# Augmenting Spatio-Temporal Dependencies for GNN-Based Short-Term Traffic Flow Prediction

A thesis presented in part fulfilment of the requirements of the Degree of Master of Science in Transportation Systems at the Department of Civil, Geo and Environmental Engineering, Technical University of Munich.


| | |
|---|---|
| **Supervisor** | M.Sc. Tao Guo |
| | M.Sc. Ningkang Yang |
| | Prof. Dr. Constantinos Antoniou |
| | Chair of Transportation Systems Engineering |
| **Submitted by** | Changxi Huang |
| | Arcisstrasse 21 |
| | 80333 München |
| **Submitted on** | Munich, 21.4.2025 |

# Acknowledgement

First, I would like to express my sincerest thanks to my supervisors, M.Sc. Tao Guo, M.Sc. Ningkang Yang, whose insightful ideas and consistent support have significantly guided my research. Their meticulous feedback and continuous encouragement do help me to explore the AI applications in the transport system and shape my research outcomes. I would also like to extend my appreciation to Prof. Dr. Constantinos Antoniou, who laid a solid foundation for my academic development. I was deeply impressed by his courses Applied Statistics for Transportation and Optimization for Transportation Systems, which broadened my horizon from traditional roadway design to AI-driven data processing. These courses played an important role in my topic selection and further exploration of this thesis.

Then, I am sincerely grateful to all the Technical University of Munich faculty members. I am impressed by the work efficiency of the administrative staff who can respond to students' problems in a timely manner, giving me a smooth learning environment. Besides, academic professors are so wise that I can learn diverse perspectives on the field of transportation.

I also want to express my gratitude to my fellow classmates. As a transportation engineer, group work is an essential part of completing complex and large-scale work efficiently. I am impressed by my classmates who not only share their highly efficient study methods with me, but also show outstanding engagement when doing the team tasks. Both my academic studies and personal characteristics are better shaped by them.

I am thankful to my family members from the bottom of my heart. Even though we are separated by thousands of miles, they have tried their best to provide me with their meticulous care and emotional support. Additionally, they offer me their generous financial assistance to support my study fees and living expenses so that I can fully focus on my studies without distraction.

Last, I would like to express my deepest appreciation to everyone who has supported my master's studies. I am glad to have my supervisors, friends, classmates, and family who lend me a hand when I feel frustrated. Completing this master's thesis is not the end of my studies, and I am always ready to dive into further studies for the rest of my life.

# Abstract

With the rapid development of urbanization, accurate traffic flow forecasting is becoming crucial in optimizing traffic operation efficiency. Recent studies mostly combine Graph Neural Network (GNN) and Recurrent Neural Network (RNN) to capture dynamic spatio-temporal traffic features due to their outstanding effectiveness. GNN plays a crucial role in aggregating spatial information from surrounding regions to a central area based on their spatial distribution and similarity. However, the misjudgment of spatio-temporal similarity is widely overlooked in existing methods, as they have not explicitly accounted for the negative influence of temporal misalignment when calculating traffic flow similarity. In particular, traffic propagation leads to high similarity between adjacent upstream and downstream road segments, but this similarity may be distorted due to phase discrepancies in the time series. Therefore, this thesis proposes a hybrid framework, named Dynamic Time Warping based Spatio-Temporal Graph Convolution Network, to address phase discrepancies in spatio-temporal traffic flow modeling. Initially, the Dynamic Time Warping (DTW) algorithm is employed to align the traffic flow sequence data, and the hidden spatio-temporal correlations are extracted, which are then used to construct dynamic adjacency matrices. Then, the traffic volume data are separately processed by Graph Convolution Network (GCN) and RNN, serving as spatial and temporal encoders to extract spatio-temporal dependencies. After that, the outputs from both encoders are fused and fed to an RNN decoder, reinforcing the model's effectiveness in capturing the temporal dependencies inherent in the data. A carefully designed experiment based on real-world data is conducted to evaluate the proposed model framework. Model robustness is further tested across pre-pandemic, mid-pandemic, and post-pandemic periods. The experimental results of the proposed framework demonstrate an improvement in model performance and indicate a certain level of robustness against the impact of the COVID-19 pandemic.

# Contents

# List of Figures

# List of Tables

# Glossary

## A



## C



## D



## F



## G



## I



## K



## L



## M



## R

## S



## T



## V

# 1. Introduction

## 1.1. Motivation

The rapid development of urbanization and the growth in the number of vehicles put tremendous pressure on the transportation system in a city. Accordingly, an accurate prediction of future traffic conditions is crucial to effectively assist urban transportation organization strategies, thus mitigating traffic congestion [1]. In recent years, the Intelligence Traffic System (ITS) has attracted widespread attention for its pivotal role in enhancing real-time traffic control with advanced data-driven methods since it enables the prediction of future traffic conditions by gathering and analyzing a massive amount of traffic data. Moreover, it enhances urban traffic management by reducing congestion and delivering early warnings [2].

Generally, traffic data has highly complex patterns since the traffic state constantly varies with time and space. Moreover, the present challenges in traffic prediction are caused by the complex traffic network structure and traffic state uncertainty. Thus, the primary difficulty is effectively capturing both spatial and temporal dependencies. The temporal dependency denotes that the previous traffic state can influence the current traffic condition, while the spatial dependency captures interactions among road segments. Recent studies have shown that Deep Neural Networks (DNNs) demonstrate superior performance in modeling the inherent nonlinear and non-stationary spatio-temporal characteristics in traffic systems [3]. Several typical RNN variants, such as Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM), are widely leveraged by researchers to extract temporal dependencies. Besides, these models can be hybridized with GNN to integrate the topology information of the road network [4].

Nevertheless, the performance of GNN heavily depends on the quality of the graph structure. Most existing studies [5] [6] [7] construct a predefined static graph based solely on the spatial distribution of nodes, which fails to reflect the dynamic latent characteristics of traffic flow and human daily routines. Although several studies, such as [6] and [7], have attempted to enhance graph structure quality by incorporating sequence similarity of traffic states, static graphs still struggle to represent dynamic spatial relationships accurately. For example, a strong correlation is typically observed between workplace and residential districts on weekdays, whereas the correlation tends to weaken on weekends and holidays [5]. Recent research has been conducted with dynamic graph adjacency matrices, such as adaptive graph adjacency matrices [8] [9], which can learn the dynamic spatio-temporal dependencies and automatically change the correlations between the nodes across different time periods.

However, the spatio-temporal phase discrepancies in traffic flow between network nodes have been largely overlooked in the existing studies. Notably, the traffic flow sequence on down-

stream road segments is often substantially influenced by the traffic flow on upstream segments, exhibiting high sequence similarity with a temporal phase lag that arises from the propagation of traffic state changes, such as traffic incidents, through the transportation network over time. Similarly, downstream congestion can affect upstream flow through backward shock waves [10].

