis method of the data. The processed traffic flow data first passes through the GCN-based spatial convolution module to extract spatial dependent features. After the spatial module, high-order features are input to the temporal dependence module. Then, the temporal dependence is extracted separately through the GRU/TCN module.

By adding a simple but effective residual network between spatial-temporal modules, the problem of network degradation caused by the increase of network depth is solved. Finally, the updated traffic flow feature is used as the input of the next spatial-temporal module. Feature extraction and update through several times of spatial-temporal modules, high-level feature data is input into a neural network with a fully connected layer to read out the prediction results and carry out neural network training tasks.

3.5 Attention Mechanism based MST-AGCRN and MST-AGCTN Models

In this section, we propose the MST-GCRN (MST-AGCRN) and MST-GCTN (MST-AGCTN) models based on the self-attention mechanism to improve the scalability, flexibility and accuracy of the prediction. Specifically, based on the self-attention mechanism, this thesis designs the attention model as a two-dimensional attention mechanism module about time and space (spatial attention and temporal attention). This module can learn the mutual influence weights between nodes for different spatiotemporal scenarios and adjust the input data. It can be seen from the formula that when the GCN-based model captures the spatial-temporal relationship between nodes, the weight of the model's edges is unchanged. Through the self-attention mechanism, our improved new model can dynamically adjust the weights imposed by neighbor nodes on the target node. Through this mechanism, different neighbor nodes have different influences on the target node. It makes the node's attention distribution more reasonable and can effectively capture the dynamic spatial-temporal correlation on the transportation network. Furthermore, dynamically capture the high-dimensional feature data of each node in different time and space dimensions, which can better reflect the actual characteristics of the traffic road network.

3.5.1 Spatial-temporal attention mechanism module

The spatial-temporal-based attention mechanism proposed in this work is to introduce

the attention mechanism based on the spatial-temporal graph neural network. The module contains temporal attention modules and spatial attention modules. The two modules analyze the spatial attention and temporal attention of the traffic flow, respectively, and set the weight of the edge as a learnable function between nodes to capture the dynamic temporal and spatial characteristics. By paying attention to its neighbors and following a self-attention strategy, the spatial and temporal features of the road network can be better extracted. So as to achieve a better effect than the spatiotemporal graph neural network. On this basis, the module based on the spatiotemporal attention mechanism is combined with the spatiotemporal convolutional network proposed in the previous chapter. A graph spatiotemporal convolutional network based on the attention mechanism is formed.

From the description in the previous section, it can be seen that although the GCN model based on spatial features and the GRU/TCN model based on time features can effectively extract the temporal and spatial dependence of traffic flow. However, the model does not take into account the dynamic correlation characteristics of space-time dimensions.

In the model in the previous section, the weight of the edge between nodes is only related to the degree of the node. Although the model can reflect the road network structure, because the degree matrix of the fixed network structure is fixed, the weight of the edge is fixed and unlearnable. However, in the actual situation, the weight of the "edge" should not be fixed and unlearnable. In the spatial dimension, the traffic conditions of different locations will influence each other, and this influence is highly dynamic. In the temporal dimension, there is a correlation between the traffic conditions in the same place at different time periods. This correlation also changes with the change of space and time. Therefore, considering the temporal and spatial dynamic correlation of traffic flow in the model is beneficial to improve the model's prediction accuracy.

3.5.2 Spatial attention mechanism module

In the spatial dimension, the traffic conditions of different nodes will influence each other, and this influence is highly dynamic. This work enables the spatial attention module to adaptively capture the correlation between sensors in the road network through the self-attention mechanism. The key idea is to dynamically assign different weights to different nodes at different time steps. Through SoftMax, all attention coefficients of the node at a certain time are weighted and summed to make the sum equal to 1. As shown in Figure 3.15, when calculating the spatial attention coefficient of the l-th spatiotemporal attention module of node C at t_j, the attention coefficients of node C and all nodes in the road network must be calculated separately, and the road network structure must also be considered. To this end, the hidden state and the spatial-temporal embedded module are connected together. In this way, the dynamic correlation between nodes in the spatial dimension can be adaptively captured so that the nodes in the road network can exert a reasonable influence.

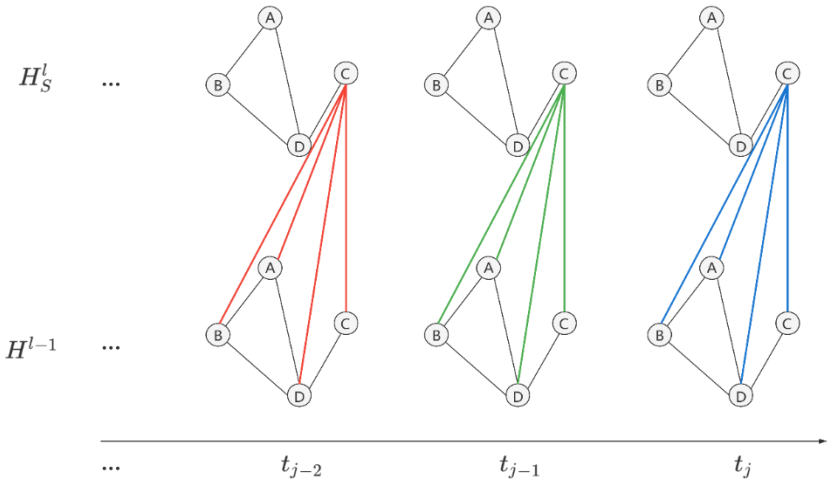


Figure 3.15 Calculation of spatial attention coefficient

At this stage, the application of graph attention mechanism in the research of the traffic flow prediction model based on the spatial-temporal dimension is still in the research and exploration stage. Among them, Guo et al.'s [18] research in this area is more representative. Their model also provided great inspiration for this research. In its ASTGCN model, the spatial attention coefficient of node v_j to node v_i can be

expressed as $\alpha_{i,j}$. According to the attention mechanism of this model, the calculation method of the spatial attention matrix is shown in formula (3-28) and formula (3-29):

$$S = V_s \cdot \sigma\left(\left(x^{(r-1)}W_1\right)W_2\left(W_3 x^{(r-1)}\right)^T + b_s\right) \quad (3\text{-}28)$$

$$S' = \alpha_{i,j} = \frac{\exp(S_{i,j})}{\sum_{j=1}^{N}\exp(S_{i,j})} \quad (3\text{-}29)$$

Among them, $x_h^{(r-1)} \in \mathbb{R}^{N\times C\times T}$, is the input of the r-th layer of the spatial-temporal module. $V_s \in \mathbb{R}^{N\times N}$, $b_s \in \mathbb{R}^{N\times N}$, where N is the number of nodes. $W_1 \in \mathbb{R}^T$, $W_2 \in \mathbb{R}^{C\times T}$, $W_3 \in \mathbb{R}^C$, T is the length of the time dimension of the input data of the r-th layer, and C is the number of channels. In addition, V_s, b_s, W_1, W_2, W_3 are all parameters that can be learned, and σ is the activation function. S is the attention matrix, and S' is the attention matrix standardized by the SoftMax function. Through the above matrix calculation, the normalized spatial attention matrix of the r-th layer finally output can be obtained as $S' \in \mathbb{R}^{N\times N}$. However, it can be seen from the formula (3-28) that when matrix calculation is performed, the information of all time dimensions on all nodes has been accumulated. From the perspective of the spatial dimension, the calculated attention coefficient is the sum of the attention coefficients between nodes in the time length T. In other words, the ideal spatial attention matrix of $S_{attention} \in \mathbb{R}^{N\times N\times T}$ is compressed into $S_{attention} \in \mathbb{R}^{N\times N}$. Therefore, the temporal attention coefficients of all nodes at time t_j to time t_i are the same. Such a method cannot fully consider the degree of influence on all nodes in different time dimensions t. It will make it difficult for the spatiotemporal attention mechanism to meet the requirements of this research (as shown in Figure 3.15) for using the attention mechanism to capture more detailed spatiotemporal dynamics in traffic flow prediction.

Therefore, this thesis proves that a more fine-grained method is to calculate the attention coefficient matrix between different nodes at different time steps t. By calculating the degree of mutual influence between different nodes simultaneously, the dynamic weight of the mutual influence of feature data in the spatial dimension is

captured. Therefore, adaptive learning using the self-attention mechanism has become the choice of this work. The self-attention mechanism is a variant of the attention mechanism, which reduces the dependence on external information and is better at capturing the internal correlation of data or features.

First, the query matrix Q, the key matrix K and the value matrix V of the spatial data input matrix of the traffic flow at time t are sequentially calculated. It can be seen intuitively from Figure 3.17 that in the process of matrix operation, if a one-dimensional convolutional network with a convolution kernel set to 1×1 is used instead of matrix multiplication as the coding function. In this way, the complexity of the calculation can be simplified, and the consistency of the input feature and the output feature in the dimension can be maintained. The matrix calculation formula is defined as follows:

$$K = k(x) = W_k x \in \mathbb{R}^{C \times N} \quad (3\text{-}30)$$

$$Q = q(x) = W_q x \in \mathbb{R}^{C \times N} \quad (3\text{-}31)$$

$$V = v(x) = W_v x \in \mathbb{R}^{C \times N} \quad (3\text{-}32)$$

Among them, $x \in \mathbb{R}^{C \times N}$ is the spatial feature vector, N is the number of spatial nodes, and C is the dimension of the data.

Secondly, normalize the weight scores obtained in the above steps, that is, use SoftMax to calculate. Let the sum of all weight factors be 1. The calculation formula is:

$$S_{attention}(K, Q) = SoftMax(QK^T) \in \mathbb{R}^{N \times N} \quad (3\text{-}33)$$

Thus, the probability representation of the degree of influence of all nodes at time t is obtained. $S \in \mathbb{R}^{N \times N}$. Let $S_{i,j}$ represent the attention matrix. In the j column, $S_{i,j}$ represents the weight of the i-th node to the j-node. The sum of the probabilities of accumulating the weights $\sum_{i=1}^{N} S_{i,j}$ is 1. By multiplying each node with the probability weight, the characteristic information of the j node can capture the dynamic

correlation between other nodes and the j-th node in the transportation network. And, it can be extended to the time length T, $S \in \mathbb{R}^{N\times N\times T}$. So as to meet the calculation requirements for the spatial attention coefficient.

At the same time, the value of V is dot multiplied, and the value is weighted and summed according to the normalized weight coefficient. The calculation formula is:

$$\hat{x}_s = VS_{attention}(K, Q) \in \mathbb{R}^{C\times N} \quad (3\text{-}34)$$

Therefore, based on the input x at time t, the output $\hat{x}_s$ with dynamic weight after being weighted by the spatial self-attention mechanism is obtained. Taking $\hat{x}_s$ as the spatial feature input of the subsequent spatial-temporal module can enable the model to better extract the spatial features of the road network. So as to achieve better results than simply using graph convolutional networks. Extending to the entire spatial-temporal input dimension, the features matrix obtained by the spatial self-attention mechanism is $\hat{X}_s \in \mathbb{R}^{N\times C\times T}$.

3.5.3 Temporal attention mechanism module

The attention mechanism of the temporal dimension is similar to that of the spatial dimension. Traffic conditions at different times affect each other, and this effect is also highly dynamic. In other words, the traffic condition of a location is related to its previous traffic condition, and the correlation changes nonlinearly with the time step. Therefore, this work uses a concept similar to the spatial attention mechanism module to construct the temporal attention mechanism module.

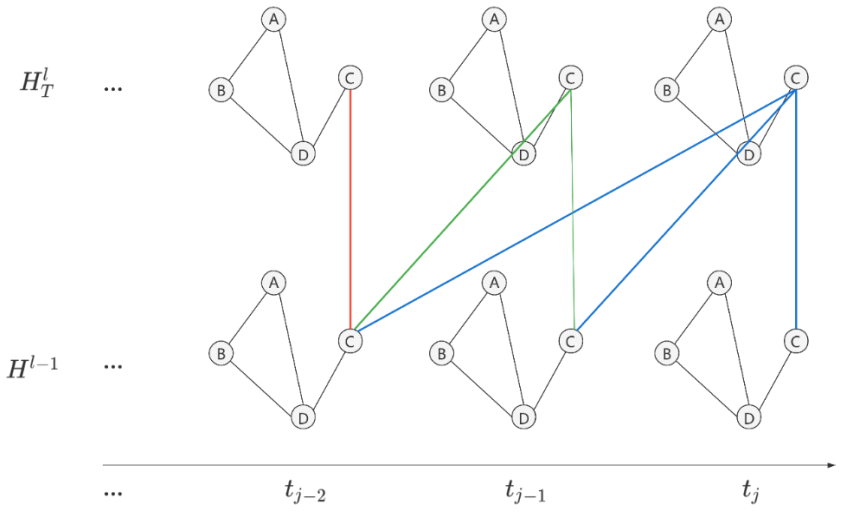


Figure 3.16 Calculation of temporal attention coefficient

As shown in Figure 3.16, at the time step t_j, the traffic condition of node C will be affected by the unexpected situation that occurred before the last or several time steps. Therefore, the non-linear correlation between traffic flow features and different time steps should be considered when calculating the attention coefficient. At the same time, the impact of the road network structure on the traffic flow must also be considered. On this basis, the temporal self-attention mechanism is used to adaptively model the nonlinear correlation between different time steps. By calculating the degree of mutual influence of the same node at different times, the dynamic weight of the mutual influence of the features data in the temporal dimension is captured.

Same as the previous section. First, the query matrix Q, the key matrix K and the value matrix V of the time data input matrix of the traffic flow at the n node are sequentially calculated. The same as the spatial self-attention mechanism, the convolution kernel is set to a 1×1 one-dimensional convolution network during the matrix operation. Use it instead of matrix multiplication as the encoding function. It is used to simplify calculation complexity and maintain the consistency of input features and output features in dimensionality. The matrix calculation formula is defined as follows:

$$K = k(x) = W_k x \in \mathbb{R}^{C \times T} \quad (3\text{-}35)$$

$$Q = q(x) = W_q x \in \mathbb{R}^{C \times T} \quad (3\text{-}36)$$

$$V = v(x) = W_v x \in \mathbb{R}^{C \times T} \quad (3\text{-}37)$$

Among them, $x \in \mathbb{R}^{C \times T}$ is the time feature vector, T is the length of the time period, and C is the data dimension.

Secondly, normalize the weight scores obtained in the above steps. That is, use SoftMax to calculate so that the sum of all weight factors is 1. The calculation formula is:

$$E_{attention}(K, Q) = SoftMax(QK^T) \in \mathbb{R}^{T \times T} \quad (3\text{-}38)$$

The probability expression of the degree of influence on a certain node at different times is obtained from this. $S \in \mathbb{R}^{N \times N}$. Let $E_{i,j}$ represent the attention matrix. In the j-th column, $E_{i,j}$ represents the weight of time i to time j. The sum of the probabilities of accumulating the weights $\sum_{i=1}^{N} E_{i,j}$ is 1. By multiplying each time with the probability weight, the feature information at time j can capture the dynamic correlation between other times and the j-th time. And, it can be extended to n nodes, $E \in \mathbb{R}^{T \times T \times N}$. In this way, the calculation requirements for the time attention coefficient can be met.

At the same time, by dot multiplying the value of V, the value is weighted and summed according to the normalized weight coefficient. The calculation formula is:

$$\hat{x}_t = E_{attention}(K, Q)V \in \mathbb{R}^{T \times C} \quad (3\text{-}39)$$

Thus, based on the input x of n nodes, the output $\hat{x}_t$ with dynamic weight after the weighting of the temporal self-attention mechanism is obtained. Taking $\hat{x}_t$ as the temporal feature input of the subsequent spatiotemporal module can make the model better extract the temporal feature of the road network. So as to achieve better results than simply using graph convolutional networks. Extending to the entire spatial-temporal input dimension, the characteristic matrix obtained by the temporal self-attention mechanism is $\hat{X}_T \in \mathbb{R}^{N \times C \times T}$.

3.5.4 Spatial-temporal self-attention mechanism module

Temporal self-attention and spatial self-attention are designed according to the above two sections. This thesis uses the self-attention mechanism to separate spatial-temporal attention effectively. This design can effectively calculate the weights of the dynamic influence of attention on different locations and moments so that the model can learn more effectively from the input data.

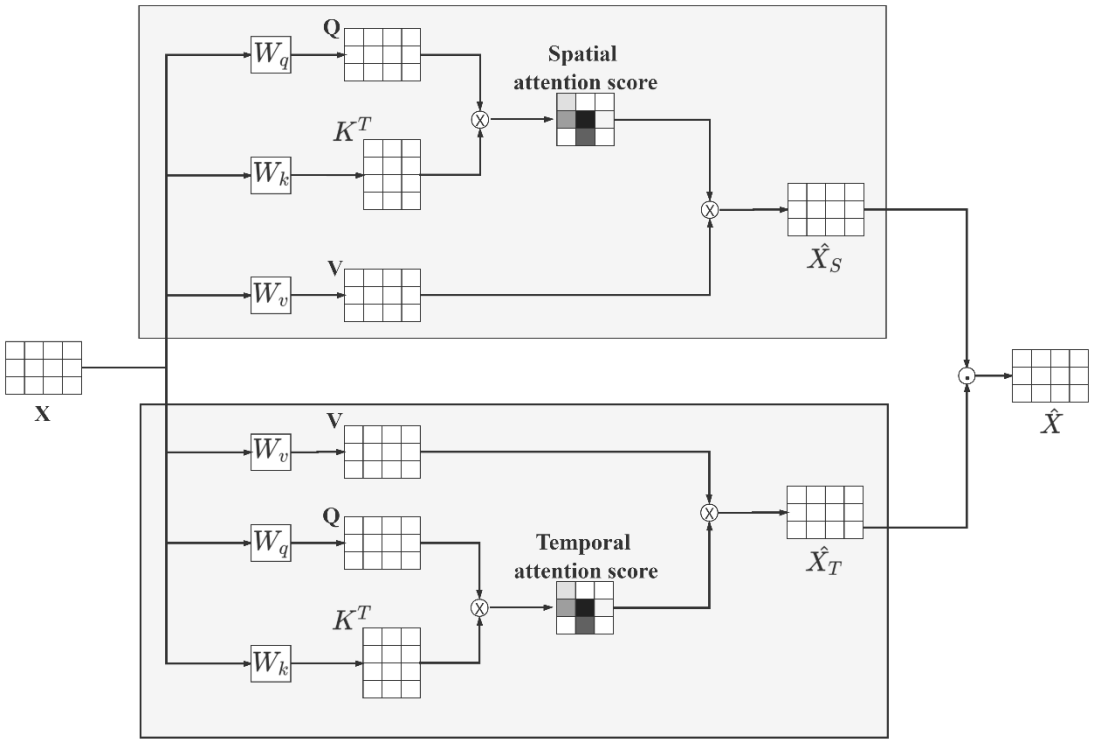


Figure 3.17 Spatial-temporal attention mechanism

As shown in Figure 3.17, the self-attention mechanism can effectively separate spatial-temporal feature vectors. Then calculate the temporal attention matrix and the spatial attention matrix using their relatively independent matrix calculation methods. Thirdly, the feature information is weighted and converged to all spatial-temporal positions by dot multiplying the V value, that is, the dot multiplying value matrix. This can adaptively capture the temporal and spatial dynamic correlation of traffic flow data. Finally, the separated two sets of eigenvectors are merged into eigenvectors containing spatiotemporal attention through Hadamard matrix multiplication: $\hat{X} =$

$\hat{X}_S \odot \hat{X}_T \in \mathbb{R}^{N \times C \times T}$.