To capture the aforementioned patterns, this thesis proposes a novel deep-learning framework named Dynamic Time Warping based Spatio-Temporal Graph Convolution Network (DTW-ST-GCN) to handle short-term traffic flow prediction. More precisely, given traffic flow sequence data, samples are generated using a sliding window technique, where each window corresponds to an individual adjacency matrix. These matrices share the same topological structure but differ in edge attributes, which are composed of three components: a spatial distance weight, a historical DTW-based weight, and a dynamically sliding window DTW weight. Among these, the spatial distance weight indicates the topology characteristics within the traffic network. Besides, the historical DTW-based weight reveals the historical long-term variation trend of the traffic state. Moreover, the dynamic DTW weight reflects real-time variations in temporal similarity, thereby enabling the graph structure to adapt dynamically to changing traffic conditions.

To avoid information leakage, the real-time DTW weight is calculated using only historical data within each window. Subsequently, the traffic flow sequence data are independently processed by a GCN and an RNN, which are spatial and temporal encoders that extract spatial and temporal dependencies, respectively. The outputs from both encoders are then fused and passed through an RNN decoder, which enhances the model's ability to capture the intrinsic temporal dependencies within the traffic data.

## 1.2. Objectives

The objectives of this thesis are divided into the following five parts:

- This thesis aims to highlight the feasibility and necessity of employing DTW to measure sequence similarity. Dynamic Spatio-Temporal Graph Convolution Network (ST-GCN) will be developed, in which the traffic network graph is composed of spatial distance weights, historical DTW-based weights, and real-time DTW-based weights.
- This study aims to enhance the learning of spatio-temporal dependencies by employing an encoder–decoder architecture.
- A dual-branch ablation study will be carefully designed to evaluate the individual and combined contributions of these components.
- The proposed model framework will be trained based on real-world data.
- The thesis will explore the model's robustness under public emergency disruptions.

## 1.3. Contributions

The contributions of this thesis are divided into the following four parts:

- This study leverages DTW to mitigate the effect of phase discrepancy when calculating the frequency-domain similarity between nodes, which has been largely overlooked in previous studies. Specifically, in this research, the similarities are fused by historical DTW-based weights and real-time DTW-based weights, which capture long-term variation trends and short-term emergent changes, respectively.
- A novel encoder-decoder architecture has been developed, demonstrating superior prediction accuracy compared to the traditional cascaded architecture in the experiments conducted on a real-world dataset.
- In this research, a robustness experiment is conducted across different time periods, revealing that the real-time DTW-based weights can retain a valid improvement during a public emergency event. However, the historical DTW-based weights can mislead the model to achieve lower performance since the traffic pattern has changed significantly during the event.
- The results of this thesis provide valuable policy insights for practical implementation. The proposed model framework is capable of providing early warnings of sudden changes in traffic states that are not predictable based on long-term trends. This capability allows traffic management stakeholders to respond more quickly and effectively to emergencies.

# 2. Literature Review

## 2.1. Traffic Flow Prediction Models in the Early Stage

Traffic state prediction aims to construct a model that is capable of predicting future traffic conditions based on historical observations. As one of the critical tasks in transportation management, short-term traffic flow prediction has been investigated in various models, which are typically classified into two categories: parametric models and non-parametric models [11].

During the early development of traffic prediction research, parametric models were frequently employed for their simplicity. Generally, models, such as Autoregressive Integrated Moving Average (ARIMA) [12] and Vector Autoregressive (VAR) [13], rely on key assumptions that the sequence data are stationary and obey certain statistical distributions, such as the Gaussian distribution. Mainly, the parametric models should be constructed by a pre-defined mathematical formula with parameters that can be optimized to better fit the sequence data. Thus, the quality of prediction outcomes largely depends on the appropriateness of the parameters and how well the formula can underlie the sequence trend. For example, a hybrid prediction model that combines improved seasonal ARIMA with multi-input autoregressive, whose parameters are optimized by a genetic algorithm, is proposed to provide accurate and efficient short-term multi-station flow prediction [14]. The parametric models have the advantage of lower computational demand due to their simple mathematical algorithms. However, due to the inherently complex and nonlinear temporal patterns in traffic data, these models are challenging to maintain their accuracy across different circumstances, as such patterns are intricate to approximate using limited parameters [3].

To address the limitations of parametric models, many efforts have been made to explore various traditional machine learning algorithms, which offer higher model complexity and improves predictive performance. Unlike parametric models, non-parametric models rely on the amount and quality of the observed data. It produces predictions by extracting the hidden information behind the historical data without explicit assumptions. The mainstream prediction models include Support Vectore Machine (SVM), K-Neareast Neighbor (KNN), and DNNs [9]. For instance, a research group [15] proposed a fully automatic dynamic procedure KNN, which can automatically adjust the model fitness to the data patterns, demonstrating superior performance than seasonal ARIMA.

Namely, the two methods have their advantages and shortcomings. Classical parametric models are favored in relatively stable circumstances for their simplicity and comparable predictions. However, the non-parametric models demonstrate better robustness in more complex scenarios such as public emergencies. The non-parametric models are in the research spotlights despite the need for a large amount of historical data to learn the inherent non-

linearity behind the data [16]. Table 1 summarizes the characteristics of parametric and non-parametric models.

**Table 1** Comparison between Parametric and Non-Parametric Models

| | **Parametric Models** | **Non-Parametric Models** |
|---|---|---|
| **Computational Cost** | Low (limited parameters) | High |
| **Flexibility** | Limited to assumptions | Adaptable to complex patterns |
| **Requirement of Functional Assumptions** | Required | Not required |
| **Sensitivity to Data Size** | Suitable for small sample sizes | Requires large datasets |
| **Sensitivity to Noise** | Robust to the noise | Can be affected by outliers |
| **Interpretability** | Easy (based on the assumptions) | Hard for model complexity |

However, the models mentioned above solely focus on the temporal dependencies, while the inherent spatial complexity of traffic flow in urban networks is neglected. As one of the branches of machine learning algorithms, DNNs have shown remarkable performance, especially in large traffic volume scenarios. Consequently, some studies have attempted to employ Convolution Neural Network (CNN) because of its ability to extract spatial dependencies in traffic networks. As illustrated in [17], a CNN-based model that transforms spatio-temporal traffic network data into images can improve accuracy by an average of 43% accuracy improvement compared to the traditional machine learning models. Still, the CNN models are more suitable for Euclidean space since they simplify the network graph into regular grids, but potentially bias the interpretation of spatial dependencies if urban traffic networks are generally irregular in structure [2]. Thus, scholars are currently devoting more efforts to Spatio-Temporal Graph Neural Network (ST-GNN) with a more intrinsic structure, capable of learning non-linear temporal dependencies and irregular graph structure.

## 2.2. State-of-the-Art DNNs in Traffic Flow Prediction

### 2.2.1. Current Challenges of Traffic Flow Prediction

Current traffic flow prediction tasks face similar challenges despite having different directions. Table 2 summarizes the current traffic flow prediction challenges, which are classified as capturing temporal dependencies, extracting spatial dependencies, and integrating external

factors [18].