3.5.5 Spatial-temporal self-attention graph convolution models

The modeling idea of the spatiotemporal graph neural network model based on the attention mechanism is similar to the previous chapter. The feature extraction model in the spatial dimension still uses the GCN model. In the temporal dimension, GRU or TCN models are also used to extract the trend of traffic flow data. The difference between the models is that the spatiotemporal graph neural network model based on the self-attention mechanism adds a spatiotemporal separation attention extraction module before the spatiotemporal feature extraction module. According to the spatial attention matrix, the feature information of other nodes is gathered into the features of all nodes to capture the dynamic correlation effects of different nodes. Then, according to the temporal attention matrix, the feature information of other moments is aggregated into the features of all moments to capture the dynamic correlation effects of different time steps. Finally, the feature vector $\hat{x}_t$ containing the temporal dimension of attention and the feature vector of the spatial dimension $\hat{x}_s$ are merged to obtain the spatiotemporal attention feature vector $\hat{x}$. Before feature extraction, each layer of spatiotemporal module passes through a layer of spatiotemporal attention module. It combines the spatiotemporal attention mechanism module and spatiotemporal module into a spatiotemporal self-attention graph convolution module. At the same time, this work uses a very popular residual network in the field of convolutional neural network models to solve the problem of network degradation caused by gradient dispersion or gradient explosion during model training. Finally, by integrating the spatial-temporal modules one by one, a traffic flow-oriented spatial-temporal self-attention graph convolution prediction model is constructed.

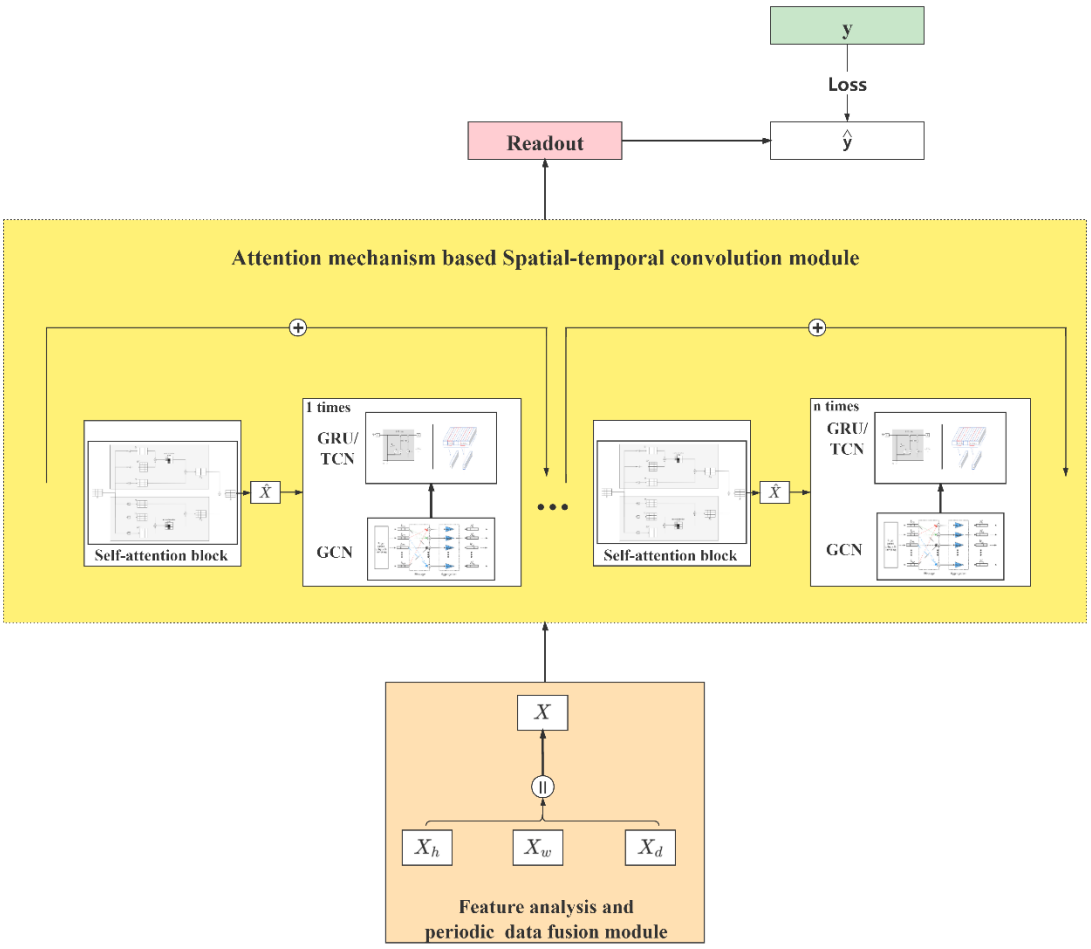


Figure 3.18 Spatial-temporal attention graph neural networks model

In short, three periodic components are merged into one component through dimensionality reduction or mapping of input data. This simplifies the model and reduces the computational complexity. The temporal and spatial features of traffic data can be captured well through the Spatial-temporal convolution module. By combining spatiotemporal self-attention mechanism and spatiotemporal module, the models we designed can effectively obtain the dynamic spatiotemporal features of data. Then, by stacking multiple spatiotemporal modules, more abundant spatiotemporal dynamic dependent features can be extracted.

Moreover, the use of residual connections makes the training process more stable. Finally, a fully connected layer is passed to ensure that the component's output has the same dimension as the predicted target. Through this process, the models can get the final prediction result.

3.6 Summary

In this chapter, this thesis starts from the reality of traffic flow prediction and discusses the pros and cons of specific methods needed for prediction. Based on existing research and prediction methods, this work proposes four models for traffic flow prediction. Two are traffic flow prediction models based on spatial-temporal graph convolutional neural networks. The other two are improved on this basis, adding a spatial-temporal graph convolutional neural network model with a self-attention mechanism. And, in this chapter, the reasons for choosing these four models are also discussed. Moreover, the rationality and implementation ability of this choice are analyzed. This thesis will use the actual traffic data set to simulate and compare the models in the next chapter.

Chapter 4

Simulation Studies

This chapter conducted extensive simulation research on the four proposed graph neural network-based models under the Pytorch framework. The MST-GCRN, MST-GCTN, MST-AGCRN, and MST-AGCTN models have been experimentally verified on two actual traffic data sets. First, introduce the model-building platform, traffic data set, and related processing of the data set used in this work. Then the relevant parameter settings in the simulations are introduced, and the commonly used indicators in traffic flow forecasting are given. Then, by comparing some baseline methods to evaluate the performance of the model proposed in this thesis. We have selected several representative models in classical statistical theories and analytical models, traditional machine learning methods, and deep learning methods. By predicting and evaluation criteria, the performance of these models on the same data set is compared objectively with the models proposed by this thesis. At the same time, to verify the importance of spatial-temporal dual-module modeling, this chapter specifically validates the proposed method in this aspect. The model's actual performance is used to demonstrate the influence of spatial correlation modeling and temporal correlation modeling on prediction accuracy. Then, based on the different performance of different models on the same data set, node and forecast period. This work also tested the four models' prediction capabilities proposed in the road network structure for different traffic flows and time steps. By comparing the predicted value and the real value, the actual prediction ability of the models and some of their advantages and disadvantages are clearly displayed and discussed. Finally, the performance of the model in training is also mentioned. Through the different performances of the TCN module and the GRU module in terms of time performance, we discussed our basic modeling ideas, the rationality and feasibility of modeling, the models' deficiencies, and the direction that needs further research. This chapter will

show the simulation results through some graphs, so that readers have a clearer understanding.

4.1 Simulation Environment

4.1.1 Hardware environment

This thesis is based on the Pytorch framework for modeling and verification. In the actual simulation, this work uses a Windows system laptop as the Simulation platform. The main hardware configuration information of it is shown in Table 4-1.

Table 4-1 Basic hardware description

CPU	Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99GHz
RAM	20 GB (Crucial DDR4 2666MHz)
Storage	Samsung MZVLB256HBHQ-000L2 (256GB/SSD)
Graphics	Nvidia GeForce MX250 (4GB)

4.1.2 Software environment

This thesis uses PyTorch as the modeling framework for the four traffic flow prediction models designed in this work. The specific configuration of the software environment is shown in Table 4-2. Among them, it calls third-party libraries such as Torch, Math, Numpy.

Table 4-2 Software configuration

Item	Version
OS	Windows 10 Home edition & Ubuntu 18.04 LTS
Anaconda3	V4.3.1
Python	V3.7.0
PyTorch	V0.4.1
CUDA	V10.0

The reason for choosing PyTorch as the modeling framework is its strong technical accumulation, stable performance, and outstanding performance in machine learning. PyTorch is a Python open-source machine learning library based on Torch. PyTorch is essentially a replacement for Numpy, and it supports GPU and has advanced features

that can be used to build and train deep neural networks [125]. Compared with TensorFlow, PyTorch is more concise and easy to use, which is very suitable for small-scale model building and testing. Moreover, PyTorch is very suitable for high-performance numerical calculations.

4.1.3 Simulation data set

This thesis selects the public traffic flow data set provided by the PeMS (Performance Measurement System) of the California Department of Transportation, the public traffic data platform, as the input data set.

PeMS (http://pems.dot.ca.gov/) data set is the most commonly used data set in the field of transportation research. PeMS is an intelligent traffic monitoring system developed by the California Department of Transportation. It is mainly composed of vehicle sensors, which cover most of the highway network in California. The PeMS system collects real-time data from more than 45,000 detectors. It processes the raw data collected by the sensors, then aggregates them according to time periods such as 5 minutes, 10 minutes, and 30 minutes, and uploads them to the platform for public disclosure. At the same time, PeMS provides Archived Data User Service (ADUS). It provides users with more than ten years of historical data. The data of this platform covers a wide range of space. The data indicators are relatively comprehensive, and the data missing rate is low. Therefore, it has become the preferred traffic flow data platform for many management, design and research personnel. Figure 4.1 shows the Web interface of the PeMS system. The red dots in the figure are the different detectors.

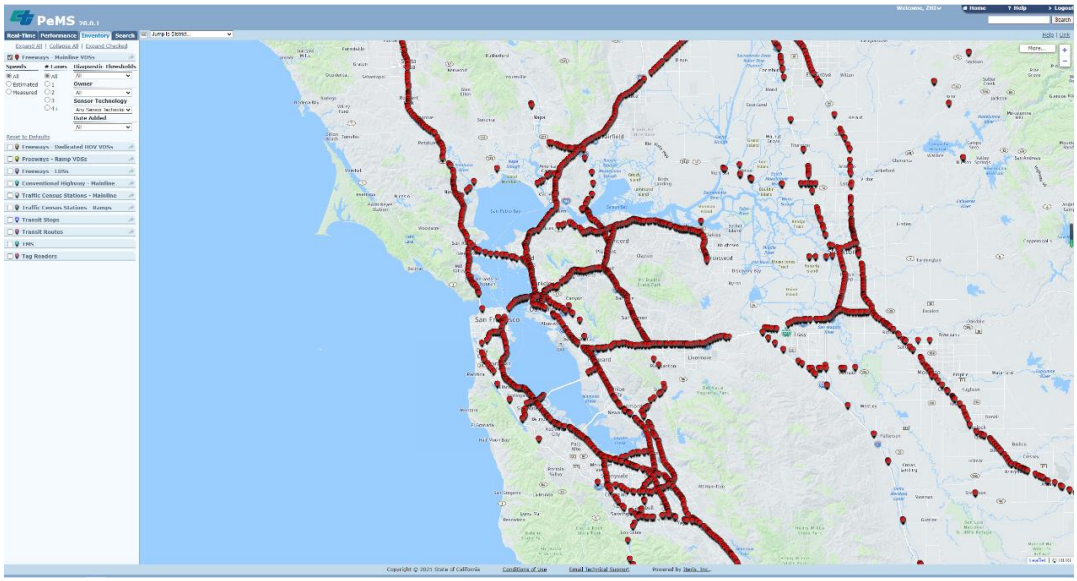


Figure 4.1 PeMS system web interface

At present, a large number of researchers have conducted related research in the field of transportation on PeMS. Huang et al. [126] used multiple detector traffic flow data on PeMS to study a short-term traffic flow prediction method based on deep learning. Oh et al. [127] selected 16 detectors on the SR-78E highway on the PeMS system to study the traffic state prediction. Lopez et al. [128] used the I5 highway to study traffic congestion. Moylan et al. [129] used multiple detectors in the San Francisco Bay Area to study the impact of congestion status, traffic demand, road conditions, and weather conditions on Travel Time. Wu et al. [36] data source is the I-405N highway on the PeMS system and proposed the DNN-BTF model, which obtained the best prediction effect at the time. Therefore, in order to better conduct comparative research, the data selection of this thesis will be consistent with many previous studies in data selection and data division. The PeMS detector collects traffic flow data every 30s and gathers them in different lengths of time to provide services for users. The original data format is shown in Figure 4.2.

								Data Quality	
5 Minutes	Lane 1 Flow (Veh/5 Minutes)	Lane 2 Flow (Veh/5 Minutes)	Lane 3 Flow (Veh/5 Minutes)	Lane 4 Flow (Veh/5 Minutes)	Lane 5 Flow (Veh/5 Minutes)	Lane 6 Flow (Veh/5 Minutes)	Flow (Veh/5 Minutes)	# Lane Points	% Observed
07/28/2021 09:00	125.0	120.0	102.0	115.0	51.0	89.0	602.0	6	100.0
07/28/2021 09:05	160.0	129.0	81.0	111.0	65.0	113.0	659.0	6	100.0
07/28/2021 09:10	127.0	125.0	78.0	110.0	54.0	101.0	595.0	6	100.0
07/28/2021 09:15	135.0	125.0	69.0	107.0	60.0	102.0	598.0	6	100.0
07/28/2021 09:20	129.0	122.0	82.0	119.0	55.0	95.0	602.0	6	100.0
07/28/2021 09:25	134.0	112.0	71.0	102.0	43.0	95.0	557.0	6	100.0
07/28/2021 09:30	126.0	110.0	69.0	97.0	52.0	94.0	548.0	6	100.0
07/28/2021 09:35	119.0	112.0	82.0	119.0	62.0	83.0	577.0	6	100.0
07/28/2021 09:40	119.0	108.0	63.0	104.0	47.0	74.0	515.0	6	100.0
07/28/2021 09:45	120.0	129.0	72.0	120.0	64.0	84.0	589.0	6	100.0
07/28/2021 09:50	112.0	99.0	85.0	99.0	53.0	85.0	533.0	6	100.0

Figure 4.2 Example of traffic flow data

The first column is a time stamp with 5-minute intervals. Columns 2 to 5 are the respective traffic flows on the four lanes. The sixth column is the total road traffic flow. The last column is the percentage of data quality given by PeMS. The table shows the section where the VDS1214209 detector is located on highway I-405S, a four-lane section. Figure 4.3 shows the street scene near the VDS1214209 detector on Google Maps. The leftmost is two High Occupancy Vehicle (HOV) lanes, the rightmost is the ramp, and the middle four are the main lanes. From left to right correspond to Lane1-Lane4 in Figure 4.2. Lane1 is close to the central separation zone and is usually a passing lane or an expressway. Lane4 is close to the shoulder of the road, and vehicles will continue to flow in. To reflect the road traffic more truly, this work uses the total road traffic flow.

Figure 4.3 Google Map street view near the VDS1214209 detector

This thesis uses PeMSD8 and PeMSD4 as the benchmark data set to verify the method and models proposed in this thesis. The acquisition frequency is the 30s/time. PeMS also provides time series data of 30s, 5min, 1h and other time slices for research needs. Studies have shown that predicting traffic flow in congested traffic conditions with a time interval of 5 minutes can provide drivers with the most effective help [36]. Therefore, this work uses a 5min time slice data set for research and prediction. This thesis will select the traffic flow, traffic flow density, and average vehicle speed as the characteristic dimensions of the data set. The specific data information is as follows:

1. PeMSD4 data set: The data comes from the traffic data of San Francisco Bay Area

highways. The transportation network in this area includes 29 highways and a total of 3848 data collection devices. The collection time range is the traffic data of 59 days from January to February 2018. In this work, the first fifty days of data will be used as the training set, and the last nine days of data will be used as the test set.

2. PeMSD8 data set: The data comes from the traffic data of the highway in San Bernardino, including eight roads and a total of 1979 collection devices. The time range is from July to August 2016, a total of 62 days of data records were collected. The first fifty days are used as the training set, and the last twelve days are used as the test set.

4.2 Data Pre-processing

4.2.1 Data analysis

1. Temporal correlation analysis

As shown in Figure 4.4, using the overall traffic flow data in the test set, it can be seen that the traffic flow trend has a significant periodicity and fluctuation law.

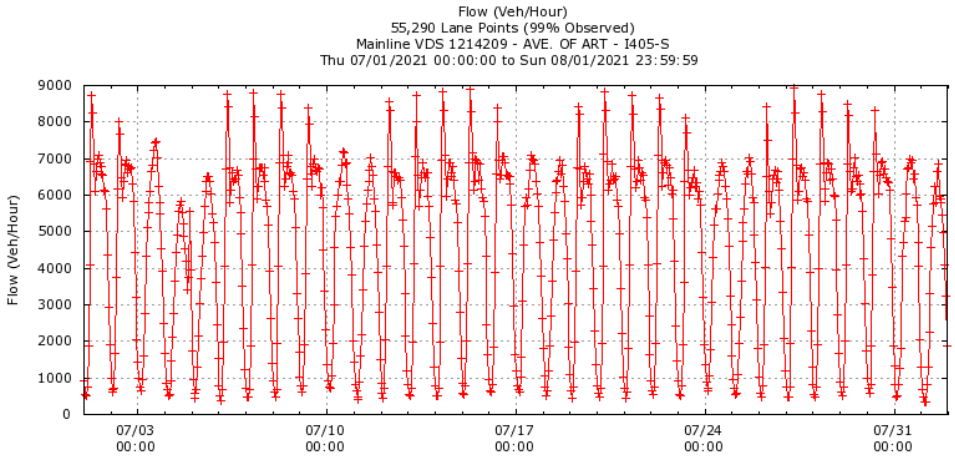


Figure 4.4 Historical traffic data of the VDS 1214209 detector in July 2021

Analyze the traffic data of a certain working day, as shown in Figure 4.4. It can be seen that the traffic flow of this working day is obviously more vehicles during the day and fewer vehicles at night, which is consistent with the law of human activities.

There are two peaks in the vehicle number curve in a day, and the number near the peak fluctuates sharply. Comparing the traffic flow data of different dates, it also has a certain degree of randomness, but the overall movement pattern is obvious and periodic. It can also be seen from the time points and data on the graph that the change curve waveform of traffic flow is similar to the fact that people commute and participate in other social activities.

2. Spatial correlation analysis

The flow of traffic circulates in space. Therefore, the traffic flow of a place is not only related to its own time change characteristics, but also closely related to its spatial distribution characteristics. This work arbitrarily selects the traffic flow data of 4 adjacent detection stations on the same day for spatial correlation analysis. As shown in Figure 4.5, although the traffic flow of different detection stations on the same day is slightly different, the overall distribution trend is roughly the same. Moreover, the higher the correlation of the distance between the two stations, the closer the traffic flow connection between the two points.