**Table 2** Current Challenges of Traffic Flow Prediction

| **Typical Challenges** | **Characteristics** |
|---|---|
| Temporal Dependency | Traffic flow exhibits distinct temporal patterns throughout the day, such as rush hour and nighttime. |
| Spatial Dependency | Traffic flow exhibit distinct spatial patterns across different regions, such as between residential and industrial zones. |
| External Factors | Weather conditions, such as humidity and temperature, can significantly impact traffic flow. |

**Table 3** Trend of Research Interest in Traffic Flow Prediction Models

| **Reference** | **Model** | **Research Interests** | **Contributions** |
|---|---|---|---|
| [19] | RNN | Temporal | Emphasizes the effectiveness of attention-based RNNs in capturing long-term trends better than parametric models. |
| [17] | CNN | Spatial | Transforms traffic data into a graph structure, enabling the model to predict the traffic state of large-scale zones. |
| [20] | GRU & GCN | Spatial & Temporal | Designs spatial matrices and fuse interval closeness to extract spatio-temporal influences. |
| [9] | LSTM & GCN | Spatial & Temporal | Proposes a dual-adaptive adjacency matrix to capture the dynamic traffic pattern. |
| [1] | LSTM & Generative Adversarial Network (GAN) | Spatial & Temporal & Dependencies; External Factors | Designs a novel encoder-decoder framework to predict traffic flow under bad weather. |

Over the past five years, studies' interests have gradually focused solely on temporal dependencies to further integrate spatial dependencies and external factors. To demonstrate this, Table 3 presents the overview of recent research trends, along with the models, research

interests, and contributions. Therein, models that extract spatio-temporal dependencies outperform many traditional parametric models. For instance, [20] fused the daily and weekly temporal blocks as a representation of temporal patterns. Then, the spatial distribution of stations is encoded in a GCN model, which is then integrated with those temporal blocks. Eventually, the results illustrate an approximately 40% of improvement in accuracy compared with ARIMA.

### 2.2.2. Graph Learning based DNNs

Solely considering temporal dependencies limits the prediction accuracy, as the distinctions can strongly affect the traffic state. Besides, DNNs are appealing for their ability to capture complex data relationships. Consequently, studies have gradually fused the temporal and spatial information as input by employing a variety of DNNs. As mentioned above, CNN models treat a graph as a regular grid, limiting their ability to capture irregular spatial characteristics. Hence, GNN, an alternative to extract spatial dependencies, is becoming increasingly attractive for its potential to handle data from irregular and non-Euclidean domains [21]. More precisely, the GNN models can treat the traffic network as a graph, in which roads serve as connectivity and intersections as nodes, or even vice versa (Shown in Figure 1). Therefore, the network topology can be represented by adjacency metrics, by which the GNN can aggregate the spatial information by processing a convolution operation. Moreover, constructing adjacency metrics can be tricky, as they indicate the presence of connections and quantify the degree of connectivity. Most existing studies define connectivity by measuring similarity between nodes [6] or assigning weights based on physical distance [21].

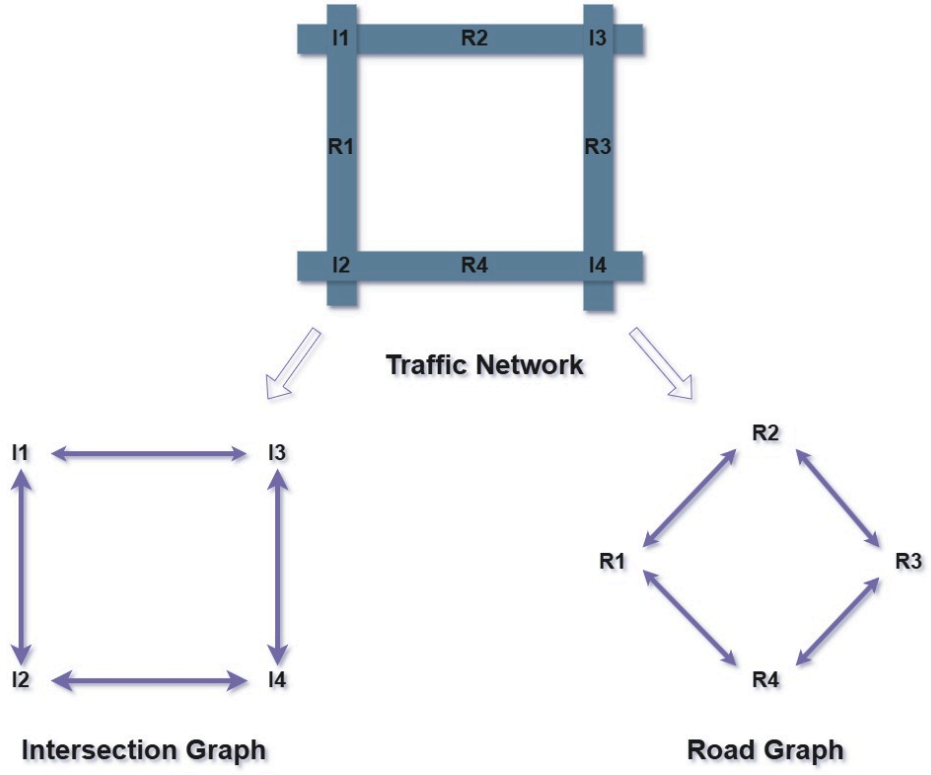



**Figure 1** Graph Construction Variation

The researchers are currently transitioning from CNN to GNN models since the traffic network has a graph structure that aligns with the capability of GNN. In response, various GNN versions have been developed and widely employed in traffic forecasting missions. The following Table 4 summarizes the recent well-developed variations of GNN models with their characteristics. Therein, the proposed models present that the aims of researchers are not only to extract the surrounding topology but also to incorporate additional modules for accuracy improvements.

However, the traffic data presents a dynamic and complex pattern that results in highly random interactions between road segments. The GNN model in many existing studies is therefore limited by their predefined static adjacency matrices, which inherently can not precisely reflect the temporal changes in the traffic network.

**Table 4** GNN Variations

| **Reference** | **Model** | **Characteristics** |
|---|---|---|
| [22] | GCN | Updates the central nodes' information by aggregating the features from surrounding nodes using a convolution operation. |
| [23] | Graph Attention Network (GAT) | Flexible and efficient in information aggregation for assigning the weight importance to the surrounding nodes. |
| [24] | Graph Multi-Attention Network (GMAN) | Improves the long-term prediction accuracy by introducing a transform attention layer to learning the importance of the future time steps. |
| [25] | Multiple Residual Recurrent Graph Neural Networks (MRes-RGNN) | Introduces Hop Mechanism to enhance the model sensitivity to capture periodic mode. Besides, the model fuses multiple Res-RGNN branches to extract both long-term and short-term temporal dependencies. |
| [26] | Graph Markov Network (GMN) | Introduces Markov Process to enable the model to account for the impact of the missing traffic information across time steps. |

### 2.2.3. Time Series Neural Network (TSNN)

As mentioned above, the traffic state shows strong temporal correlations within the sequence, and TSNN is designed to handle time series prediction tasks. Therein, one of the specific recurrent structures, named RNN, and its variants are widely employed to capture temporal dependencies due to their capability to fit stochastic and non-linear data features.