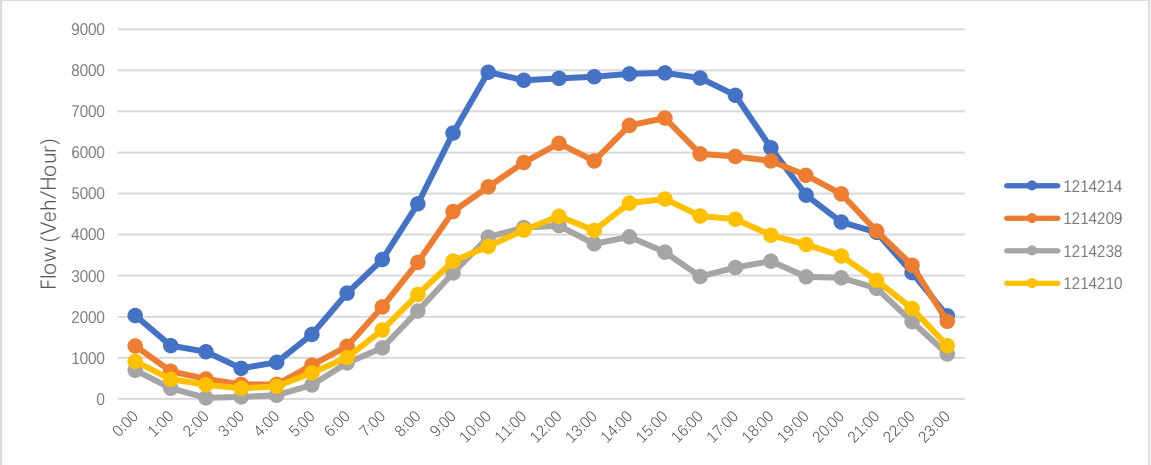


Figure 4.5 Traffic flow on the same day at different detection stations

4.2.2 Data filling

	A	B	C	D	E	F	G	H
1	5 Minutes	Lane 1 Flow (V	Lane 2 Flow (V	Lane 3 Flow (V	Lane 4 Flow (V	Flow (Veh/5 Mi	# Lane Points	% Observed
449	7/27/2021 13:15	93	90	90	94	367	4	100.00
450	7/27/2021 13:20	123	105	102	97	427	4	100.00
451	7/27/2021 13:25	99	106	95	106	406	4	100.00
452	7/27/2021 13:30	102	84	102	102	390	4	100.00
453	7/27/2021 13:35	93	105	87	105	390	4	100.00
454								
455	7/27/2021 13:45	105	101	113	111	430	4	100.00
456	7/27/2021 13:50	110	104	99	110	423	4	100.00
457	7/27/2021 13:55	106	109	107	92	414	4	100.00
458	7/27/2021 14:00	95	116	104	103	418	4	100.00
459	7/27/2021 14:05	95	107	99	95	396	4	100.00
460	7/27/2021 14:10	102	103	102	106	413	4	100.00
461	7/27/2021 14:15	123	100	105	119	447	4	100.00
462	7/27/2021 14:20	122	120	117	120	479	4	100.00
463	7/27/2021 14:25	122	100	110	116	448	4	100.00
464	7/27/2021 14:30	110		121	111		4	
465	7/27/2021 14:35	100	107	101	117	425	4	100.00
466	7/27/2021 14:40	115	124	105	107	451	4	100.00
467	7/27/2021 14:45	114	107	107				
468	7/27/2021 14:50	123	112	102	128	465	4	100.00

Figure 4.6 Schematic diagram of missing data

In the data processing, it is found that some traffic flow data of the detection station is missing, as shown in Figure 4.6. The lack of traffic data will have a greater impact on the prediction of subsequent data analysis so that this work will repair the missing part of the data. To make the filling data more accurate and able to combine the characteristics of both historical data and the data of the day, this thesis uses the KNN algorithm to fill the missing data. The main calculation idea of KNN is: calculating the distance between the target value and the data record in the data set. Then select the k values with the smallest distance from the target data as the nearest neighbors of the target value. The weighted average or mode of the k nearest neighbor values is the estimated value of missing target data.

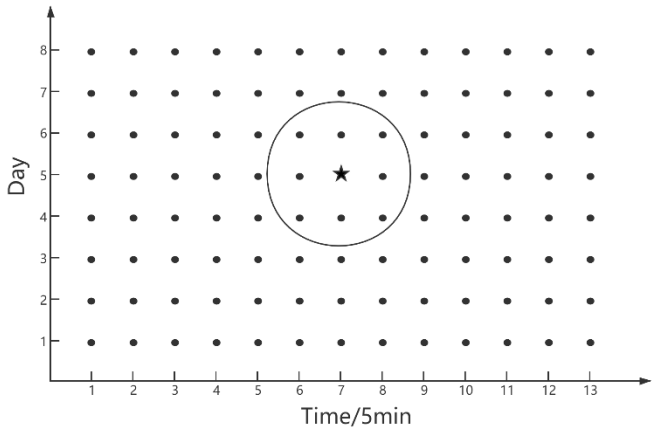


Figure 4.7 Schematic diagram of filling data with KNN algorithm

Divide the time into two-dimensional data according to 5 minutes as a unit. The horizontal axis is 288 units of time in a day, and the vertical axis is days, as shown in

Figure 4.7. Then the nearest neighbor algorithm is to determine the points in the circle in the figure. In this way, the data in the entire two-dimensional coordinates are taken into account. That is, both historical data and current data are taken into consideration. For the choice of distance, this work uses Manhattan distance:

$$d\big((x_1, y_1), (x_2, y_2)\big) = |x_1 - x_2| + |y_1 - y_2| \quad (4\text{-}1)$$

After parameter adjustment, K=3 is finally selected as the final result.

4.2.3 Node screening

To better conduct comparative simulations. In this work, the node screening method is consistent with the node screening method adopted by Guo et al. [18] in the study of MCSTGCN and ASTGCN models. The detectors in the PeMSD4 data set and the PeMSD8 data set are screened separately, and some detectors that are too close are removed to ensure that the distance between the detector nodes is greater than 3.5 miles. Due to the characteristics of the highway dataset, the data similarity of the nodes with closer distances is high. While increasing the complexity of the data, it cannot help the spatiotemporal model to extract feature data very well.

After processing, the PeMSD4 data set retains a total of 307 data collection stations related to traffic flow data information. The PeMSD8 data set retains a total of 170 data collection stations related to traffic flow data information.

4.2.4 Data standardization

This work uses the Z-score method to standardize the data. Every deep learning model must perform data normalization, which is one of the most basic tasks of deep learning. When forecasting traffic flow, data using different evaluation indicators often have different magnitudes and units. This situation will affect the temporal and spatial feature extraction of traffic flow data. Therefore, the data in the traffic flow data set must be normalized to the same magnitude for subsequent feature extraction. The standard deviation and mean value of the data are uniformly processed by the

standardized method so that the data in the data set conforms to the standard normal distribution. After processing, the mean value of the data is equal to 0, and the standard deviation is equal to 1. Normalized data are all values with a size between $[0,1]$, which facilitates data processing and simplifies complex data [130]. The formula is as follows:

$$\hat{x} = \frac{x-\mu}{\delta} \quad (4\text{-}2)$$

In the above formula, x is the traffic flow data obtained by the traffic sensor, μ is the mean of the overall traffic flow data, and δ is the standard deviation of the overall traffic flow data.

4.3 Data Set Division and Evaluation Criteria

4.3.1 Data set division

As mentioned in node screening, the PeMSD8 data set retains 170 data nodes, and the PeMSD4 data set retains 307 data nodes. According to the division and fusion method of periodic components discussed in section 3.3.2. This thesis keeps the size settings of the recent periodic dependency, daily periodic dependency, and weekly periodic dependency consistent with the settings in the MSTGCN and ASTGCN models [18]. Then, perform data fusion on the input feature data according to the data fusion method in section 3.3.3. Since the data sample collection interval is selected to be 5 minutes, the duration of the model's prediction target is 60 minutes. Therefore, the dimension of the input data is $(\frac{60}{5}\times 3+\frac{60}{5}\times 1+\frac{60}{5}\times 1)\times C = 60\times C$. C is the channel dimension (the number of different types of feature information).

According to the results of data processing and the basic rules of data set division in the field of machine learning, this work divides the data set into a proportion of 60% for the training set, 20% for the validation set, and 20% for the test set. The feature input matrix dimension of the PeMSD4 data set is spatial dimension (307 nodes) × channel dimension (3 traffic flow features information) × temporal dimension (60

periodic time data points at 5-minute intervals). The feature input matrix dimension of the PeMSD8 data set is spatial dimension (170) × channel dimension (3) × temporal dimension (60). The output dimension of PeMSD4 is 307×12 (12 time steps of 307 nodes with a time window of 60 minutes) of predicted information (traffic flow). The output dimension of PeMSD8 is 170×12. The description of the data set is shown in Table 4-3.

Table 4-3 Dataset description

Datasets	Number of sensors	Time range	Number of samples
PeMSD4	307	1/1/2018-2/28/2018	16992
PeMSD8	170	7/1/2016-8/31/2016	17856

4.3.2 Evaluation criteria

To evaluate the pros and cons of traffic flow prediction models, evaluation criteria are usually used. Commonly used evaluation indicators are: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Relative Error (MRE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE). The specific calculation formula is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y} - y_i|^2 \quad (4\text{-}3)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}|\hat{y} - y_i|^2} \quad (4\text{-}4)$$

$$MRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}-y_i|}{y_i} \quad (4\text{-}5)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y} - y_i| \quad (4\text{-}6)$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{|\hat{y}-y_i|}{y_i} \quad (4\text{-}7)$$

In the formula, y_i represents the true value, $\hat{y}$ represents the predicted value, and n represents the total number of samples.

RMSE measures the error between the predicted and true values and squares the error value, making RMSE more sensitive to outliers. MAPE not only considers the error

between the predicted value and the true value but also considers the difference between the error and the true value. Proportion. Some researchers believe that it is more practical to use MAPE as a performance indicator because the peaks and valleys of the traffic flow may be quite different. At the same time, the use of MAPE can evaluate the effects of different models on different data sets to a certain extent [26]. However, when the real traffic flow is zero ($y_i = 0$), MAPE cannot be calculated. This work will use RMSE and MAE as the main evaluation indicators, and at the same time, it will be supplemented by the observation of changes in MAPE. When calculating MAPE, zero-value samples will be eliminated.

4.3.3 Baseline method

This thesis will compare and analyze with the following methods: HA (History Average), Auto-regressive integrated moving average (ARIMA), Long and short-term memory neural network model (LSTM). In addition, a Multi-Component Spatial-Temporal Graph Convolution Networks (MSTGCN) and an attention based spatial-temporal graph convolutional network (ASTGCN). For the settings of HA, ARIMA, LSTM and GRU, refer to the prediction results of Guo et al. [18]. MSTGCN, ASTGCN refer to the default settings of the original author's paper [18].

(1) HA (History Average): The average value of historical traffic is used as the predicted value of the traffic to be predicted.

(2) ARIMA (Autoregressive Integrated Moving Average) [131]: This model regards the data sequence formed by the forecast object over time as a random sequence, based on the autocorrelation analysis of the time series, and predicts future values through the historical data of the time series.

(3) LSTM [71]. LSTM alleviates the problem of gradient disappearance to a certain extent through the "gate" mechanism. As a common method of time series prediction, it takes the flow of the previous moment as input to predict the flow of the next moment.

(4) MSTGCN [18]: A spatial-temporal prediction model that contains a convolution module of spatial-temporal graphs similar to a sandwich structure. The road network structure models the relationship between the stations as the basis of graph convolution. A good prediction accuracy has been achieved on the traffic flow prediction issue.

(5) ASTGCN [18]: A spatiotemporal graph convolution model based on spatiotemporal attention mechanism. It uses the road network structure to model the relationship between stations as the basis for graph convolution. A good prediction accuracy has been achieved on the problem of road flow prediction.

4.4 Analysis of Simulation Results

4.4.1 The parameter settings of the prediction models

The main parameters involved in the simulation of the traffic flow prediction models (MST-GCRN, MST-GCTN, MST-AGCRN, and MST-AGCTN) proposed in this thesis are:

(1) As mentioned in the GCN-based spatial module modeling section, Chebyshev graph convolution is used in the graph convolution module, where the convolution kernel size K is 2, and the number of convolution kernels is 64.

(2) The number of layers of Recurrent neural network. In temporal dependency modeling, the sequence-to-sequence model is used to predict the temporal dimension. The encoder and decoder in the model are both recurrent neural networks with GRU. The selection of the number of layers of the recurrent neural network should not be too large. In the simulation of this work, both the encoder and the decoder are equipped with two layers of recurrent neural networks, which achieves the best results.

(3) The number of GRU in the recurrent neural network. The recurrent neural network of each layer of the encoder and decoder has many GRU. The selection of the number of GRU is generally an exponent of 2, and it is generally appropriate to set it between

16 and 128. If it is too large, it will increase the complexity of the calculation and make the training time-consuming. If it is too small, it will also affect the effectiveness of the model. In the simulation of this work, the best performance is reached when it is set to 24.

(4) The size of the convolution kernel of the temporal convolution model is $k = 4$, and the number of convolution kernels is 32. The dilation factor $d = 2^{n-1}$ is the expansion factor of the n-th spatial-temporal convolution module.

(5) Common parameters of deep learning. The number of layers of the spatial-temporal module is set to 5. Too many layers will cause overfitting, and too few layers will cause insufficient feature learning. Epoch is set to 100, and early stopping technology is used to avoid overfitting. The initial learning rate is set to 0.003. Starting from the 5th epoch, the epoch learning rate decay interval is 5, 20, 40, 70. Each time the learning rate decays to 0.3 times the initial learning rate. In this thesis, Mini-Batch is used in training, the Batch-Size is set to 64, and ReLU is used as the activation function of the neural network. The loss function uses Mean Absolute Error (MAE), and the optimizer chooses Adam.

4.4.2 Case study 1: The impact of traffic flow volume on the accuracy of prediction Models

This section demonstrates and discusses the impact of traffic flow volume at different spatial-temporal locations on the model's prediction accuracy to enable readers to understand the performance of the four models designed by this work on the PeMSD4 and PeMSD8 data sets. Our models all have the ability to predict traffic flow, but their prediction accuracy differs significantly when dealing with different volumes of traffic flow at different spatial-temporal locations. As mentioned in the data processing section, the minimum interval for data collection of the data set is 5 minutes. So in the following discussion, we will use the time step as the unit of time, and each time step represents 5 minutes. Therefore, the one-day predict curve will be

divided into 288-time steps, and the one-hour predict curve will be divided into 12-time steps. Based on this, the actual effect of the models in extracting traffic flow is discussed by comparing the predicted value with the real value.

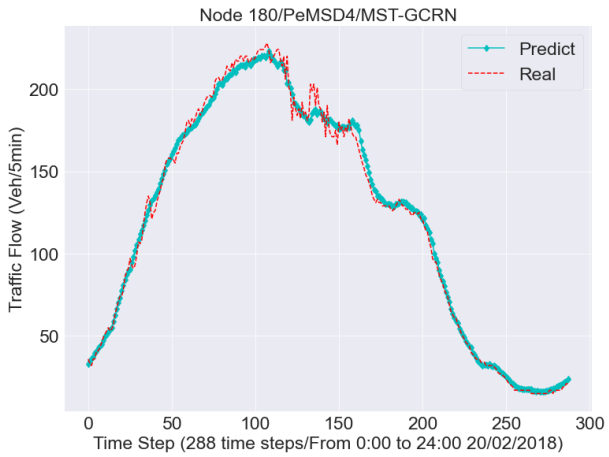


Figure 4.8 The MST-GCRN prediction accuracy on PeMSD4-node180

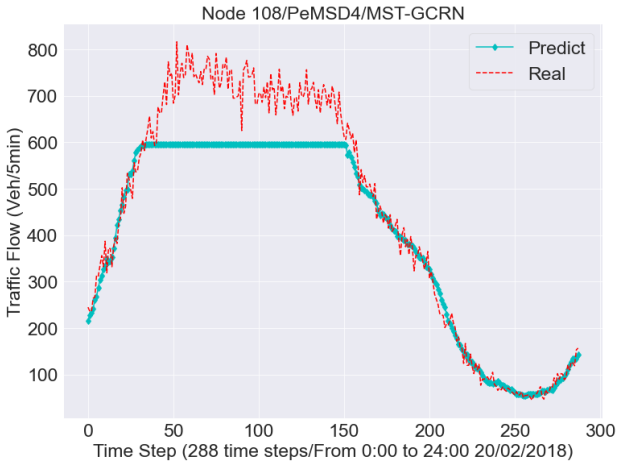


Figure 4.9 The MST-GCRN prediction accuracy on PeMSD4-node108

The figure above shows the prediction accuracy of the MST-GCRN model on the PeMSD4 data set. Figures 4.8and 4.9 show the comparison between the predicted data and actual data of nodes 180 and 108 on February 20, 2018. The red is the value of the actual traffic flow, and the green is the predicted result. The X-axis coordinates represent time steps, and each time step is 5 minutes. So a day is 288 time steps. We found that the prediction performance of the MST-GCRN model will be affected by the volume of traffic flow. At nodes where the volume of traffic flow is small, the model performs better.

On the contrary, The performance of the model has decreased. As shown in Figure 4.9,

especially in the period when the traffic flow value is large, the model's prediction accuracy decreases significantly. As shown in Figure 4.10 (a) and (b), this phenomenon is also reflected in our MST-AGCRN model.

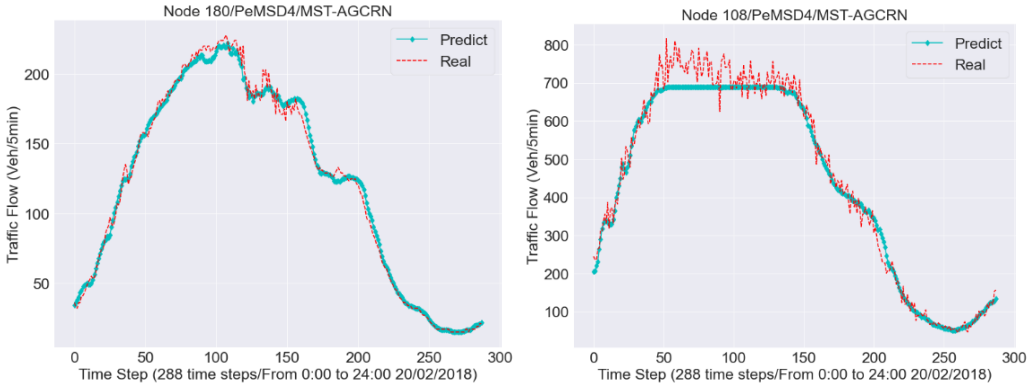


Figure 4.10 (a/b) The MST-AGCRN prediction accuracy on PeMSD4-node180/108

Judging from the prediction performance of the MST-AGCRN model, during rush hours, the model's prediction accuracy is worse than usual. To analyze this phenomenon more fine-grained. We selected the models' prediction performance during the peak hours of these two nodes on this day. Figures 4.11 and 4.12 show the comparison between the predicted data and actual data of nodes 180 and 108 within one hour from 11:00 to 12:00 on February 20, 2018, on the MST-GCRN model.