A classic RNN structure contains three layers, including an input layer, a hidden layer, and an output layer. The observed value at each time step is fed into the input layer, while the hidden layer integrates the historical temporal dependencies and propagates the hidden state across the time steps. Eventually, the model can produce predictions based on the learned sequential correlations. Numerous studies [27] [9] [16] have proved the RNN's capability to predict short-term traffic state. However, the high computational overhead and difficulties in tackling long sequence data can be disadvantages of RNN models. Besides, during the backpropagation process, the RNN models can experience gradient explosion and gradient vanishing as the models accumulate historical errors through hidden layers [11]. Even more critically, on the one hand, the model can lose the capability of capturing long-range temporal dependencies when gradient vanishing occurs, as low gradients prevent the effectiveness of updating parameters. On the other hand, gradient explosion can lead to model instability and even fail to converge.

In response to those drawbacks, GRU and LSTM are proposed by adding novel gate mechanisms into the classic structure of RNN models. The gate mechanism allows both models to retain important time steps while discarding those less relevant time steps by adjusting the weights [5]. An LSTM shows a better generalization under different traffic states for short-term traffic prediction tasks and achieves fewer prediction errors in road sections and intersections compared to classic machine learning algorithms [28]. Moreover, the performance of LSTM and GRU is compared in a short-term truck traffic flow prediction task, which illustrates that both models have similar performance and are significantly better than the traditional ARIMA model throughout the entire prediction period [29].

### 2.2.4. ST-GNN

Traffic state has strong temporal and spatial dependencies, so solely modeling the spatial or temporal characteristics severely limits the model's performance. Therefore, the ST-GNN comes into the spotlight. Its core concept is combining GNN and TSNN as a hybrid model to capture the temporal and spatial dependencies simultaneously. The current distinction among ST-GNN studies is whether static or dynamic adjacency matrices are adopted to represent the node connectivity.

Typically, the static adjacency matrices refer to fixed or predefined graph structures in which the binary connectivity can be determined by the existence of links between regions. Moreover, the spatial characteristics, such as node distance or similarity, can be fused and transformed into an edge weight that indicates the node's influence. Table 5 presents the current research interests of the methods to determine edge attributes.

One of the advantages of a static adjacency matrix is that it has relatively low computational demand as it can be pre-defined before training the models, and the graph structure is assumed to be unchangeable across the time steps. However, as a penalty for lower computational expense, the static adjacency matrix fails to represent the detailed temporal

characteristics. For example, a high correlation between residential zones and workplaces can be observed during the traffic rush hour. However, this correlation may drop dramatically during the off-peaks. Moreover, the node correlations can also decrease if the connecting routes are blocked because of the construction. Figure 2 pictures a detailed temporal variation in the correlations between the neighbor and central nodes. Generally, the farther the neighbor nodes are, the less the correlation is. Interestingly, some correlations can drop significantly due to the edge disconnections, which force drivers to take detours and consequently increase travel time.

**Table 5** Approaches for Static Graph Construction

| **Reference** | **Adjacency Matrix Weight Basis** | **Methods** | **Connectivity Derivation** |
|---|---|---|---|
| [6] | Nodes Similarity | DTW | Determines the nodes similarities based on historical traffic flow. |
| [7] | Stream Connectivity & Competitive Relationship | Traffic Flow Degree-Ware Weighting | Evaluates the traffic flow in-degree and out-degree at junctions to compute the edge attributes. |
| [21] | Topology-Based Connectivity | Binary Connectivity | Constructs an unweighted static graph with binary values (1 or 0) to indicate connectivity. |
| [24] | Topology-Based Connectivity | Distance Proximity | Measures the network distance to determine the edge weights (longer distance leads to less weight). |

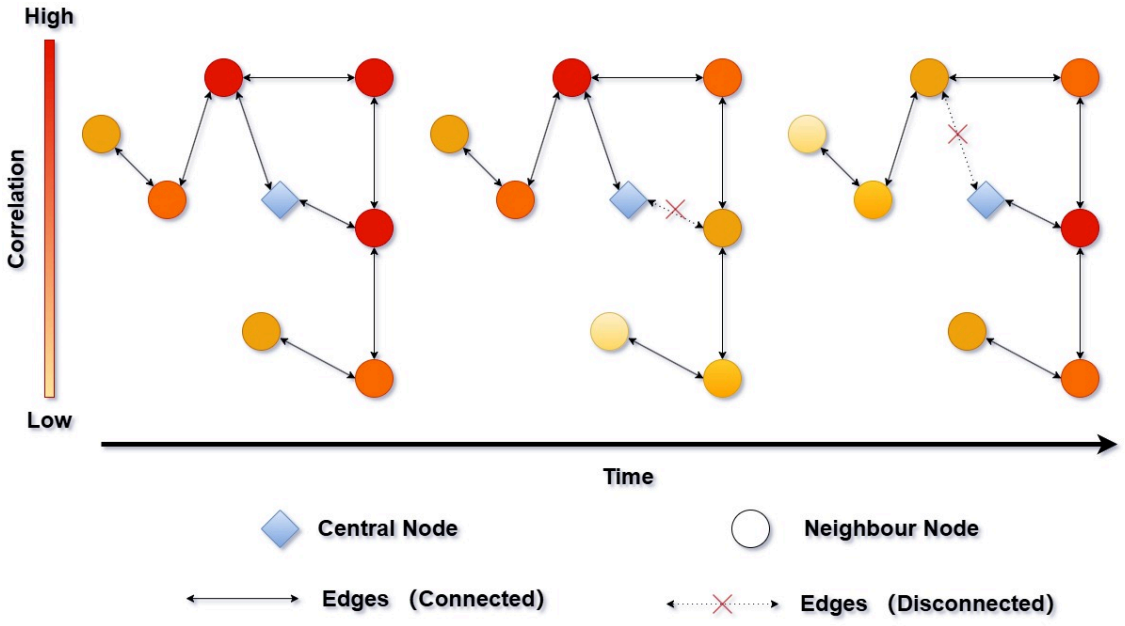



**Figure 2** ST-GNN Correlation

To address the aforementioned issues, researchers introduce dynamic adjacency matrices in ST-GNN for further performance improvement. Unlike the pre-defined static adjacency matrix, the dynamic adjacency matrix enables a changing graph structure across the time steps. Accordingly, this graph structure can precisely capture the fine-grained spatio-temporal events in traffic networks, such as edge disconnections due to accidents or congestion. In recent years, a growing number of studies have explored the contribution of dynamic graphs by leveraging various methods. Table 6 summarizes the respective papers and their proposals for constructing the dynamic adjacency matrix.