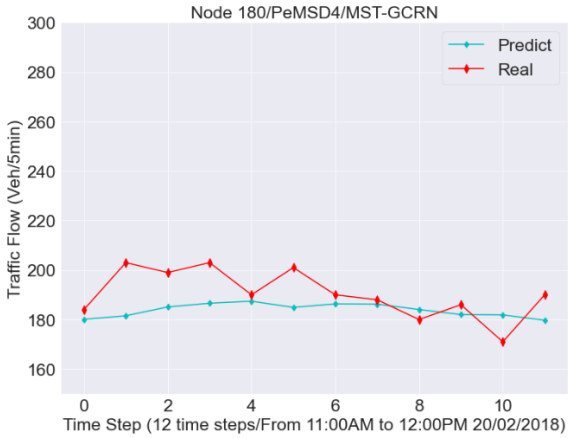


Figure 4.11 The MST-GCRN prediction accuracy on PeMSD4-node180-1hour

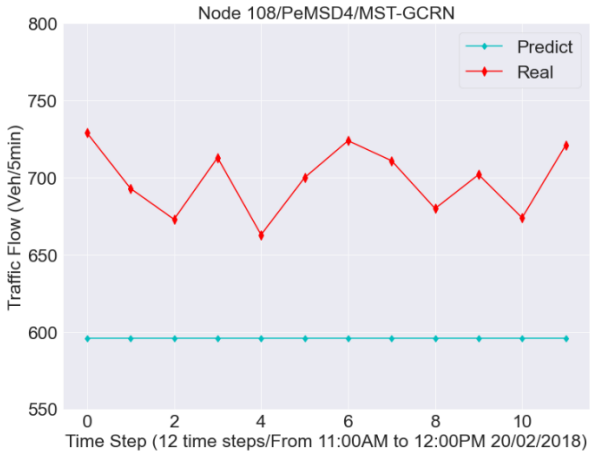


Figure 4.12 The MST-GCRN prediction accuracy on PeMSD4-node108-1hour

The performance of the MST-GCRN model during peak hours reflects this phenomenon more directly. From the degree of fit between the predicted value and the true value and the fluctuation of the predicted value, it can be seen that when the flow volume is small, the model's predictive ability is stronger. On the contrary, the gap between the predicted and true values is obvious, and the predicted value does not fluctuate over time. This conclusion can also be clearly reflected in Figure 4.13 (left) and (right). It shows the comparison between the predicted data and actual data of nodes 180 and 108 from 11:00 to 12:00 on February 20, 2018, on MST-AGCRN.

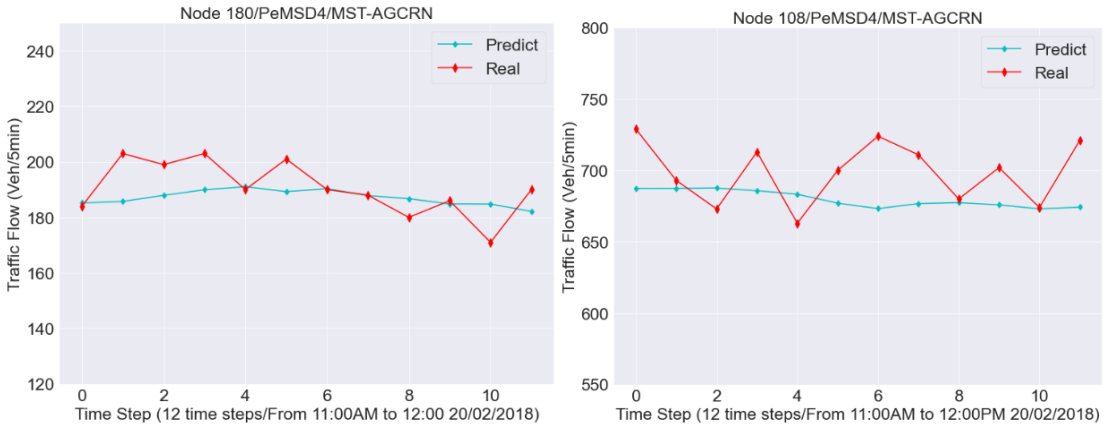


Figure 4.13 The MST-AGCRN prediction accuracy on PeMSD4-node180/108

It can be seen from the above two sets of figures that the traffic flow extraction performance of the models is better on nodes with small traffic flow volumes than on nodes with large volumes. To further confirm this conclusion, we did the same simulation on PeMSD8. The result is shown in the figure below.

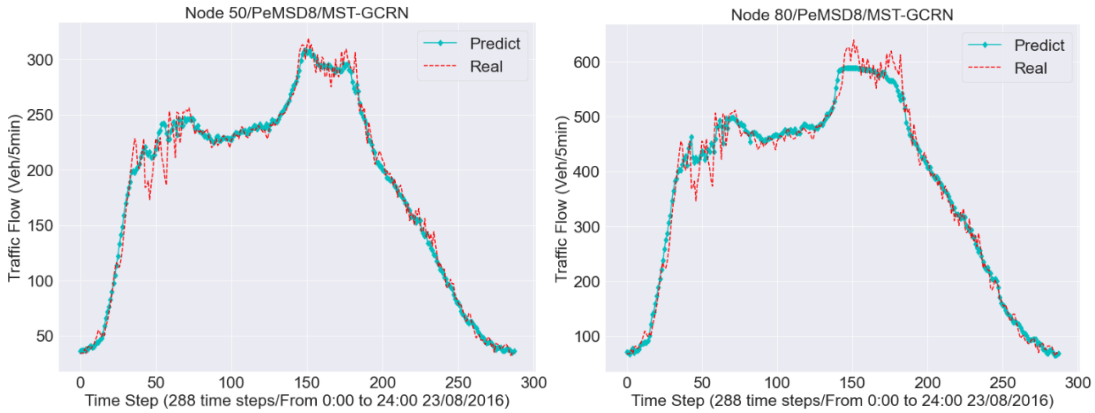


Figure 4.14 The MST-GCRN prediction accuracy on PeMSD8-node50/80

Because the PeMSD8 data set is smaller than the PeMSD4 data set. Its performance is not as obvious as the data set, but we can still clearly see this phenomenon during peak hours. As shown in the figure below, this performance is clearer within one hour.

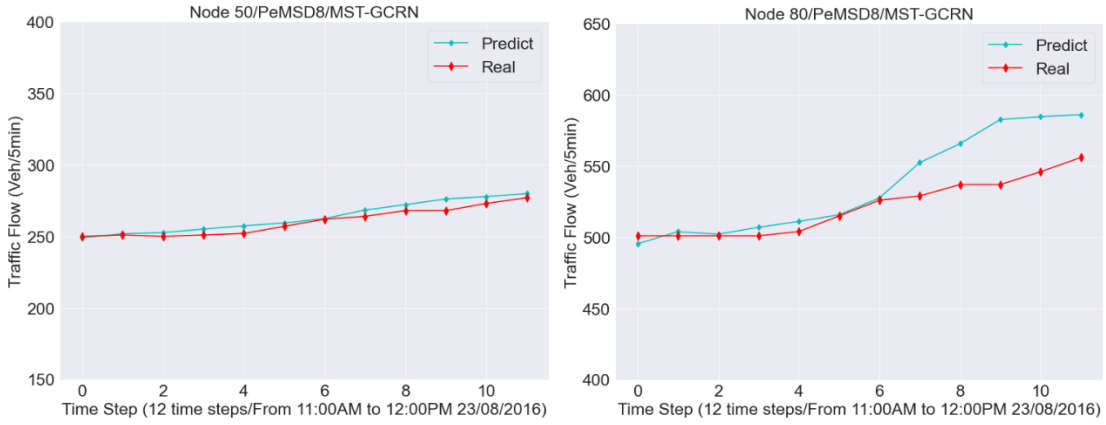


Figure 4.15 The MST-GCRN prediction accuracy on PeMSD8-node50/80-1hour

In addition, this work uses PeMSD4 data set to do the same simulation on the MST-GCTN model and MST-AGCTN model, respectively. Firstly, the simulation selects the same node and time period as the MST-GCRN model and MST-AGCTN model. The result is shown in the figure below. Among them, the blue line is the prediction accuracy of the MST-GCTN model. Green is the prediction accuracy of the MST-AGCTN model. The red is the actual value.

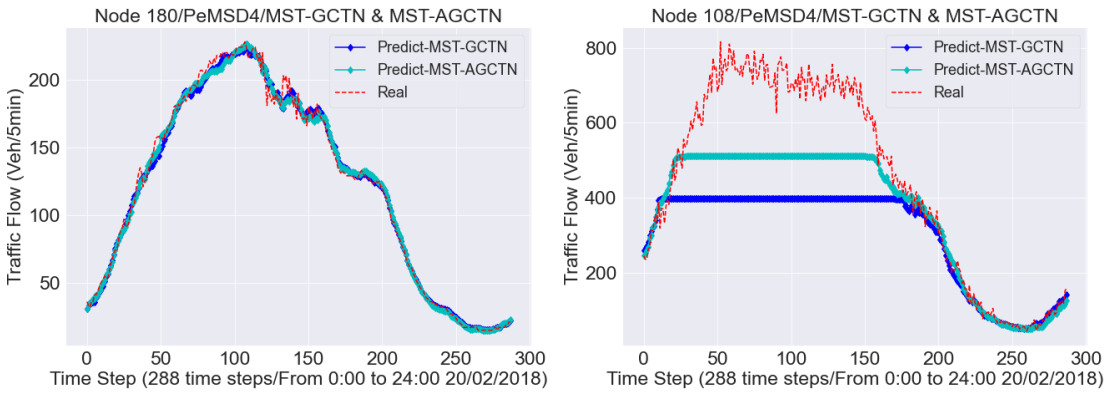


Figure 4.16 The predictive accuracy of the TCN models on the PeMSD4

Secondly, we selected two other nodes (node 20 and node 57). As shown in the figure below, the left figure shows the prediction accuracy of node 20 with a small traffic flow. The following figure shows the prediction accuracy of node 57 with a larger volume of traffic flow.

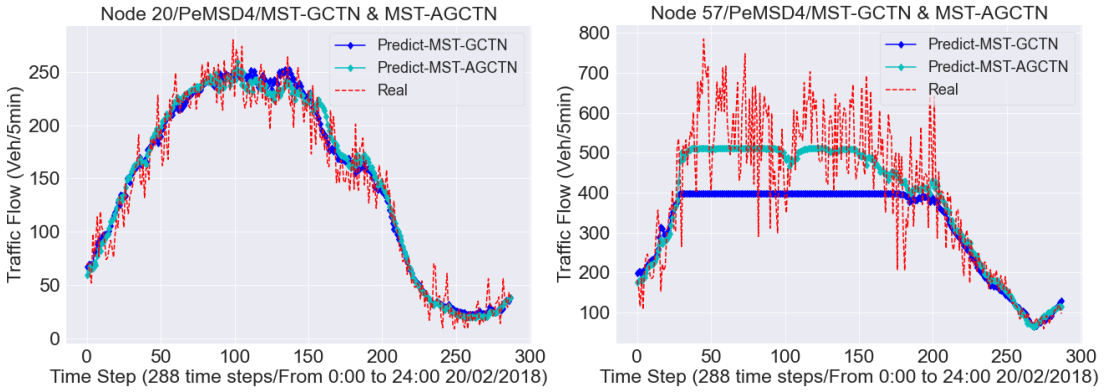


Figure 4.17 The predictive accuracy of the TCN models on the PeMSD4

From the above series of analyses, we can draw such a conclusion from case study 1. The prediction performance of our proposed models at different nodes will vary according to the volume of traffic flow of the node. At nodes with small traffic flow, the models have excellent predictive capabilities. But at nodes with large traffic flow, the predictive ability of the models has decreased. In addition, in horizontal comparison, the prediction accuracy of the model with the self-attention mechanism added is better than that of the model without the self-attention mechanism. Longitudinal comparison, the accuracy of the model based on GCN+RNN is better than the accuracy of the model based on GCN+TCN. This feature is caused by the

GCN model. The disadvantage of GCN in the embedding phase is that it cannot assign different weights to different neighbor nodes. However, in actual situations, the impact of different traffic flow nodes on other nodes is very different in temporal and spatial dimensions, and there are dynamic effects between nodes. Therefore, at nodes with high frequency or large traffic flow volume, such shortcomings will be prominent. For this reason, we have added a self-attention mechanism to capture the dynamic dependence of traffic flow to make up for the deficiencies brought by GCN. As the results show, although this characteristic still exists, the prediction results have been effectively improved. These aspects will be demonstrated in detail in the next case study.

4.4.3 Case study 2: The impact of self-attention mechanism on the accuracy of prediction Models

In this section, we will simulate the performance of the self-attention mechanism on our proposed models. From case study 1, we can see that the models perform better on nodes with smaller traffic flow. To highlight the difference in prediction accuracy between the models with the self-attention mechanism and the models without the self-attention mechanism, we chose the nodes with a larger traffic flow volume as a comparison in this case study. First, on the PeMSD4 data set, we show the prediction performance of MST-GCRN and MST-AGCRN on node 108.

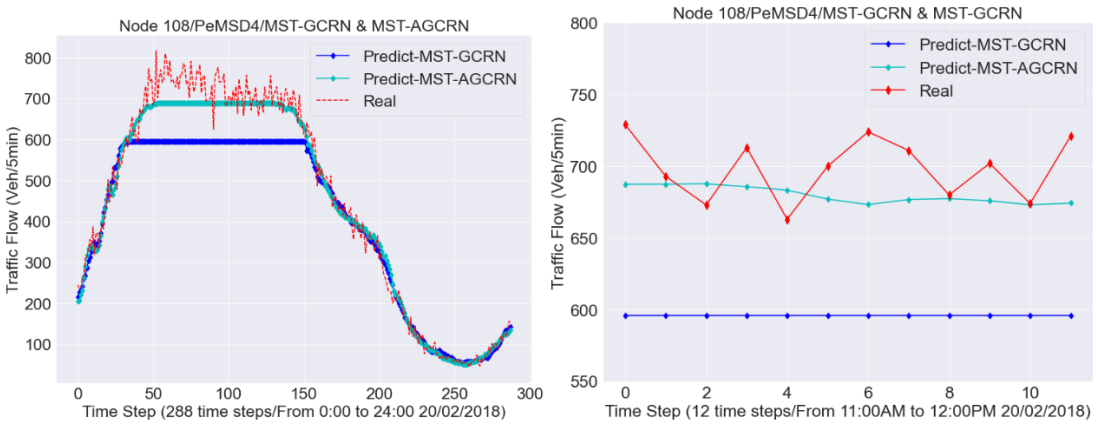


Figure 4.18 The predictive accuracy of MST-GCRN and MST-AGCRN on node 108

In Figure 4.18, the time period in the left figure is one day. The picture on the right shows the time range from 11 o'clock to 12 o'clock. The blue line is the performance of the MST-GCRN model without the self-attention mechanism, the green is the MST-AGCRN model with the self-attention mechanism, and the red is the actual traffic flow. It can be clearly seen from the figure that the traffic flow prediction accuracy of the model with the self-attention mechanism is significantly better than that of the model without the self-attention mechanism. The difference in predictive ability between the two models is most obvious during peak hours. In addition, we chose node 57 with a larger traffic flow as further proof.

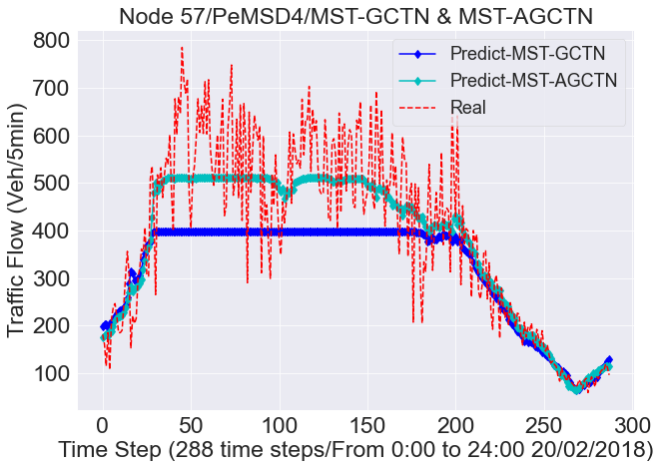


Figure 4.19 The predictive accuracy of MST-GCTN and MST-AGCTN on node 57

As shown in Figure 4.19, the predictive ability of MST-AGCTN is better than MST-GCTN. In the prediction performance during peak hours, the addition of the self-attention mechanism enhances the model's ability to extract dynamic traffic flow data and makes the prediction curve fluctuate. Next, this work performed similar simulations on the other models (MST-GCTN and MST-AGCTN) on PeMSD4 and PeMSD8 data sets.

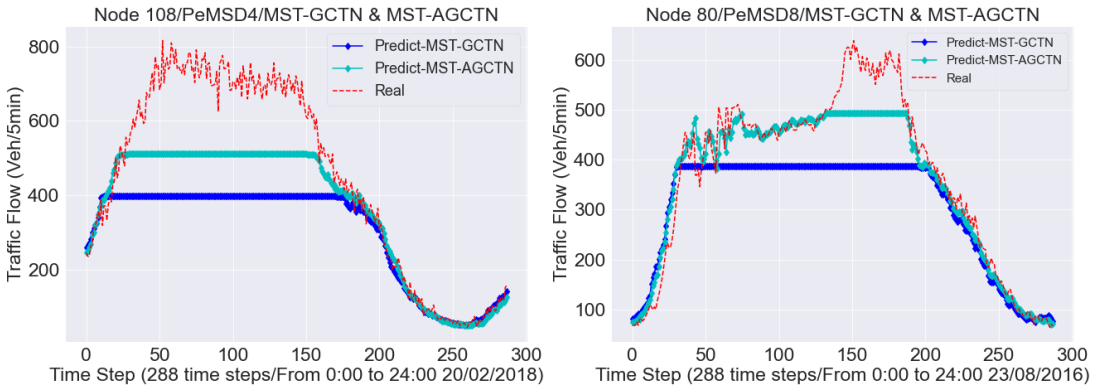


Figure 4.20 The predictive accuracy of MST-GCTN and MST-AGCTN

In Figure 4.20, the figure on the left is the performance of the MST-GCTN and MST-AGCTN models on node 108 of PeMSD4. On the right is their performance on node 80 of PeMSD8. As mentioned earlier, these two nodes are the nodes with larger traffic flow volumes in their respective data sets. It can be seen that on the two data sets, the addition of the self-attention mechanism has an obvious effect on the improvement of the model's prediction accuracy. By increasing the dynamic dependence of features, the predictive model can better capture the temporal and spatial dependence of the traffic flow between nodes. We can prove that the model with self-attention is better in terms of overall prediction accuracy through comparison and analysis. The addition of the self-attention mechanism can well enhance the model's ability to extract traffic flow, which is rich in dynamic features. In addition, in the course of this simulation, we also found differences in the predictive capabilities of the four models we proposed.