**Table 6** Approaches for Dynamic Graph Construction

| **Reference** | **Adjacency Matrix Weight Basis** | **Methods** | **Connectivity Derivation** |
|---|---|---|---|
| [5] | Nodes Similarity | Adaptive Graph | Encodes daily and weekly patterns into a temporal embedding matrix, which can be transformed as a weighted adjacency matrix. |
| [3] | Real-Time Traffic States | Dynamic Filters | Concatenates the nodes feature with hidden state from the previous time step, and applies a dynamic filter to determine its influence on neighbor nodes. |
| [30] | Attention Mechanism | Spatio-Temporal Attention | Implicitly increases edge weights of nodes with stronger influence during the model training phase. |
| [2] | Nodes Similarity | Vehicle Trajectory-Based | Three distinct methods are proposed, which can present the current traffic state, respectively. Dynamic adjacency matrices are then constructed by fusing dynamic matrices and a static binary matrix. |
| | | Traffic Feature Ratio-Based | |
| | | Traffic Feature Difference-Based | |

Both graph structures have their own advantages and weaknesses. In short, the dynamic adjacency matrix has better adaptability to sequential data characteristics but significantly lower computational efficiency. In contrast, the static adjacency matrix has weaker adaptability but less computational load as compensation. More characteristics of the static graphs and dynamic graphs are summarized in Table 7. Fortunately, modern computers are becoming increasingly powerful with rapid advanced development. Therefore, researchers are gradually leveraging dynamic graphs despite their high computational demand.

**Table 7** Graph Characteristics

| | **Static Graph** | **Dynamic Graph** |
|---|---|---|
| **Computational Cost** | Low | High |
| **Adaptability to emergencies** | Cannot reflect unexpected changes | High adaptability |
| **Model Complexity** | Simple structure | Relative complex |
| **Interpretability** | Easy for static structure | Hard for dynamic graph complexity |

As presented in Table 6, recent researchers tend to investigate node similarity based on traffic characteristics, such as traffic flow. However, there are no existing studies that explicitly account for the negative influence of temporal misalignment when calculating the node similarities with traffic flow sequence data. Traffic propagation inherently induces strong temporal correlations between upstream and downstream road segments. Precisely, the vehicle detectors on upstream road segments can capture traffic patterns earlier than those detectors located on downstream segments, as there is an inherent traffic flow propagation delay, and vice versa. Therefore, proposing a proper algorithm for similarity calculation to avoid phase discrepancy is crucial for genuine spatio-temporal relationships between segments [10].

To mitigate the impact of misalignment, this paper proposes ST-GCN models with a dynamic graph structure that incorporates the DTW algorithm to calculate the traffic sequence data similarity by aligning them in advance. A detailed experiment is designed to check the model's performance as well as its robustness across the different time periods of the COVID-19 pandemic. Additionally, several ST-GCN models with static graph structures are employed as baselines.

# 3. Methodology

## 3.1. Problem Definition

In this research, the traffic road networks are simplified into grid cells as nodes and defined as a directed graph $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$. Therein, the number of $N$ nodes are represented as $\mathbf{V} = \{v_1, v_2, ..., v_N\}$ while $\mathbf{E} = \{e_1, e_2, ..., e_M\}$ refers to the number of $M$ directed edges connecting grids cells. Accordingly, the adjacency matrix of the grid-based traffic network can be determined as $\mathbf{A} \in \mathbb{R}^{N \times N}$. Additionally, with a length of observed $T$ time steps of all nodes, the feature matrix $\mathbf{X} \in \mathbb{R}^{T \times N \times F}$ illustrates the traffic state with a number of $F$ traffic features of each node across the time steps.

The traffic prediction task refers to utilizing a sequence of historical data to forecast a sequence of future traffic states. Therefore, at the current time step $t$, the observed historical data of all nodes can be represented as $\mathbf{X} = [X_{t-T+1}, X_{t-T+2}, ..., X_t] \in \mathbb{R}^{T \times N \times F}$. Thus, the future traffic patterns can be written as $\mathbf{X}' = [X_{t+1}, X_{t+2}, ..., X_{t+T'}] \in \mathbb{R}^{T' \times N \times F}$, whereas $T'$ is the representation of the length of future short-term prediction time steps. Eventually, the prediction task can be summarized with an objective function, which aims to minimize the difference between the predicted value and the ground truth, shown in Equation (3.1).

$$min\ Loss = min\ \frac{1}{N} \sum_{i=1}^{N} ||\hat{\mathbf{X}}'_i - \mathbf{X}'_i||^2 \tag{3.1}$$

**Where:**

- $\hat{\mathbf{X}}'_i$ and $\mathbf{X}'_i$ are the prediction outputs and the ground truth, respectively

## 3.2. Graph Construction

### 3.2.1. Connectivity Determination

In the grid-based map $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, the two adjacency nodes are considered connected if there is a road across them. For a given node located at $(r, c)$, Equation 3.2 defines a set of neighbor nodes and the central node that are potential candidates for such connections.

$$\begin{aligned} N(r,c) = \{&(r,c),\ (r-1,c),\ (r+1,c),\ (r,c-1),\ (r,c+1),\\ &(r-1,c-1),(r-1,c+1),\ (r+1,c-1),\ (r+1,c+1)\} \end{aligned} \tag{3.2}$$

**Where:**

- $r$ and $c$ represent the row and column index, respectively.
- These neighbor nodes are in eight directions: North, Northeast, East, Southeast, South, Southwest, West, and Northwest.

To enhance the central nodes' connectivity, edges are additionally constructed to higher-order neighbors based on the shortest path principle with the method encapsulated in the Equation 3.3. In other words, an edge is deemed to exist between central nodes and higher-order neighbors if a vehicle can reach the neighbor nodes within the maximum allowable number of steps.

$$\mathbf{E}' = \mathbf{E} \cup \{(v, u) \mid v, u \in \mathbf{V},\ d(v, u) \leq Max,\ (v, u) \notin \mathbf{E}\} \tag{3.3}$$

**Where:**

- $v$ and $u$ are the central nodes and the higher-order neighbors, respectively.
- $d(v, u)$ denotes the shortest path length from the central nodes to the neighbors.

Figure 3 presents an example of edge construction. In the figure, the yellow cells represent the accessible cells with roads, while the blue cells refer to inaccessible regions. Initially, edges are constructed between adjacent valid cells. Then, additional edges are established between cell A and all other valid cells, as vehicles can reach them within an allowable maximum assumption of five steps.

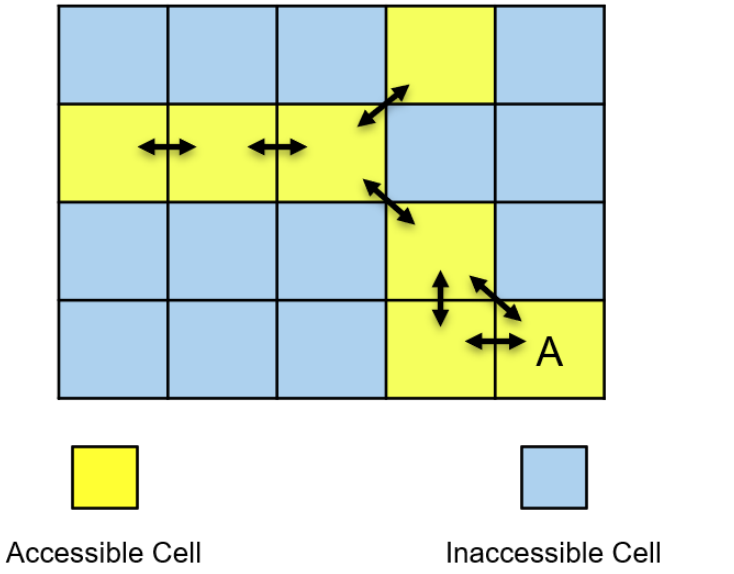



**Figure 3** Grid-Based Graph Connections Establishment Example

### 3.2.2. Distance-Based Static Graph

Normally speaking, closer neighbor nodes typically have a stronger influence on the central node. Therefore, this thesis replaces the traditional binary connectivity with a distance-weighted approach, which can effectively indicate the differences in connection strength. Specifically, the Gaussian kernel function is leveraged to determine the edge weight based on the length of the travel steps [31]. The distance is described in Equation 3.4, where the numerator refers to the step length from the center nodes to a neighbor node, and the denominator is employed to control the scale.