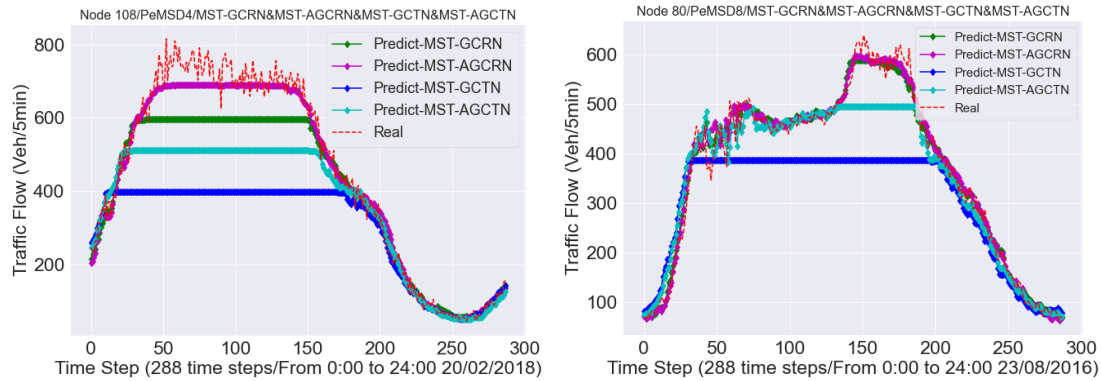


Figure 4.21 The predictive accuracy of our models

Figure 4.21 shows the prediction accuracy of our four models at node 108 of PeMSD4 and node 80 of PeMSD8. Green and magenta represent MST-GCRN and MST-AGCRN, respectively. Blue and green represent MST-GCTN and MST-AGCTN, respectively. As shown in the figure, the prediction accuracy of the MST-GCTN model and MST-AGCTN model is significantly lower than that of the MST-GCRN model and MST-AGCRN model. Moreover, the difference is even more obvious at node 108 and node 80, with larger traffic flow. The simulation results are quite different from the assumptions in the model design chapter. Originally, this simulation wanted to use the causal and dilated convolution of TCN to increase the interpretability and reliability of the model to achieve more accurate prediction results. However, the actual simulation results show significant differences. According to the analysis, this thesis believes that as the dilation factor in the TCN model gradually increases, the model accuracy will decrease. The reason is that the increase in the dilation factor leads to a larger receptive field. The benefit of the increased receptive field is that it can obtain a longer temporal dependency of the traffic flow. However, as the dilation factor increases, the number of network layers gradually deepens, and the amount of calculation and computational complexity increases. At the same time, this also increases the unpredictability of the data. When convolutional layers with the same dilation factor are stacked multiple times, part of the data may not participate in the calculation, so that the model ignores this part of useful information. As a result, the model is more difficult to train, and the model's accuracy decreases.

Secondly, the reason for the unsatisfactory performance of the MST-GCTN model and MST-AGCTN model may also be related to the preprocessing of the data set. From the perspective of data processing, there is a big difference between holiday data and weekday data due to the periodic characteristics of traffic flow data. This work believes that this is also a reason that restricts TCN from exerting its advantages. Generally speaking, there will be two peaks in traffic flow from Monday to Friday: morning peak and evening peak. The traffic volume on weekdays is significantly higher than that on weekends. There is only one peak in traffic flow for two days on

weekends, which usually occurs around noon. Suppose the model cannot explain the difference between working days and non-working days well. In that case, it will extract too much irrelevant feature information, thereby reducing the performance of the model. In addition, as far as the causality of data is concerned, we believe that the data at the subsequent time point still has a certain impact and reference value on the data at the previous time point. The idea of only focusing on the data before the time point may be under consideration.

Taken together, all of the above factors will have a subtle impact on the predictive performance of the TCN model. The TCN model may be more suitable for long-term sequence forecasting. In the follow-up work, the data can be pre-processed more finely, and the model and parameter settings can be further improved and explored. Perhaps by removing some holiday data and improving the TCN model, it can achieve more ideal prediction results.

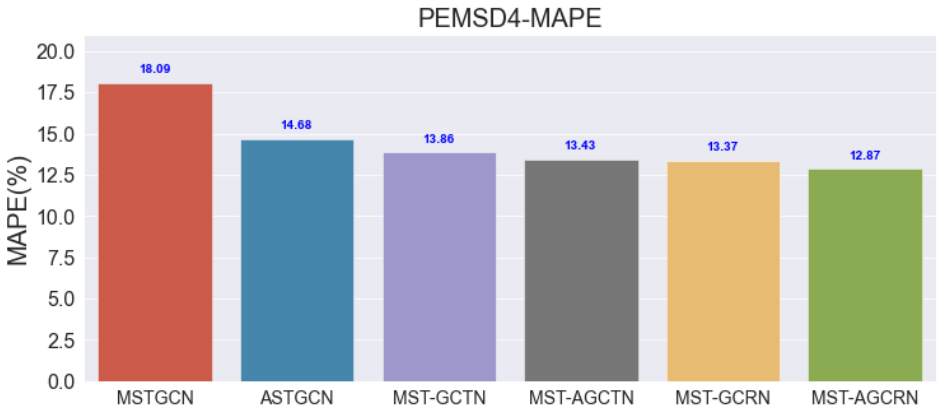
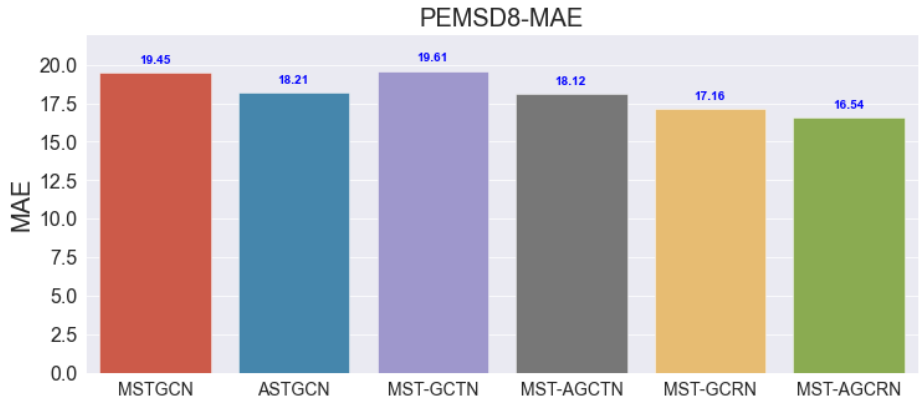


Figure 4.22 The evaluation results of the models on the PeMSD4

Figure 4.22 is the evaluation results of deep learning models on PeMSD4. From the results, we can see the problem raised in the previous paragraph. The two models of GCTN are 1.89 and 1.35 higher in MAE than the two models of GCRN we proposed. In addition, it is 4.2 and 2.7 higher in RMSE and 0.49% and 0.56% higher in MAPE. Moreover, it can also be seen from the figure that the prediction performance of the GCTN model is worse than the performance of MSTGCN and ASTGCN that we have chosen as the comparison baseline. This pattern can also be seen from the evaluation results of the models on PeMSD8 in Figure 4.23.



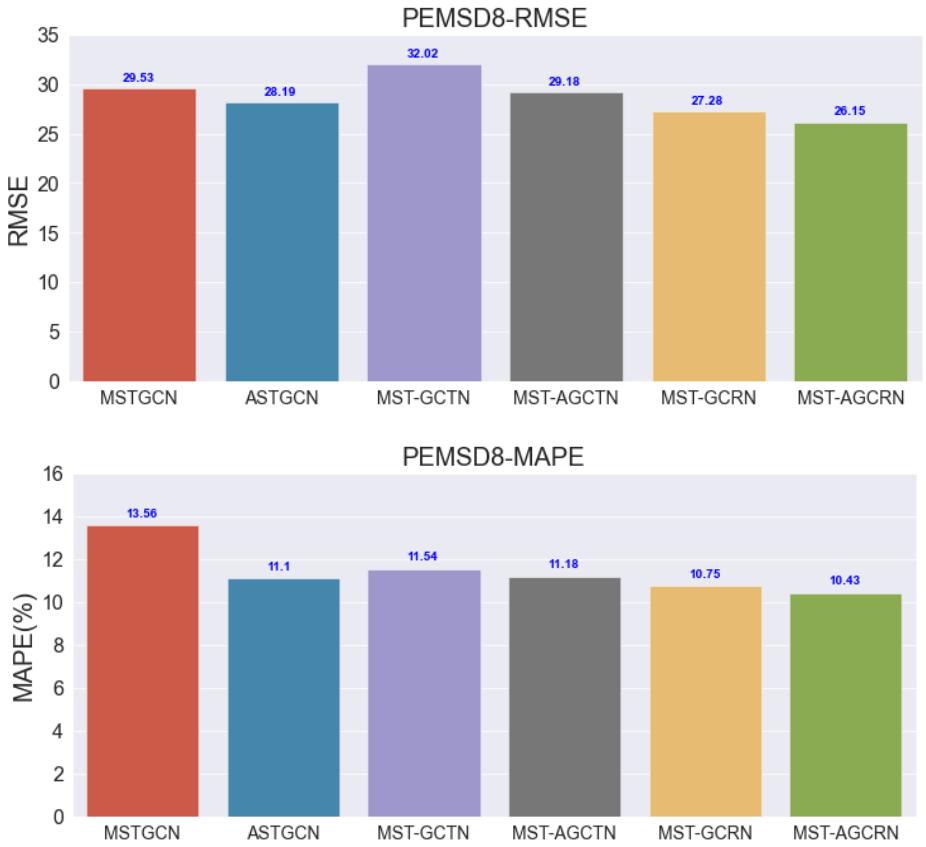


Figure 4.23 The evaluation results of the models on the PeMSD8

On the whole, from the evaluation results of the models. As shown in Figures 4.22 and 4.23. The performance of the MST-GCTN model and MST-AGCTN model is worse than the MST-GCRN model and MST-AGCRN model. Moreover, it is not as effective as previous models based on GCN and CNN (such as MSTGCN, ASTGCN). Therefore, this work will not discuss the MST-GCTN model and the MST-AGCTN model in the following sections.

In summary, As can be seen from case study 1 and case study 2. In our proposed models, the performance of the MST-AGCRN model on the data set is the best, followed by the MST-GCRN model. The addition of the self-attention mechanism is the reason why the MST-AGCRN model is more effective than the MST-GCRN model. The difference in the performance of predicting traffic flow data of different volumes also proves the model's ability to perceive the temporal and spatial features of traffic flow. The different performances of the models can be clearly seen from the comprehensive evaluation results in Figure 4.22 and Figure 4.23. Therefore, this

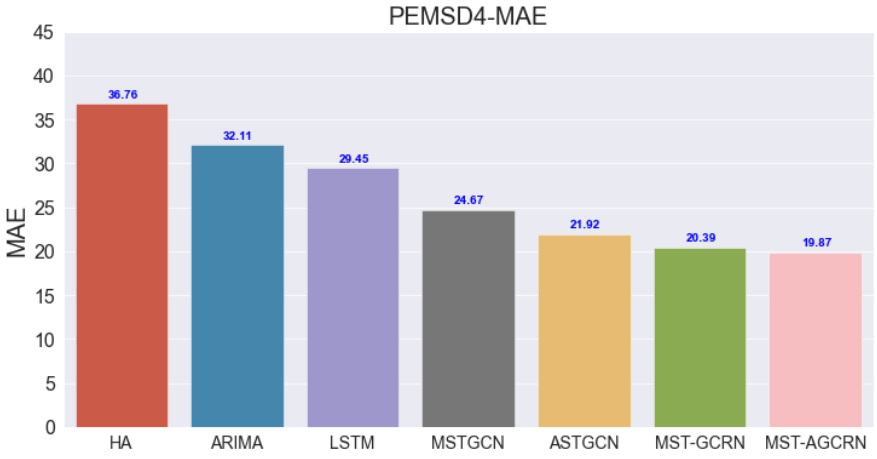
thesis selects the MST-GCRN model and MST-AGCRN model as the final model of this thesis in the following sections. Through them, the overall prediction accuracy of the models is compared and analyzed with the baseline models. Due to time and capacity constraints, the TCN-based models can be left in future research for more in-depth investigation and research.

4.4.4 Case study 3: The accuracy comparison study with some benchmarking models

Through the model analysis and comparison in section 4.4.3, this section mainly uses the MST-GCRN model and the MST-AGCRN model as the final model of this thesis. It compares it with several typical traffic flow prediction models on the PeMSD4 data set and PeMSD8 data set. Based on the three evaluation criteria results, we analyze the performance of different models on two data sets and focus on demonstrating the good performance of the models proposed in this thesis on the two data sets.

1. Overall performance analysis of prediction models

Figure 4.24, Figure 4.25 and Figure 4.26 show the comprehensive prediction results of several different traffic flow prediction models. First, calculate the MAE, RMSE, and MAPE results of each traffic flow prediction model on two different data sets. Then carry out a comparative analysis.



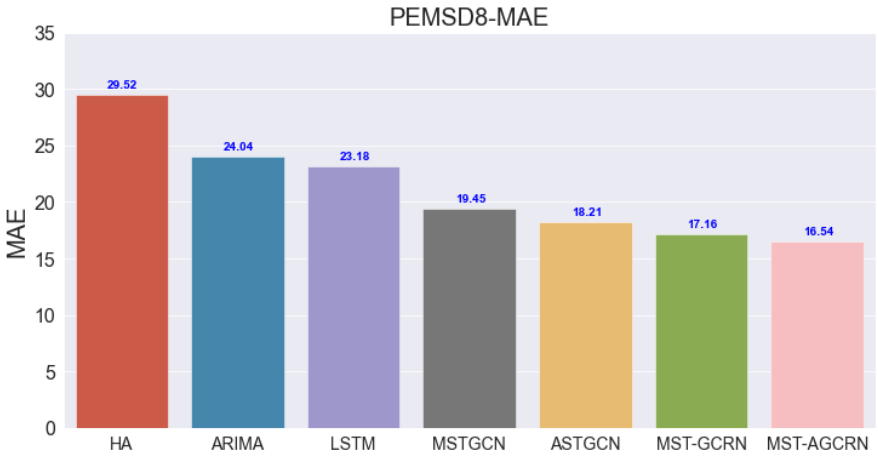


Figure 4.24 The MAE results of the models

It is not difficult to see from the above figures that the MST-GCRN model and MST-AGCRN model proposed by this thesis show good performance in comprehensive prediction performance. On the PeMSD4 data set, the MAE results of our two models are 1.53 and 2.05 respectively, lower than the previous best-performing ASTGCN model. On the PeMSD8 data set, the MAE results of our two models are 1.05 and 1.67 respectively, lower than the previous best-performing ASTGCN model. From the most concise MAE results, we can see that our MST-GCRN model and MST-AGCRN model are more excellent than classical statistical theories and analytical models (HA, ARIMA). They use the powerful feature extraction capabilities of neural networks to learn the nonlinear data structure. Compared with traditional machine learning methods (LSTM), they can extract data features from two dimensions of time and space through multi-layer modules. Through the graph convolutional neural network, the features representation of the node in the spatial and temporal dimensions are obtained, so more powerful features of traffic flow data are extracted. This also brings about an improvement in forecast accuracy.

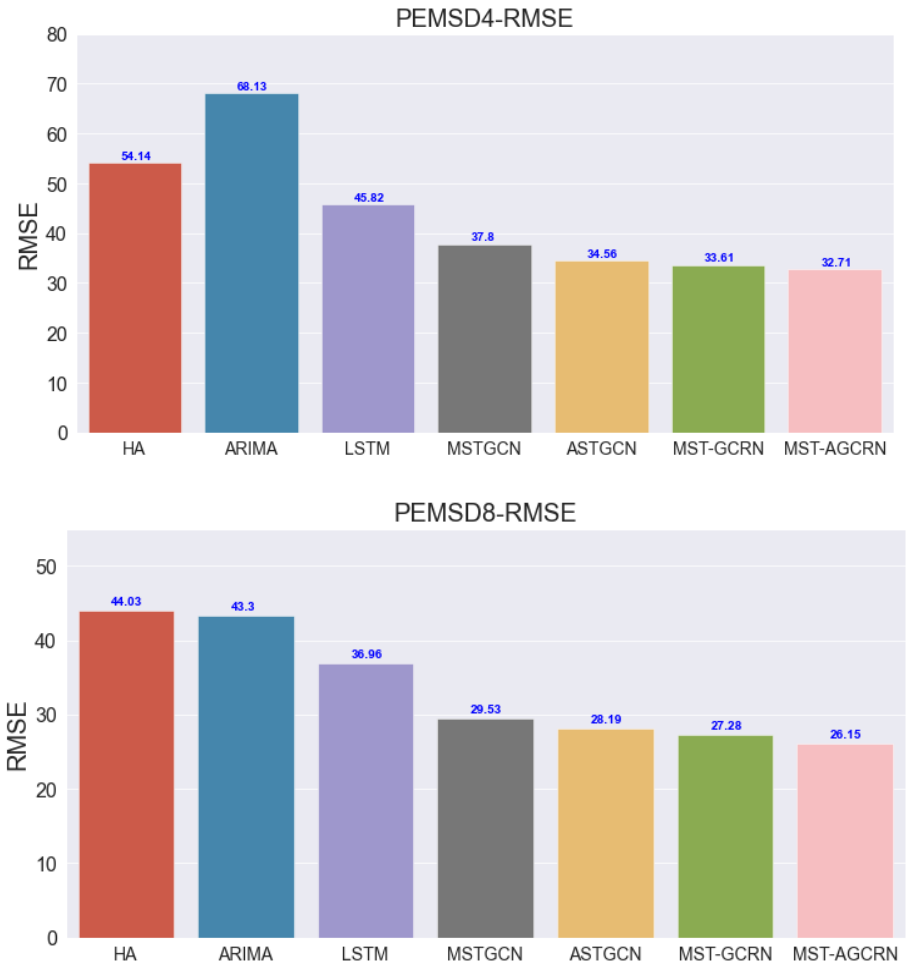


Figure 4.25 The RMSE results of the models

Compared with the ASTGCN model, the RMSE results on PeMSD4 are reduced by 0.95 and 1.85 respectively. Compared with the ASTGCN model, the RMSE results on PeMSD8 are reduced by 0.91 and 2.04, respectively. From the RMSE results, our MST-GCRN model and MST-AGCRN model have also improved their prediction accuracy. Compared with the ASTGCN model with the smallest root mean square error among other models, our model uses a GRU module and a self-attention mechanism instead of a CNN module and a graph attention mechanism. Therefore, the accuracy has been improved to a certain extent under similar circumstances.

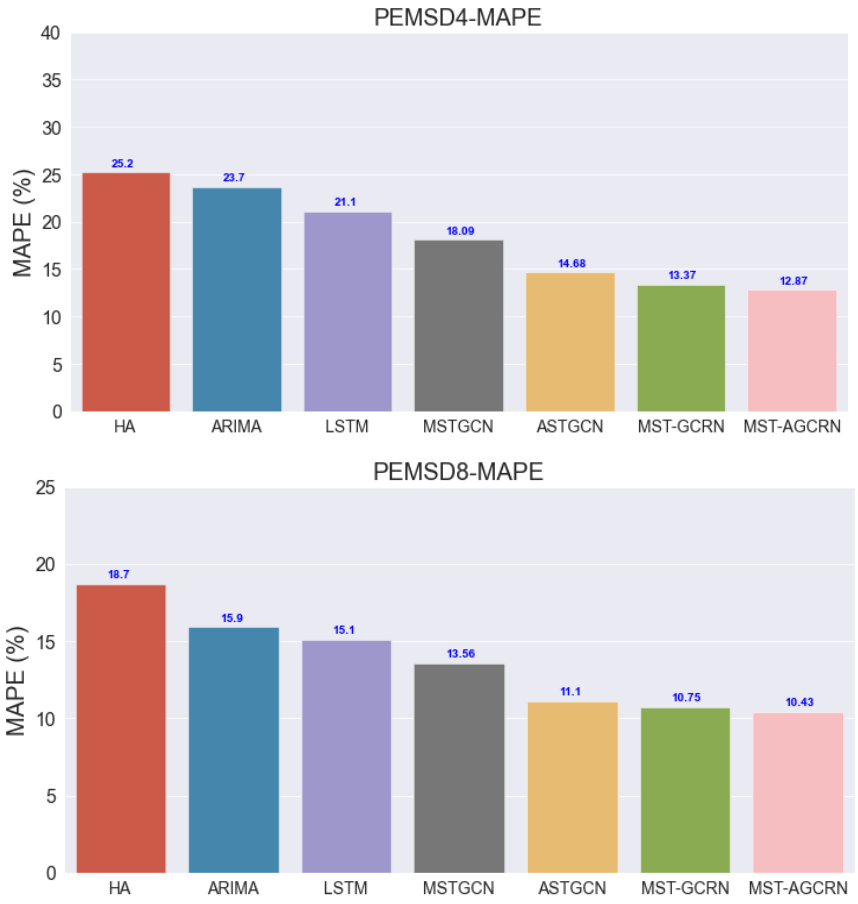


Figure 4.26 The MAPE results of the models

Compared with the ASTGCN model, the MAPE results of the models are respectively reduced by 1.31% and 1.81% on PeMSD4. Compared with the ASTGCN model, the MAPE results on PeMSD8 are reduced by 0.63% and 0.67%, respectively. In general, the prediction model proposed in this work has achieved advanced prediction accuracy on both real data sets.

According to the type of model, among all models, the traffic flow prediction models based on deep learning (LSTM, MSTGCN, ASTGCN, MST-GCRN, MST-AGCRN) outperform classical statistical theories and analytical models (HA, ARIMA). The indexes of the deep learning model in MAE, RMSE, and MAPE are all lower, which shows that the deep learning method is more accurate and deeper in the feature extraction of traffic flow data. They are more powerful in analyzing non-linear data structures. In the deep learning prediction model, the performance of LSTM is weaker than that of the prediction network model based on graph neural networks. This is

because the LSTM model does not take the spatial information of the road network structure into consideration. This also fully shows that it is important to consider the structure of the graph of the road network in the prediction model.

2. Performance changes at different prediction time steps

In addition, this thesis also compares the performance changes of four deep learning models based on graph neural networks (MSTGCN, ASTGCN, MST-GCRN, MST-AGCRN) on two data sets at different prediction time steps. With 5 minutes as the interval, the predicted time step increases from 5 minutes to 1 hour. The simulation results are shown in Figure 4.27, Figure 4.28 and Figure 4.29.

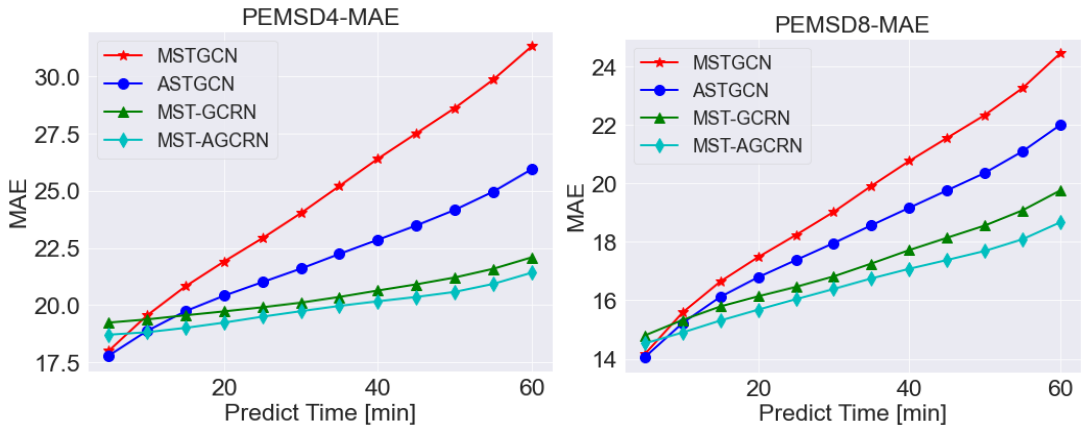


Figure 4.27 The MAE results of the models at different time steps

It can be seen from Figure 4.27 that our models have an outstanding ability to predict longer time steps. They are the most obvious difference in the prediction accuracy of 60-minute predictions than the two deep learning baseline models. As shown in Figures 4.28 and 4.29, this trend is similar in the other two evaluation results.

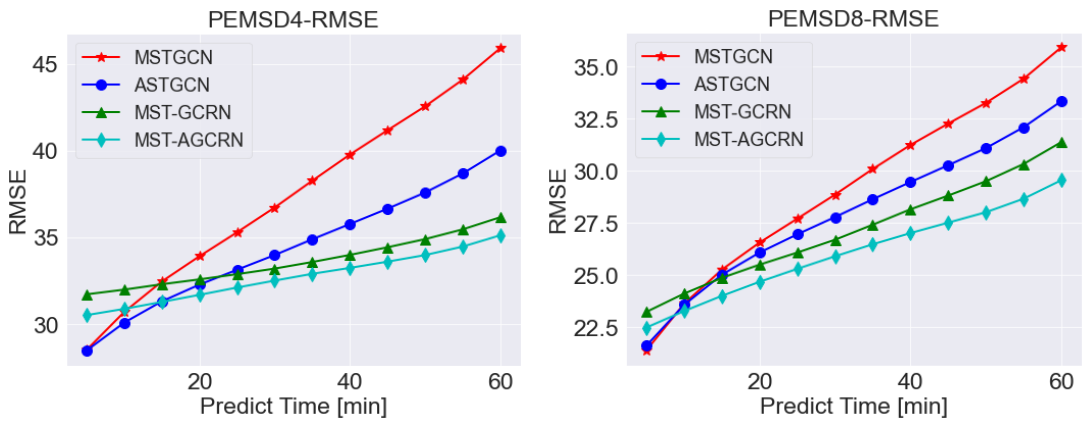


Figure 4.28 The RMSE results of the models at different time steps

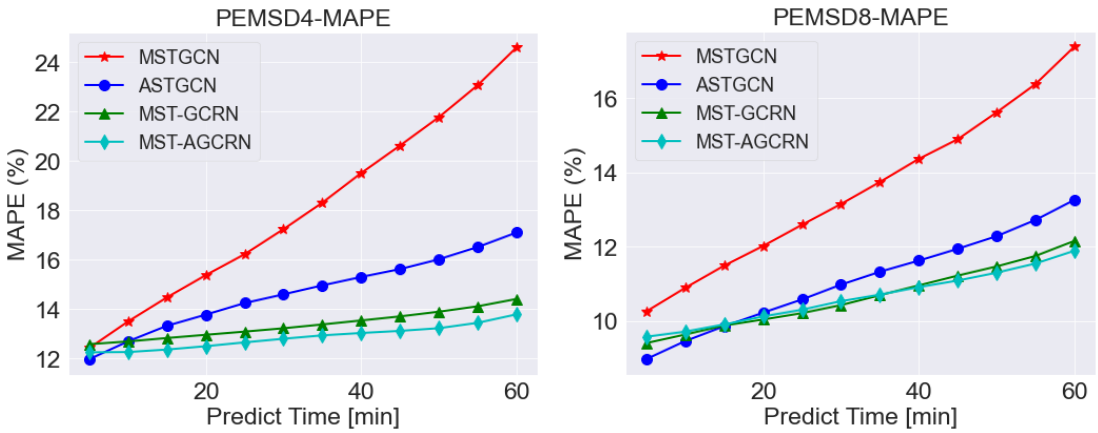


Figure 4.29 The MAPE results of the models at different time steps

As can be seen from the figures, as a whole, with the gradual increase of prediction time, the difficulty of prediction becomes more and more difficult. Their errors generally show an upward trend. The four models showed the same growth trend on the PeMSD4 data set and PeMSD8 data set. Compared with the MSTGCN model, the prediction errors of the three prediction models, ASTGCN, MST-GCRN, and MST-AGCRN, increase relatively slowly. Moreover, the prediction accuracy of the two models used in this study are better than the MSTGCN model and the ASTGCN model at 12 time steps (except for the first 5 minutes). This shows that the MST-GCRN model and MST-AGCRN model can fully mine the spatial-temporal patterns of data, and show more obvious advantages in mid-and long-term predictions. This advantage is due to the use of GRU modules instead of CNN modules. It is because the recurrent neural network has a stronger ability to extract time series information and has a better ability to process historical information.

3. Performance changes at different nodes (stations)

To analyze the prediction accuracy of the MST-GCRN model and MST-AGCRN model more three-dimensionally. Figure 4.30, Figure 4.31 and Figure 4.32 show the RMSE, MAE and MAPE results of the four deep learning models' traffic predictions at each node.

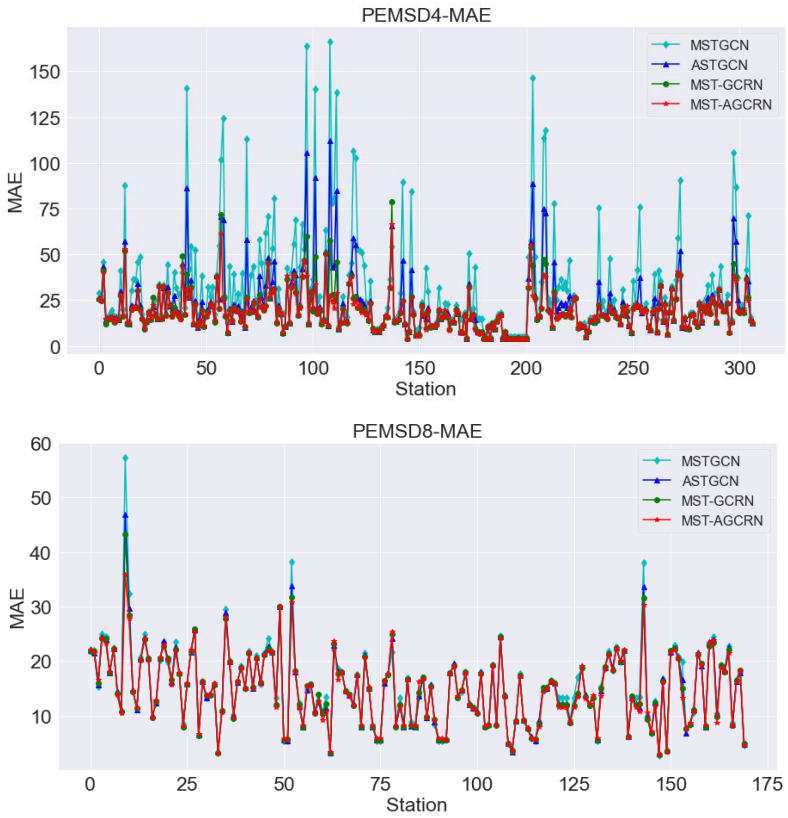
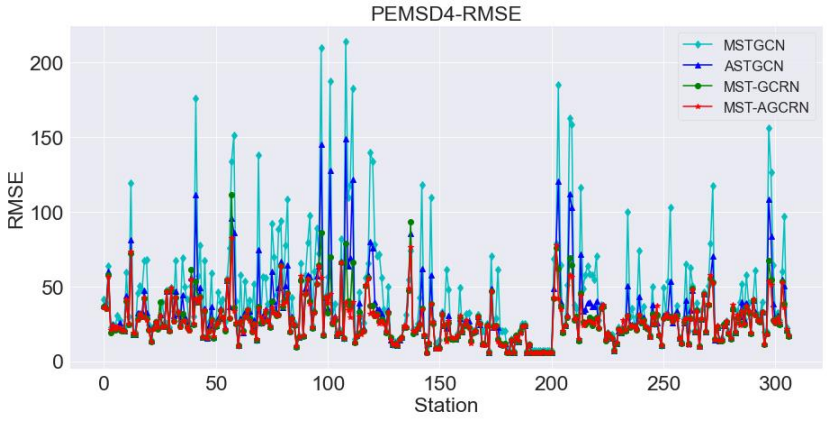


Figure 4.30 The MAE results of the models at each node

It can be seen from Figure 4.30 that the MST-GCRN model and MST-AGCRN model proposed in this thesis can achieve the best prediction accuracy for most stations. For nodes that are more difficult to predict and have greater prediction errors in public data sets, the prediction advantages of the two models of this thesis are more obvious. In addition, the model of this work also shows more prominent and outstanding results on the PeMSD4 data set, which is more complex than the PeMSD8 data set. The trend of such evaluation results is also evident in Figures 4.31 and 4.32.



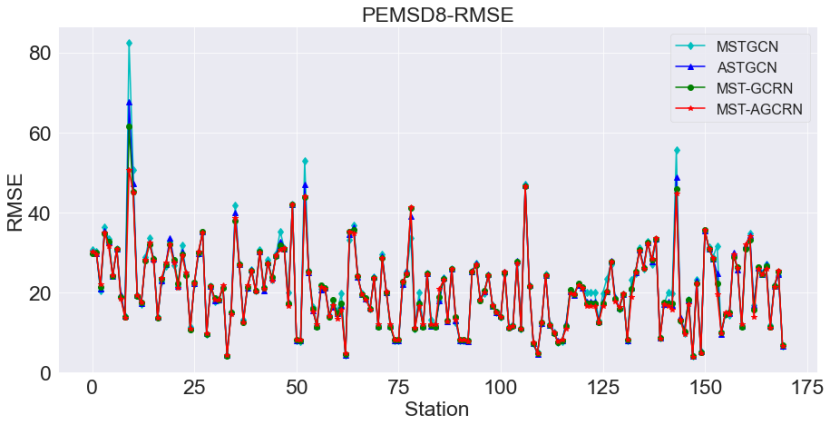


Figure 4.31 The RMSE results of the models at each node

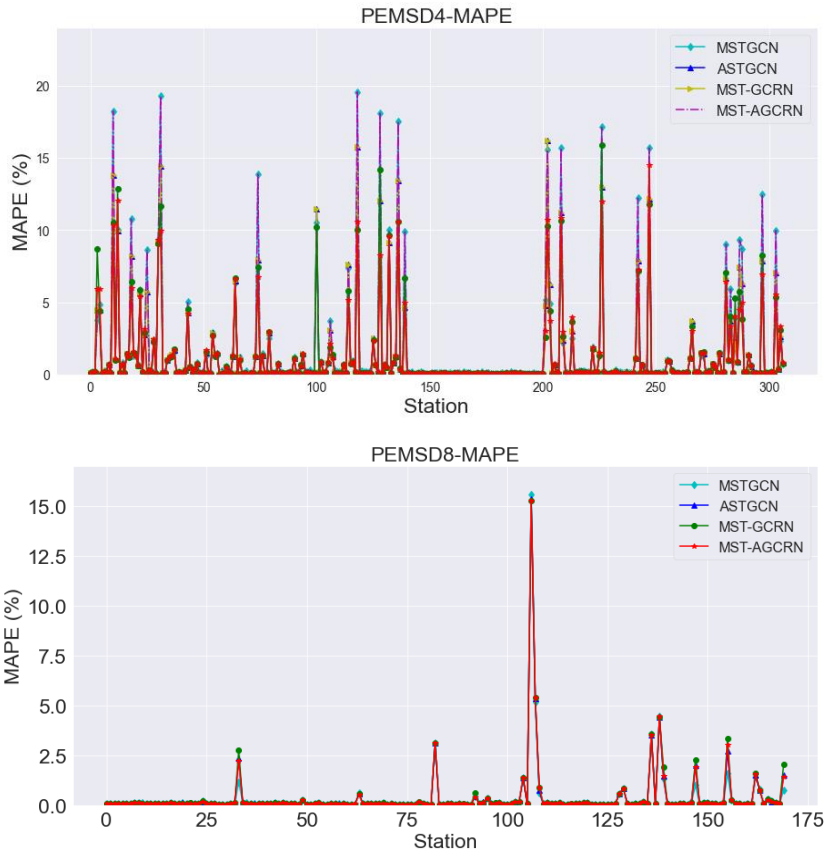


Figure 4.32 The MAPE results of the models at each node

In summary, the MST-GCRN model and MST-AGCRN model proposed by this thesis can effectively capture the relationship between stations and learn rich feature representations, which is beneficial to traffic flow prediction. Therefore, the models can be effectively applied to traffic flow prediction tasks on two real data sets (PeMSD4 and PeMSD8) to achieve ideal prediction results.

4.4.5 Case study 4: Model training performance comparison

In this section, we will discuss the comprehensive capabilities of the MST-(A)GCRN models and the MST-(A)GCTN models through the model's performance in training. The actual training effect of each deep learning model and the algorithm efficiency of the models are discussed through the time performance of the model in the training process. And, through the evolution of the loss function of the models, we will discuss the actual effects of using the residual network, the self-attention mechanism and adding the dropout layer in the Readout phase.

1. Time performance.

Based on the hyperparameter settings, the time performance of the four models in PeMSD4 and PeMSD8 is shown in the following table.

Table 4-4 Time performance (mins) 100 epoch

	PeMSD4	PeMSD8
MST-GCRN	62	34
MST-AGCRN	49	40
MST-GCTN	178	46
MST-AGCTN	179	46

It can be seen from the time performance of model training. Due to the relatively small sample size of the data set, the four models have little difference in the time performance of the PeMSD8 data set. Moreover, due to the low computational complexity of the self-attention mechanism module, there is no obvious difference in the time performance of training between the model with the self-attention mechanism and the model without the self-attention mechanism. On the PeMSD4 data set, the training time of the TCN-based model is longer than that of the GRU-based model. It can be seen that the TCN model is affected by the number of model layers, hyperparameters, and the size of the data set is larger than other models. Therefore, as the amount of data increases, the temporal convolutional network model will increase the training time more significantly than the training time of other models.

2. The loss function of the models

The figure below shows the evolution of the loss (MAE) metric of each model.