$$W_{Gaussian}(u, v) = \exp\left(-\frac{\|u - v\|^2}{2\sigma^2}\right) \tag{3.4}$$

**Where:**

- $W_{Gaussian}(u, v)$ denotes the edge weight between node $u$ and node $v$.
- $\|u - v\|^2$ is the squared Euclidean distance between two nodes.
- $\sigma$ controls the scale of the kernel.

Then, the distance-based weights are normalized with Equation 3.5 to produce a more stable weight value. The normalization allows the influence of neighbor nodes to be evaluated in proportion to the total weight, summing to one, so that the relative weight can be better captured.

$$\widetilde{A}_{Gaussian}(u, v) = \frac{\exp\left(W_{Gaussian}(u, v)\right)}{\sum_{k \in \mathcal{N}(u)} \exp\left(W_{Gaussian}(u, k)\right)} \tag{3.5}$$

**Where:**

- $k \in \mathcal{N}(u)$ is the set of neighbor nodes and the central node.
- $\widetilde{\mathbf{A}}_{\mathbf{Gaussian}}$ is the distance-based weighted static adjacency matrix.

### 3.2.3. Sequence-Based Static Graph

Upstream detectors detect approaching traffic flows ahead of downstream detectors due to the inherent delay of traffic flow propagation. Accordingly, the phase discrepancy can be exhibited in the traffic flow sequence from different detectors, resulting in the potential misestimation of sequence similarity. To mitigate the spatio-temporal phase discrepancy effect, the DTW is employed to capture temporal similarity by aligning the sequences[32]. In short, in this thesis, DTW captures the spatio-temporal correlations by finding the optimal warping path, which denotes the alignment between traffic flow sequence data. Figure 4 illustrates

how DTW aligns sequences through the optimal warping path.

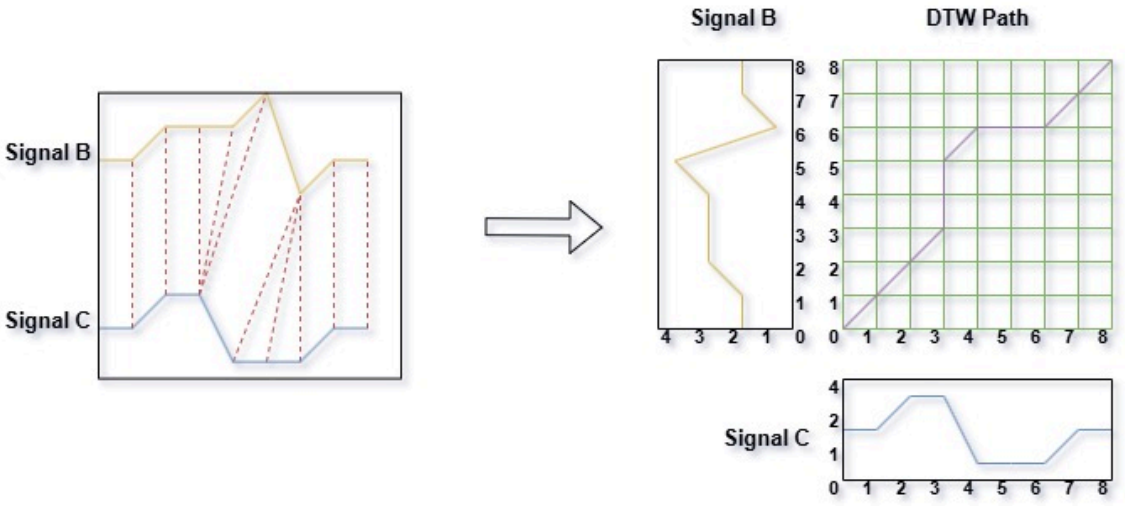



**Figure 4** DTW for Traffic Sequences Alignment

The sequence-based static graph determines the nodes' similarity by utilizing DTW on the historical data sequences within data samples, which are generated by sliding window techniques (introduced in section 5.1.3). For a given graph structure $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, at a given current time step $t$, the historical traffic sequences in two nodes with a number of T time steps can be written as $\mathbf{B} = (b_1, b_2, b_3, ..., b_T)$ and $\mathbf{C} = (c_1, c_2, c_3, ..., c_T)$, and a warping path can be written as $\mathbf{W} = (w_1, w_2, w_3, ..., w_M)$, in which $w_M = (t_n, t_k)$ refers to a alignment pair with index from both sequences [33]. Consequently, the similarity calculation process can be reformulated as an optimization problem summarized in Equation 3.6.

$$DTW(B, C) = min \sum_{m=1}^{M} d(B_{t_n}, C_{t_k}) \tag{3.6}$$

**Where:**

- $DTW(B, C)$ refers to the minimum length of the warping path.
- $d(B_{t_n}, C_{t_k})$ refers to the Euclidean distance between the two most similar elements originating from the two respective sequences.

Notably, the DTW algorithm operates under the following three criteria [33]. Figure 5 provides the visual cases that violate and satisfy the criteria.

1. **Boundary Condition:** The beginning and end of the warping path should align with the first and last time step of the sequence data.
2. **Continuity Condition:** The warping path should move in unit steps, and only transitions to adjacent points are allowed at a time.

3. **Monotonicity Violated:** A monotonic increase in the warping path is required.