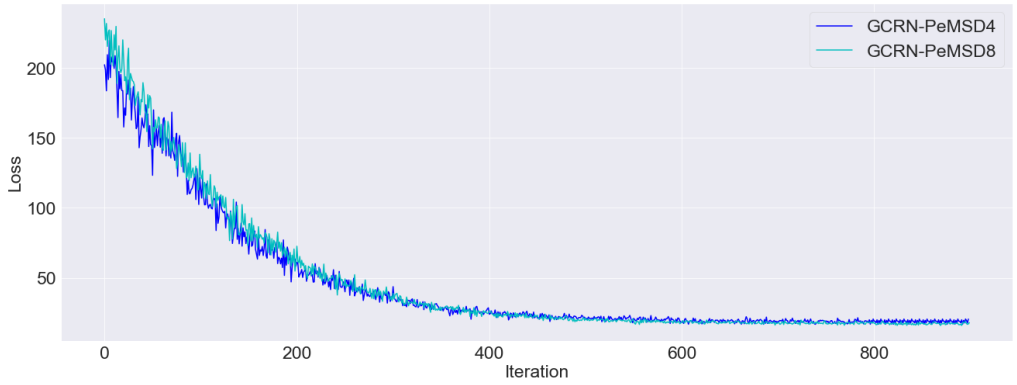


Figure 4.33 The loss metric of the MST-GCRN model on the data set

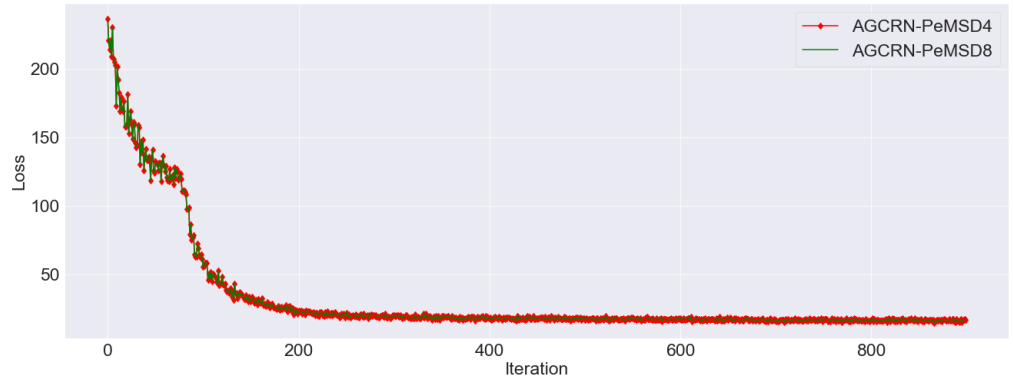


Figure 4.34 The loss metric of the MST-AGCRN model on the data set

As shown from Figure 4.33, the evolution of the loss function of the MST-GCRN model on the PeMSD4 and PeMSD8 data sets. Around the 80th epoch, the loss function of the model tends to stabilize at around 0.019/0.018. Figure 4.34 shows the loss function of the MST-AGCRN model, as for the model that adds the self-attention mechanism. The model's loss function converges faster than the model without the attention mechanism. It can be seen from the figure that the loss function of the MST-AGCRN model on the PeMSD4 and PeMSD8 data sets began to stabilize at about 0.0198/0.0178 at the 60th epoch. After adding the self-attention mechanism, this fully shows that the model has successfully learned the dynamic related features of spatiotemporal data. With the enrichment of feature data, the flexibility of the model and the accuracy of prediction are improved.

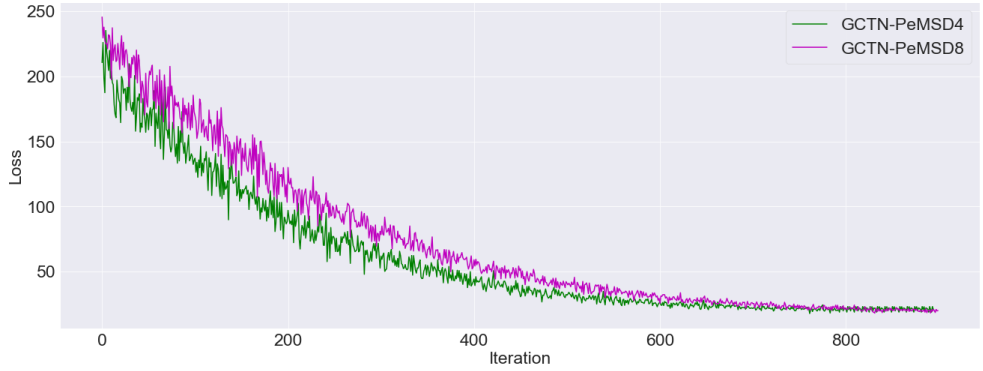


Figure 4.35 The loss metric of the MST-GCTN model on the data set

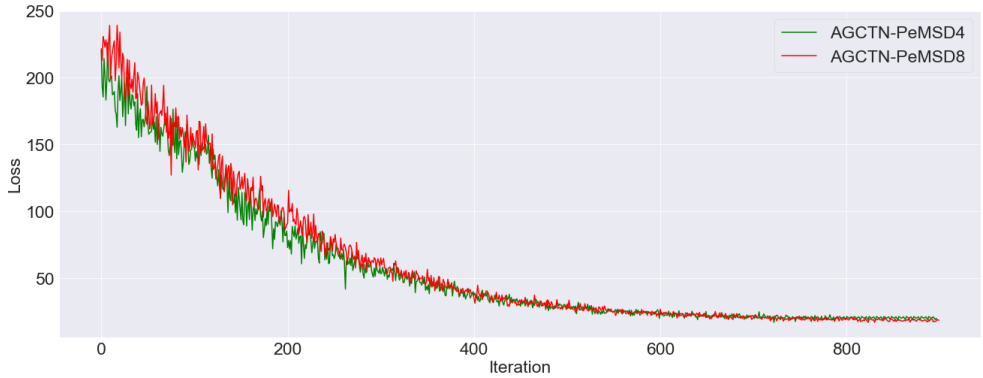


Figure 4.36 The loss metric of the MST-AGCTN model on the data set

As shown from Figure 4.35, the evolution process of the loss function of the MST-GCTN model on the PeMSD4 and PeMSD8 data sets. Around the 100th epoch, the loss function of the model tends to stabilize at about 0.0229/0.0208. As shown from Figure 4.36, the evolution process of the loss function of the MST-AGCTN model on the PeMSD4 and PeMSD8 data sets. For models that add a self-attention mechanism. The model's loss function converges faster than the model without the attention mechanism. The loss function of the MST-AGCTN model on the PeMSD4 and PeMSD8 data sets began to stabilize at about 0.0237/0.0227 at the 80th epoch.

In general, from the evolution curve of the model loss function. The models we proposed can effectively converge in the final stage. This also proves that the machine learning method we used solves the two problems raised in the Readout phase of the previous chapter. In addition, from the overall performance and efficiency, the MST-AGCRN model with the self-attention mechanism is the best. It can converge most effectively during training. Moreover, as can be seen from the previous sections, its prediction accuracy is also the best.

4.5 Summary

This chapter conducts simulations on real traffic flow data sets based on the proposed four models and the selected baseline models. Simulations show that the overall prediction effect of the MST-AGCRN model is the best of them. Simulations have verified from three case studies that the model has advantages in extracting spatiotemporal features and spatial correlation. Moreover, the model based on graph neural networks also has good interpretability. In case study 1, the models' predicted values at different nodes were compared with the true values. The Simulation results show the advantages of the MST-AGCRN model in the extraction of spatial-temporal features, especially in the nodes with large traffic flow. However, the two models based on temporal convolutional networks are not as ideal as those in the modeling chapter due to the range of receptive fields, model parameter settings and computational complexity. In case study 2, the comparison with the baseline also fully demonstrated the outstanding capabilities of MST-AGCRN. Furthermore, It also shows the shortcomings of models based on temporal convolutional networks. In case study 3, the comprehensive training performance of the models is discussed. Overall, MST-AGCRN is still the best performer among them.

Chapter 5

Conclusion and Future Work

The purpose of this thesis was to provide an in-depth understanding of the fundamentals of traffic flow prediction problems through graph neural networks based approach in the field of ITS. Using the characteristics of graph neural networks to capture the spatial and temporal dependence of road structure, enhance the learnability and universal applicability of the prediction models, so as to improve the prediction accuracy of the prediction models are the focus of this thesis. By using a neural network model based on graph neural network and self-attention mechanism, we have contributed new knowledge and solutions to the traffic flow prediction problem in ITS.

5.1 Conclusion

Through the analysis of different types of prediction methods, some research gaps such as low predictive credibility, insufficient general applicability and insufficient feature capture ability have been identified. In this thesis, we propose two types of traffic flow prediction models based on graph neural networks. These two types of models are mainly used to solve the gaps in previous research and improve the model's prediction accuracy. The simulation studies have validated that the prediction accuracy of the models proposed in this thesis is better than other existing baseline methods.

First, we improved the credibility of the prediction. Some previous models and methods lacked the extraction of periodic characteristics of traffic flow data, which led to the inability to learn more powerful traffic flow features reasonably. Based on this problem, we divided the period dependence of the traffic flow data into three types of time components. Furthermore, we carried out a Pearson correlation analysis

on the real traffic flow data. Through this work, the periodic correlation degree of traffic flow data is quantitatively analyzed instead of dividing the periodic components based on experience. On this basis, this work also adopted reasonable data dimensionality reduction strategies to reduce the complexity of input data and the complexity of model calculations. After the above methods, the model's ability to extract traffic flow feature data is improved in the data input stage.

Second, based on the graph neural networks, our models use GRU and TCN respectively, to capture data features in the temporal dimension. The existing traffic flow prediction methods modeled from the temporal dimension do not fully consider the correlation between traffic flow sequences in the spatial dimension, which makes the general applicability of the model insufficient. In addition, some combined deep neural network models ignore the characteristics of the traffic road network graph structure and cannot express the high-order correlation between different nodes. Based on the research of existing traffic flow prediction methods and models in the field of traffic flow prediction, this work proposes spatiotemporal traffic flow prediction modeling methods based on graph neural networks. Through the advantages of the GRU module in MST-GCRN and the TCN module in MST-GCTN in sequence data processing, the interpretability and accuracy of the traffic flow prediction models in temporal dimension are improved. Based on using the GCN module to model the spatial dimension, MST-GCRN and MST-GCTN also fully consider the correlation of traffic flow data in the temporal dimension. Therefore, the universal applicability and prediction accuracy of the models are enhanced at the same time.

Third, this thesis uses the self-attention mechanism of temporal and spatial separation to capture the temporal and spatial dynamics of traffic flow data at a more detailed level. Although the latest traffic flow prediction application models have improved feature capture and scalability, they are often insufficient in capturing the dynamic correlation between nodes. By adding a two-dimensional spatial-temporal self-attention module to the models, our proposed MST-AGCRN and MST-AGCTN

models can allocate node weights more reasonably and effectively in the machine learning process. This enables the models to more accurately capture the dynamic spatial-temporal correlations on the traffic network.

5.2 Future Work

Considering the limitations of the research time of this thesis and the rapid development trend in the field of machine learning, this research has also realized some areas that need improvement and further research.

First of all, the complexity of the data set can be further improved. The data sets used in this simulation are two highway traffic data sets (PeMSD4 and PeMSD8). These data sets have two characteristics. One is the lower difficulty of collection. Compared with complex road conditions, the loop detector used in traffic flow data collection is very easy to plan and deploy on highways. The second is the low data complexity. The collected data is relatively tidy and easy to use because highways generally do not have interference factors such as intersections, traffic lights, and pedestrians. Although these two features improve data quality, reduce the difficulty of data processing, and improve the accuracy of model prediction, from another perspective, they also reduce the general applicability of the model. Lower data complexity means lower data features, which makes the model unable to be applied to the traffic flow prediction of urban roads and crossroads.

Furthermore, from a modeling perspective, the universal applicability of the model can be further strengthened. It can be seen from the simulation that the performance of the models in periods and nodes with a large amount of data is worse than under normal circumstances, which is a common shortcoming of traffic flow prediction models at this stage. With the advancement of neural network models, this problem needs to be noticed and resolved in future work. It can be foreseen that with the emergence of more excellent neural network models and data collection methods in the future, increasing the complexity of the data set and designing better model algorithms may be able to bring better universality and wider application scope to the

model. In addition, the application range of the model can be further expanded. The traffic flow prediction model can also be applied to people flow prediction and other similar flow prediction fields.

More specifically, the performance and application of the temporal convolution model are worthy of further exploration. In our Simulations, the performance of the temporal convolution model is far from as good as expected. The reason is more complicated, and the operation mechanism for this has not been completely clarified in this work. Therefore, clarifying the mechanism of TCN and improving the ability of the TCN model is worthy of further research in the future.

The prediction accuracy of the model can be further improved by increasing the dimensionality of the data. With the improvement of machine learning capabilities, some recent studies have proposed to improve the dimensionality of input data further. For example, when predicting traffic flow, real-time weather, temperature, human factors and other more multi-dimensional environmental factors are taken into account in the input of the model. Although most of these studies stay at the theoretical stage, it is expected that these methods should improve the model's prediction accuracy, thus theses directions are worthy of further research.

Reference

[1] Zhang, S., et al., *Fine-grained vehicle emission management using intelligent transportation system data.* Environmental Pollution, 2018. **241**: p. 1027-1037.

[2] Yang, H.-F., T.S. Dillon, and Y.-P.P. Chen, *Optimized structure of the traffic flow forecasting model with a deep learning approach.* IEEE transactions on neural networks and learning systems, 2016. **28**(10): p. 2371-2381.

[3] Lv, Y., et al., *Traffic flow prediction with big data: a deep learning approach.* IEEE Transactions on Intelligent Transportation Systems, 2014. **16**(2): p. 865-873.

[4] Zhang, J., Y. Zheng, and D. Qi. *Deep spatio-temporal residual networks for citywide crowd flows prediction*. in *Thirty-first AAAI conference on artificial intelligence*. 2017.

[5] Yin, X., et al., *A comprehensive survey on traffic prediction.* arXiv preprint arXiv:2004.08555, 2020.

[6] Scarselli, F., et al., *The graph neural network model.* IEEE transactions on neural networks, 2008. **20**(1): p. 61-80.

[7] Transport, M.o., (2019). *Annual fleet statistics*, M.o. transport, Editor. 2020, Ministry of transport: transport.govt.nz.

[8] Association, N.Z.a. (2019, July). *Auckland Congestion Report 2018*. 2019. New Zealand automobile association. https://www.aa.co.nz/assets/Congestion-Monitoring-Collateral/AA-Auckland-Congestion-Report-2018-FINAL.pdf

[9] Umare, P.R., et al. *Smart Solution for Traffic Control*. in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*. 2019. IEEE.

[10] Anderson, D. and H. Mohring, *Congestion Costs and Congestion Pricing for the Twin Cities.* 1996.

[11] Falcocchio, J.C. and H.S. Levinson, *The costs and other consequences of traffic congestion*, in *Road Traffic Congestion: A Concise Guide*. 2015, Springer. p. 159-182.

[12] Guo, Y., Z. Tang, and J. Guo, *Could a smart city ameliorate urban traffic congestion? A quasi-natural experiment based on a smart city pilot program in China.* Sustainability, 2020. **12**(6): p. 2291.

[13] Haydari, A. and Y. Yilmaz, *Deep reinforcement learning for intelligent transportation systems: A survey.* IEEE Transactions on Intelligent Transportation Systems, 2020.

[14] Zhang, J., et al., *Data-driven intelligent transportation systems: A survey.* IEEE Transactions on Intelligent Transportation Systems, 2011. **12**(4): p. 1624-1639.

[15] Zanella, A., et al., *Internet of things for smart cities.* IEEE Internet of Things journal, 2014. **1**(1): p. 22-32.

[16] Habtemichael, F.G. and M. Cetin, *Short-term traffic flow rate forecasting based on identifying similar traffic patterns.* Transportation research Part C: emerging technologies, 2016. **66**: p. 61-78.

[17] Jiang, X., L. Zhang, and X.M. Chen, *Short-term forecasting of high-speed rail demand: A hybrid approach combining ensemble empirical mode decomposition and gray support vector machine with real-world applications in China.* Transportation Research Part C: Emerging Technologies, 2014. **44**: p. 110-127.

[18] Guo, S., et al. *Attention based spatial-temporal graph convolutional networks for traffic flow*

forecasting. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019.

[19] Liu, J. and W. Guan, *A summary of traffic flow forecasting methods [J]*. Journal of highway and transportation research and development, 2004. **3**: p. 82-85.

[20] García-Jurado, I., et al., *Predicting using box—jenkins, nonparametric, and bootstrap techniques.* Technometrics, 1995. **37**(3): p. 303-310.

[21] Antoniou, C., H.N. Koutsopoulos, and G. Yannis. *An efficient non-linear Kalman filtering algorithm using simultaneous perturbation and applications in traffic estimation and prediction*. in *2007 IEEE Intelligent Transportation Systems Conference*. 2007. IEEE.

[22] Okutani, I. and Y.J. Stephanedes, *Dynamic prediction of traffic volume through Kalman filtering theory.* Transportation Research Part B: Methodological, 1984. **18**(1): p. 1-11.

[23] Holden, K., *Vector auto regression modeling and forecasting.* Journal of Forecasting, 1995. **14**(3): p. 159-166.

[24] Hamilton, J.D., *Time series analysis*. 2020: Princeton university press.

[25] Chen, C., et al. *Short-time traffic flow prediction with ARIMA-GARCH model*. in *2011 IEEE Intelligent Vehicles Symposium (IV)*. 2011. IEEE.

[26] Williams, B.M. and L.A. Hoel, *Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results.* Journal of transportation engineering, 2003. **129**(6): p. 664-672.

[27] Zheng, Z. and D. Su, *Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm.* Transportation Research Part C: Emerging Technologies, 2014. **43**: p. 143-157.

[28] Wei, Y. and M.-C. Chen, *Forecasting the short-term metro passenger flow with empirical mode decomposition and neural networks.* Transportation Research Part C: Emerging Technologies, 2012. **21**(1): p. 148-162.

[29] Sun, Y., B. Leng, and W. Guan, *A novel wavelet-SVM short-time passenger flow prediction in Beijing subway system.* Neurocomputing, 2015. **166**: p. 109-121.

[30] Liu, Y., Z. Liu, and R. Jia, *DeepPF: A deep learning based architecture for metro passenger flow prediction.* Transportation Research Part C: Emerging Technologies, 2019. **101**: p. 18-34.

[31] Yang, Z., et al., *Research on short-term traffic flow prediction method based on similarity search of time series.* Mathematical Problems in Engineering, 2014. **2014**.

[32] Han, C., S. Song, and C.-h. Wang, *Real-time adaptive prediction of short-term traffic flow based on ARIMA model [J]*. Journal of System Simulation, 2004. **16**(7): p. 1530-1532.

[33] Liu, Y., et al. *Short-term traffic flow prediction with Conv-LSTM*. in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*. 2017. IEEE.

[34] Koesdwiady, A., R. Soua, and F. Karray, *Improving traffic flow prediction with weather information in connected cars: A deep learning approach.* IEEE Transactions on Vehicular Technology, 2016. **65**(12): p. 9508-9517.

[35] Li, J. and J. Wang, *Short term traffic flow prediction based on deep learning*, in *CICTP 2019*. 2017. p. 2457-2469.

[36] Wu, Y., et al., *A hybrid deep learning based traffic flow prediction method and its understanding.* Transportation Research Part C: Emerging Technologies, 2018. **90**: p. 166-180.

[37] Jin, W., et al. *Spatio-temporal recurrent convolutional networks for citywide short-term crowd flows prediction*. in *Proceedings of the 2nd International Conference on Compute and Data Analysis*. 2018.

[38] Yao, H., et al. *Deep multi-view spatial-temporal network for taxi demand prediction*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018.

[39] Yao, H., et al., *Modeling spatial-temporal dynamics for traffic prediction.* arXiv preprint arXiv:1803.01254, 2018.

[40] Seo, Y., et al. *Structured sequence modeling with graph convolutional recurrent networks*. in *International Conference on Neural Information Processing*. 2018. Springer.

[41] Zhao, L., et al., *T-gcn: A temporal graph convolutional network for traffic prediction.* IEEE Transactions on Intelligent Transportation Systems, 2019. **21**(9): p. 3848-3858.

[42] Li, Y., et al., *Diffusion convolutional recurrent neural network: Data-driven traffic forecasting.* arXiv preprint arXiv:1707.01926, 2017.

[43] Yu, B., H. Yin, and Z. Zhu, *Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting.* arXiv preprint arXiv:1709.04875, 2017.

[44] Dauphin, Y.N., et al. *Language modeling with gated convolutional networks*. in *International conference on machine learning*. 2017. PMLR.

[45] Diao, Z., et al. *Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting*. in *Proceedings of the AAAI conference on artificial intelligence*. 2019.

[46] Chai, D., L. Wang, and Q. Yang. *Bike flow prediction with multi-graph convolutional networks*. in *Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems*. 2018.

[47] Geng, X., et al. *Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting*. in *Proceedings of the AAAI conference on artificial intelligence*. 2019.

[48] Keogh, E. and S. Kasetty, *On the need for time series data mining benchmarks: a survey and empirical demonstration.* Data Mining and knowledge discovery, 2003. **7**(4): p. 349-371.

[49] Jia, X., *Characteristic analysis and demand forecast of residents' travel demand based on data of online car arrangement.* Transport Eng, 2018. **18**(05): p. 39-45.

[50] Zheng, Y. and X. Xie, *Learning travel recommendations from user-generated GPS traces.* ACM Transactions on Intelligent Systems and Technology (TIST), 2011. **2**(1): p. 1-29.

[51] Lu, Y.-S., et al., *On successive point-of-interest recommendation.* World Wide Web, 2019. **22**(3): p. 1151-1173.

[52] Feiyan, Z., J. Linpeng, and D. Jun, *Summary of Convolutional Neural Network Research [J]*. Chinese Journal of Computers, 2017. **6**.

[53] LeCun, Y. and Y. Bengio, *Convolutional networks for images, speech, and time series.* The handbook of brain theory and neural networks, 1995. **3361**(10): p. 1995.

[54] Sun, Y., X. Wang, and X. Tang. *Deep learning face representation from predicting 10,000 classes*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[55] Albawi, S., T.A. Mohammed, and S. Al-Zawi. *Understanding of a convolutional neural network.* in *2017 International Conference on Engineering and Technology (ICET)*. 2017. Ieee.

[56] Akhtar, N. and U. Ragavendran, *Interpretation of intelligence in CNN-pooling processes: a methodological survey.* Neural computing and applications, 2020. **32**(3): p. 879-898.

[57] Aghdam, H.H. and E.J. Heravi, *Guide to convolutional neural networks.* New York, NY: Springer, 2017. **10**(978-973): p. 51.

[58] Wu, X.Y., *A hand gesture recognition algorithm based on DC-CNN.* Multimedia Tools and Applications, 2020. **79**(13): p. 9193-9205.

[59] Garland, J. and D. Gregg, *Low complexity multiply accumulate unit for weight-sharing convolutional neural networks.* IEEE Computer Architecture Letters, 2017. **16**(2): p. 132-135.

[60] Davis, N., G. Raina, and K. Jagannathan, *Grids versus graphs: Partitioning space for improved taxi demand-supply forecasts.* IEEE Transactions on Intelligent Transportation Systems, 2020.

[61] Liu, L., et al., *Contextualized spatial–temporal network for taxi origin-destination demand prediction.* IEEE Transactions on Intelligent Transportation Systems, 2019. **20**(10): p. 3875-3887.

[62] Wang, D., et al. *When will you arrive? estimating travel time based on deep neural networks*. in *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[63] Guo, S., et al., *Deep spatial–temporal 3D convolutional neural networks for traffic data forecasting.* IEEE Transactions on Intelligent Transportation Systems, 2019. **20**(10): p. 3913-3926.

[64] Lin, Z., et al. *Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis*. in *Proceedings of the AAAI conference on artificial intelligence*. 2019.

[65] Jiang, R., et al. *Deepurbanevent: A system for predicting citywide crowd dynamics at big events*. in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019.

[66] Lee, D., et al., *Forecasting taxi demands with fully convolutional networks and temporal guided embedding.* 2018.

[67] Mikolov, T., et al. *Recurrent neural network based language model*. in *Eleventh annual conference of the international speech communication association*. 2010.

[68] Hughes, T. and K. Mierle. *Recurrent neural networks for voice activity detection*. in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013. IEEE.

[69] Liu, P., X. Qiu, and X. Huang, *Recurrent neural network for text classification with multi-task learning.* arXiv preprint arXiv:1605.05101, 2016.

[70] Mandic, D. and J. Chambers, *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. 2001: Wiley.

[71] Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.

[72] Cheng, J., L. Dong, and M. Lapata, *Long short-term memory-networks for machine reading.* arXiv preprint arXiv:1601.06733, 2016.

[73] Malhotra, P., et al. *Long short term memory networks for anomaly detection in time series*. in *Proceedings*. 2015.

[74] Xu, Y., et al. *Classifying relations via long short term memory networks along shortest dependency paths*. in *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015.

[75] Vlachas, P.R., et al., *Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks.* Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 2018. **474**(2213): p. 20170844.

[76] Van Houdt, G., C. Mosquera, and G. Nápoles, *A review on the long short-term memory model.* Artif. Intell. Rev., 2020. **53**(8): p. 5929-5955.

[77] Chung, J., et al., *Empirical evaluation of gated recurrent neural networks on sequence*

modeling. arXiv preprint arXiv:1412.3555, 2014.

[78] Bai, S., J.Z. Kolter, and V. Koltun, *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.* arXiv preprint arXiv:1803.01271, 2018.

[79] LeCun, Y., et al., *Backpropagation applied to handwritten zip code recognition.* Neural computation, 1989. **1**(4): p. 541-551.

[80] Chen, C., et al. *Gated residual recurrent graph neural networks for traffic prediction*. in *Proceedings of the AAAI conference on artificial intelligence*. 2019.

[81] Cui, Z., et al., *Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting.* IEEE Transactions on Intelligent Transportation Systems, 2019. **21**(11): p. 4883-4894.

[82] Guo, K., et al., *Optimized graph convolution recurrent neural network for traffic prediction.* IEEE Transactions on Intelligent Transportation Systems, 2020. **22**(2): p. 1138-1149.

[83] Liao, B., et al. *Deep sequence learning with auxiliary information for traffic prediction*. in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018.

[84] Song, J., et al., *A survey of remote sensing image classification based on CNNs.* Big earth data, 2019. **3**(3): p. 232-254.

[85] Gidaris, S. and N. Komodakis. *Object detection via a multi-region and semantic segmentation-aware cnn model*. in *Proceedings of the IEEE international conference on computer vision*. 2015.

[86] Zhang, J. and C. Zong, *Deep Neural Networks in Machine Translation: An Overview.* IEEE Intell. Syst., 2015. **30**(5): p. 16-25.

[87] Gori, M., G. Monfardini, and F. Scarselli. *A new model for learning in graph domains*. in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* 2005. IEEE.

[88] Bollobás, B., *Modern graph theory*. Vol. 184. 2013: Springer Science & Business Media.

[89] Kerzner, E., et al. *Graffinity: Visualizing connectivity in large graphs*. in *Computer Graphics Forum*. 2017. Wiley Online Library.

[90] Merris, R., *Laplacian matrices of graphs: a survey.* Linear algebra and its applications, 1994. **197**: p. 143-176.

[91] Kipf, T.N. and M. Welling, *Semi-supervised classification with graph convolutional networks.* arXiv preprint arXiv:1609.02907, 2016.

[92] Lai, Y., et al., *Fine-grained emotion classification of Chinese microblogs based on graph convolution networks.* World Wide Web, 2020. **23**(5): p. 2771-2787.

[93] Li, P.-N., et al. *Sequence-guided protein structure determination using graph convolutional and recurrent networks*. in *2020 IEEE 20th international conference on bioinformatics and bioengineering (BIBE)*. 2020. IEEE.

[94] XIANG, M., et al., *Software-defined power communication network routing control strategy based on graph convolution network.* Journal of Electronics and Information, 2021. **43**(2): p. 388-395.

[95] Niepert, M., M. Ahmed, and K. Kutzkov. *Learning convolutional neural networks for graphs*. in *International conference on machine learning*. 2016. PMLR.

[96] Li, C., et al., *Action-attending graphic neural network.* IEEE Transactions on Image Processing, 2018. **27**(7): p. 3657-3670.

[97] Cui, Y., et al., *Attention-over-attention neural networks for reading comprehension.* arXiv preprint arXiv:1607.04423, 2016.

[98] Bruna, J., et al., *Spectral networks and locally connected networks on graphs.* arXiv preprint arXiv:1312.6203, 2013.

[99] Defferrard, M., X. Bresson, and P. Vandergheynst, *Convolutional neural networks on graphs with fast localized spectral filtering.* Advances in neural information processing systems, 2016. **29**: p. 3844-3852.

[100] Wu, Z., et al., *Graph wavenet for deep spatial-temporal graph modeling.* arXiv preprint arXiv:1906.00121, 2019.

[101] Song, C., et al. *Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.

[102] He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[103] Jia, Y., et al. *Caffe: Convolutional architecture for fast feature embedding*. in *Proceedings of the 22nd ACM international conference on Multimedia*. 2014.

[104] Mnih, V., N. Heess, and A. Graves. *Recurrent models of visual attention*. in *Advances in neural information processing systems*. 2014.

[105] Ba, J., V. Mnih, and K. Kavukcuoglu, *Multiple object recognition with visual attention.* arXiv preprint arXiv:1412.7755, 2014.

[106] Bahdanau, D., K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate.* arXiv preprint arXiv:1409.0473, 2014.

[107] Vaswani, A., et al. *Attention is all you need*. in *Advances in neural information processing systems*. 2017.

[108] Hu, D. *An introductory survey on attention mechanisms in NLP problems*. in *Proceedings of SAI Intelligent Systems Conference*. 2019. Springer.

[109] Xu, K., et al. *Show, attend and tell: Neural image caption generation with visual attention*. in *International conference on machine learning*. 2015. PMLR.

[110] Lu, J., et al., *Hierarchical question-image co-attention for visual question answering.* Advances in neural information processing systems, 2016. **29**: p. 289-297.

[111] Chan, W., et al. *Listen, attend and spell: A neural network for large vocabulary conversational speech recognition*. in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016. IEEE.

[112] Luong, M.-T., H. Pham, and C.D. Manning, *Effective approaches to attention-based neural machine translation.* arXiv preprint arXiv:1508.04025, 2015.

[113] Veličković, P., et al., *Graph attention networks.* arXiv preprint arXiv:1710.10903, 2017.

[114] Gehring, J., et al., *A convolutional encoder model for neural machine translation.* arXiv preprint arXiv:1611.02344, 2016.

[115] Lin, Z., et al., *A structured self-attentive sentence embedding.* arXiv preprint arXiv:1703.03130, 2017.

[116] Chen, W., et al. *Multi-range attentive bicomponent graph convolutional network for traffic forecasting*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.

[117] Geng, X., et al., *CGT: Clustered Graph Transformer for Urban Spatio-temporal Prediction.* 2019.

[118] Li, Y. and J.M. Moura, *Forecaster: A graph transformer for forecasting spatial and time-dependent data.* arXiv preprint arXiv:1909.04019, 2019.

[119] Li, Y., et al. *Learning heterogeneous spatial-temporal representation for bike-sharing demand prediction*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019.

[120] Park, C., et al. *ST-GRAT: A novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed*. in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020.

[121] Tian, Y. and L. Pan. *Predicting short-term traffic flow by long short-term memory recurrent neural network*. in *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*. 2015. IEEE.

[122] Fu, R., Z. Zhang, and L. Li. *Using LSTM and GRU neural network methods for traffic flow prediction*. in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. 2016. IEEE.

[123] Xiangxue, W., X. Lunhui, and C. Kaixun, *Data-driven short-term forecasting for urban road network traffic based on data processing and LSTM-RNN.* Arabian Journal for Science and Engineering, 2019. **44**(4): p. 3043-3060.

[124] Cho, K., et al., *Learning phrase representations using RNN encoder-decoder for statistical machine translation.* arXiv preprint arXiv:1406.1078, 2014.

[125] Paszke, A., et al., *Automatic differentiation in pytorch.* 2017.

[126] Huang, W., et al., *Deep architecture for traffic flow prediction: deep belief networks with multitask learning.* IEEE Transactions on Intelligent Transportation Systems, 2014. **15**(5): p. 2191-2201.

[127] Oh, S., Y.-J. Byon, and H. Yeo, *Improvement of search strategy with k-nearest neighbors approach for traffic state prediction.* IEEE Transactions on Intelligent Transportation Systems, 2015. **17**(4): p. 1146-1156.

[128] Lopez-Garcia, P., et al., *A hybrid method for short-term traffic congestion forecasting using genetic algorithms and cross entropy.* IEEE Transactions on Intelligent Transportation Systems, 2015. **17**(2): p. 557-569.

[129] Moylan, E.K. and T.H. Rashidi, *Latent-segmentation, hazard-based models of travel time.* IEEE Transactions on Intelligent Transportation Systems, 2017. **18**(8): p. 2174-2180.

[130] Abdullah, N.F., et al. *Vehicles classification using Z-score and modelling neural network for forward scattering radar*. in *2014 15th International Radar Symposium (IRS)*. 2014. IEEE.

[131] Contreras, J., et al., *ARIMA models to predict next-day electricity prices.* IEEE transactions on power systems, 2003. **18**(3): p. 1014-1020.

Appendix A: Sample Codes for Proposed Models

AGCRN module

Below is part of the sample code of the AGCRN module.

```
class GCRN(nn.Module):
    def __init__(self, node_num, dim_in, dim_out, cheb_k, embed_dim,
num_layers=1, num_node=170, use_att=True):
        super(GCRN, self).__init__()
        assert num_layers >= 1, 'At least one GCRN layer in the
Encoder.'
        self.node_num = node_num
        self.input_dim = dim_in
        self.num_layers = num_layers
        self.gcrnn_cells = nn.ModuleList()
        self.gcrnn_cells.append(AGCRNCell(node_num, dim_in, dim_out,
cheb_k, embed_dim, num_node, use_att))
        for _ in range(1, num_layers):
        self.gcrnn_cells.append(AGCRNCell(node_num, dim_out, dim_out,
cheb_k, embed_dim, num_node, use_att))

    def forward(self, x, init_state, node_embeddings):
        #shape of x: (B, T, N, D)
        #shape of init_state: (num_layers, B, N, hidden_dim)
        assert x.shape[2] == self.node_num and x.shape[3] ==
self.input_dim
        seq_length = x.shape[1]
        current_inputs = x
        output_hidden = []
        for i in range(self.num_layers):
            state = init_state[i]
            inner_states = []
            for t in range(seq_length):
                state = self.gcrnn_cells[i](current_inputs[:, t, :, :],
state, node_embeddings)
                inner_states.append(state)
```

```
            output_hidden.append(state)
            current_inputs = torch.stack(inner_states, dim=1)
        #current_inputs: the outputs of last layer: (B, T, N, hidden_dim)
        #output_hidden: the last state for each layer: (num_layers, B,
N, hidden_dim)
        #last_state: (B, N, hidden_dim)
        return current_inputs, output_hidden

    def init_hidden(self, batch_size):
        init_states = []
        for i in range(self.num_layers):
            init_states.append(self.gcrnn_cells[i].init_hidden_state(ba
tch_size))
        return torch.stack(init_states, dim=0)      #(num_layers, B, N,
hidden_dim)

class AGCRN(nn.Module):
    def __init__(self, args):
        super(AGCRN, self).__init__()
        self.num_node = args.num_nodes
        self.input_dim = args.input_dim
        self.hidden_dim = args.rnn_units
        self.output_dim = args.output_dim
        self.horizon = args.horizon
        self.num_layers = args.num_layers
        self.use_att = args.use_att

        self.default_graph = args.default_graph
        self.node_embeddings = nn.Parameter(torch.randn(self.num_node,
args.embed_dim), requires_grad=True)

        self.encoder = GCRN(args.num_nodes, args.input_dim, args.rnn_units,
args.cheb_k, args.embed_dim, args.num_layers, self.num_node, self.use_att)

        #predictor
        self.end_conv = nn.Conv2d(1, args.horizon * self.output_dim,
kernel_size=(1, self.hidden_dim), bias=True)

    def forward(self, source, targets, teacher_forcing_ratio=0.5):
        #source: B, T_1, N, D
        #target: B, T_2, N, D
        #supports = F.softmax(F.relu(torch.mm(self.nodevec1,
self.nodevec1.transpose(0,1))), dim=1)
```

```
        init_state = self.encoder.init_hidden(source.shape[0])
        output, _ = self.encoder(source, init_state,
self.node_embeddings)      #B, T, N, hidden
        output = output[:, -1:, :, :]   #B, 1, N, hidden

        #CNN based predictor
        output = self.end_conv((output))   #B, T*C, N, 1
        output = output.squeeze(-1).reshape(-1, self.horizon,
self.output_dim, self.num_node)
        output = output.permute(0, 1, 3, 2)    #B, T, N, C

        return output
```

AGCTN module

Below is part of the sample code of the AGCTN module.

```
class AGCTN(nn.Module):
    def __init__(self, args):
        super(AGCTN, self).__init__()
        self.num_node = args.num_nodes
        self.input_dim = args.input_dim
        self.output_dim = args.output_dim
        self.horizon = args.horizon
        self.use_att = args.use_att
        self.hidden_dim = 12

        self.default_graph = args.default_graph
        self.node_embeddings = nn.Parameter(torch.randn(self.num_node,
args.embed_dim), requires_grad=True)

        self.encoder = GCTN(args.num_nodes, args.input_dim, args.cheb_k,
args.embed_dim, self.num_node, self.use_att)

        #predictor
        self.end_conv = nn.Conv2d(1, args.horizon * self.output_dim,
kernel_size=(1, self.hidden_dim), bias=True)

    def forward(self, source, targets, teacher_forcing_ratio=0.5):
        init_state = self.encoder.init_hidden(source.shape[0])
        output, _ = self.encoder(source, init_state,
self.node_embeddings)      #B, T, N, hidden
```

```
        output = output[:, -1:, :, :]             #B, 1, N, hidden
        #CNN based predictor
        output = self.end_conv((output))          #B, T*C, N, 1
        output = output.squeeze(-1).reshape(-1, self.horizon,
self.output_dim, self.num_node)
        output = output.permute(0, 1, 3, 2)       #B, T, N, C

        return output
```

Self-attention module

Below is part of the sample code of the Self-attention module.

```
class SelfAttention(nn.Module):
    def __init__(self, in_dim, activation=F.relu):
        super(SelfAttention, self).__init__()
        self.chanel_in = in_dim
        self.activation = activation

        self.f = nn.Conv2d(in_channels=in_dim, out_channels=in_dim//8,
kernel_size=1)
        self.g = nn.Conv2d(in_channels=in_dim, out_channels=in_dim//8,
kernel_size=1)
        self.h = nn.Conv2d(in_channels=in_dim, out_channels=in_dim,
kernel_size=1)

        self.gamma = torch.zeros(1).cuda()
        #self.gamma = torch.zeros(1)

        self.softmax  = nn.Softmax(dim=-1)

        init_conv(self.f)
        init_conv(self.g)
        init_conv(self.h)

    def forward(self, x):
        """
            inputs :
                x : input feature maps( B X C X DIM)
            returns :
                out : self attention feature maps
        """

        m_batchsize, C, dim = x.size()
```

```
x_1 = x.view(m_batchsize, C, dim, 1) # B * C * dim * 1
f = self.f(x_1).view(m_batchsize, -1, dim) # B * C * dim
g = self.g(x_1).view(m_batchsize, -1, dim) # B * C * dim
h = self.h(x_1).view(m_batchsize, -1, dim) # B * C * dim

attention = torch.bmm(f.permute(0, 2, 1), g) # B * dim * dim
attention = self.softmax(attention)

self_attention = torch.bmm(h, attention) # B * C * dim
self_attention = self_attention.view(m_batchsize, C, dim)
out = self.gamma * self_attention + x

return out
```