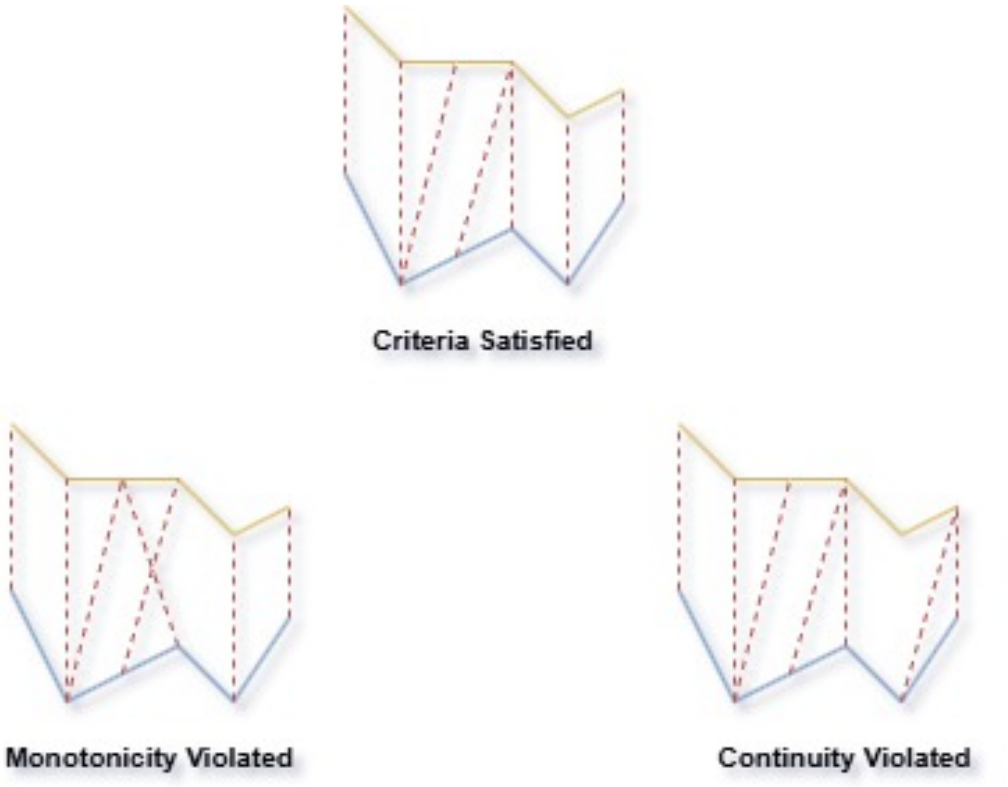



**Figure 5** Illustration of DTW Constraints: Satisfied vs. Violated Cases

The sequence-based static graph is designed to encode the long-term variation pattern of the traffic flow. Specifically, the weights in static graphs are calculated based on the historical data sequence, which precedes the current research period of interest. It is worth noticing that a shorter warping path leads to higher sequence similarity as less cost is needed for alignment. Accordingly, a negative sign is introduced to emphasize this inverse relationship. Besides, a temperature parameter is employed to smooth the weight distribution and mitigate the impact of outliers.

$$\widetilde{A}_{DTW}^{(hist)}(u,v) = \frac{\exp\left(-\frac{DTW(u,v)}{\tau}\right)}{\sum_{k\in\mathcal{N}(u)}\exp\left(-\frac{DTW(u,k)}{\tau}\right)} \tag{3.7}$$

**Where:**

- $k \in \mathcal{N}(u)$ is the set of neighbour nodes and the central node.
- $\widetilde{\mathbf{A}}_{\mathbf{DTW}}^{(\mathbf{hist})}$ is the sequence-based weighted dynamic adjacency matrix based on the historical long-term variation trend.
- $\tau$ refers to the temperature parameter which controls the scale of the weights

To better illustrate the construction of sequence-based static graphs, Algorithm 1 presents the procedure for generating sequence-based static adjacency matrices with historical traffic sequences. In this algorithm, the graphs are constructed within each sliding window to avoid the risk of misjudgment by excessive error accumulation across long time steps. Eventually, to summarize the long-term trend, their edge attributes are averaged to form a static graph.

**Algorithm 1:** Generating Sequence-Based Static Graphs with DTW (Step Size = 1)

**Input:** Grid-based graph $G = (V, E)$; historical traffic feature tensor that precede the target time period $X' \in R^{T \times N \times L}$, where $T$ denotes number of time step, $N$ refers to the number of Nodes and $L$ represents the number of features; history time steps length $H$; forecast time steps length $F$.

**Output:** Average sequence-based static graph $G_{avg}$

$S \leftarrow T - H - F + 1$ `// Total number of sliding windows samples`
Initialize lists: `DTW_graphs`
**for** $i \in \{0, 1, 2, ..., S-1\}$ **do**
  `hist_data` $\leftarrow X'[i : i + H]$
  $G_{dtw} \leftarrow$ copy of $G$
  **foreach** *edge* $(u, v)$ *in* `G_dtw` **do**
    $seq_u \leftarrow$ historical sequence at node $u$ from `hist_data`
    $seq_v \leftarrow$ historical sequence at node $v$ from `hist_data`
    `dtw_dist` $\leftarrow \texttt{DTW}(\texttt{seq}_u, \texttt{seq}_v)$
    `G_dtw`$[u][v][$"$DTW\_weight$"$] \leftarrow$ `dtw_dist`
  **end**
  Append $G_{dtw}$ to `DTW_graphs`
**end**
Initialize directed graph $G_{avg}$ with union of all nodes in `DTW_graphs`
Initialize maps `edge_weights` and `edge_counts`
**foreach** $G \in$ *DTW_graphs* **do**
  **foreach** *edge* $(u, v)$ *with* "$DTW\_weight$" $w$ *in* $G$ **do**
    `edge_weights`$[u, v] \mathrel{+}= w$
    `edge_counts`$[u, v] \mathrel{+}= 1$
  **end**
**end**
**foreach** *edge* $(u, v)$ *in* `edge_weights` **do**
  $w_{avg} \leftarrow$ `edge_weights`$[u, v]$/`edge_counts`$[u, v]$
  Add edge $(u, v)$ with weight $w_{avg}$ to $G_{avg}$
**end**
**return** $G_{avg}$

### 3.2.4. Sequence-Based Dynamic Graph

Similar to the sequence-based static graph, the sequence-based dynamic graph determines the edge weight via the DTW algorithm. However, the sequence-based dynamic graph is de-

signed to encode the short-term dynamic pattern of the traffic volume. The sequence-based dynamic weights are normalized with Equation 3.8, which allows the influence of neighbor nodes to be evaluated in proportion with the total weight summing to one.

$$\widetilde{A}_{DTW}^{(realtime)}(u,v) = \mathrm{Softmax}(A_{DTW}^{(realtime)}(u,v)) = \frac{\exp\left(-\frac{DTW(u,v)}{\tau}\right)}{\sum_{k\in\mathcal{N}(u)}\exp\left(-\frac{DTW(u,k)}{\tau}\right)} \tag{3.8}$$

**Where:**

- $k \in \mathcal{N}(u)$ is the set of neighbour nodes and the central node.
- $\widetilde{\mathbf{A}}_{\mathbf{DTW}}^{(\mathbf{realtime})}$ is the sequence-based weighted dynamic adjacency matrix.
- $\tau$ refers to the temperature parameter, which controls the scale of the weights.

Algorithm 2 illustrates the procedure for generating sequence-based dynamic adjacency matrices within each sliding window sample. Namely, starting from a time step, this algorithm calculates the nodes' similarity by DTW within a sliding window of length $H$. The dynamic graphs are stored in a list, which will be retrieved during the model training phase.

**Algorithm 2:** Generating Sequence-Based Dynamic Graphs with DTW and Sliding Window (Step Size = 1)

**Input:** Grid-based graph $G = (V, E)$; traffic feature tensor $X \in R^{T\times N\times L}$, where $T$ denotes number of time step, $N$ refers to the number of Nodes and $L$ represents the number of features; history time steps length $H$; forecast time steps length $F$.

**Output:** List of DTW-weighted dynamic graphs `DTW_graphs`

$S \leftarrow T - H - F + 1$ `// Total number of sliding windows samples`
Initialize lists: `DTW_graphs`
**for** $i \in \{0, 1, 2, ..., S-1\}$ **do**
  `hist_data` $\leftarrow X[i : i + H]$
  `G_dtw` $\leftarrow$ `copy of G`
  **foreach** *edge* $(u, v)$ *in* `G_dtw` **do**
    $\mathtt{seq}_u \leftarrow$ historical sequence at node $u$ from `hist_data`
    $\mathtt{seq}_v \leftarrow$ historical sequence at node $v$ from `hist_data`
    `dtw_dist` $\leftarrow \mathtt{DTW}(\mathtt{seq}_u, \mathtt{seq}_v)$
    `G_dtw`$[u][v][$"$DTW\_weight$"$] \leftarrow$ `dtw_dist`
  **end**
  Append `G_dtw` to `DTW_graphs`
**end**
**return** *DTW_graphs*

### 3.2.5. Graph Fusion

To encode the long-term temporal variation trend, short-term temporal variation trend, and spatial distribution characteristics, a composite graph is formed by linearly combining three

individual graphs. As shown in Equation 3.9, each graph contributes equally to the fusion through uniform weighting of its edge weights.

$$\widetilde{\mathbf{A}}_{\mathbf{fused}}^{(\mathbf{t})} = \frac{1}{3}\left(\widetilde{\mathbf{A}}_{\mathbf{DTW}}^{(\mathbf{realtime},\,\mathbf{t})} + \widetilde{\mathbf{A}}_{\mathbf{DTW}}^{(\mathbf{hist})} + \widetilde{\mathbf{A}}_{\mathbf{Gaussian}}\right) \tag{3.9}$$

**Where:**

- $\widetilde{\mathbf{A}}_{\mathbf{DTW}}^{(\mathbf{realtime},\,\mathbf{t})}$ is the sequence-based dynamic adjacency matrices at time step $t$.
- $\widetilde{\mathbf{A}}_{\mathbf{DTW}}^{(\mathbf{hist})}$ is the sequence-based static adjacency matrix.
- $\widetilde{\mathbf{A}}_{\mathbf{Gaussian}}$ is the distance-based static adjacency matrix.

## 3.3. Spatial Dependencies Encoding Module

### 3.3.1. Multi-layer Graph Convolution Network Structure and Formulation

As previously discussed, CNN was widely adopted for extracting spatial dependencies in the early stage of traffic prediction tasks. However, the research applications were limited to defining the data as a regular grid structure, while the real-world traffic networks exhibit highly irregular and non-Euclidean characteristics. To address this drawback, this thesis employs a multi-layer GCN based on the proposed framework in [34] to exploit the complex topology information.

In this thesis, the hidden state representation of nodes in $l^{th}$ layer is written as $\mathbf{H}^{(\mathbf{l})} = \left[h_1^{(l)}, h_2^{(l)}, h_3^{(l)}, ..., h_N^{(l)}\right] \in \mathbb{R}^{N \times Q^{(l)}}$, in which $h_N^{(l)}$ is the representation of node $N$ with the length of hidden size $Q^{(l)}$. With the pre-constructed hybrid dynamic graph, the general GCN mode can be formulated as shown in Equation 3.10. Therein, feature information is aggregated from the neighbor nodes to the central nodes, which constitutes the core process of graph convolution. This aggregation is mathematically implemented by the multiplication of the adjacency matrix and the feature matrix, which enables the selective propagation of information only from connected nodes. Figure 6 shows the convolution operation of one layer, which summarizes the first-order neighbor features to the central node.

$$\mathbf{H}^{(\mathbf{l+1})} = \sigma\left(\tilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}_{\mathbf{fused}}^{(\mathbf{t})}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(\mathbf{l})}\mathbf{W}^{(\mathbf{l})}\right) \tag{3.10}$$

**Where:**

- $\tilde{\mathbf{D}}$ denotes the degree matrix, which represents the weighted sum of connections (i.e., edges) associated with each node.

- $\widetilde{\mathbf{A}}^{(\mathbf{t})}_{\mathbf{fused}}$ is the fused dynamic adjacency matrix at time step $t$.
- $\mathbf{W}^{(\mathbf{l})}$ denotes the learnable weight parameter matrix.
- $\sigma$ refers to a non-linear activation function.
- $\mathbf{H}^{(\mathbf{l})}$ represents the hidden state of current layer.
- The starting layer $\mathbf{H}^{(\mathbf{0})}$ is denoted as $\mathbf{X}$ which is the input of the graph features.

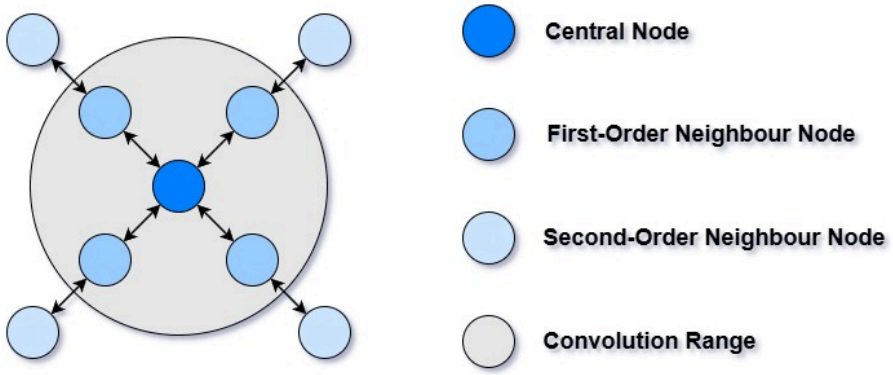



**Figure 6** Graph Convolution Visualization in GCN

Stacking multiple layers in GCN models can lead to over-smoothing since the node information is updated from the neighbor nodes in each layer, where the nodes features are gradually becoming similar and even indistinguishable. Therefore, this thesis introduces a Fully Connected Layer (FC) between the GCN hidden layers. Besides, the output dimensions from the Spatial Dependencies Encoding Module and Temporal Dependencies Encoding Module (introduced in section 3.4) can differ significantly, which can lead to a significant information dilution during the fusion process. The existence of FC enables flexible control over the output dimensions to mitigate the impact of feature domination by either the spatial or temporal encoder module. Equation 3.11 provides the mathematical formulation of the FC.

$$\mathbf{H}^{(\mathbf{l})} = \sigma\left(\mathbf{H}^{(\mathbf{l-1})}\mathbf{W}^{(\mathbf{l})\top} + \mathbf{b}^{(\mathbf{l})}\right) \tag{3.11}$$

**Where:**

- $\sigma$ refers to a non-linear activation function, which introduces non-linearity into the model.
- $\mathbf{H}^{(l-1)}$ is the hidden state output from the last GCN layer.
- $\mathbf{W}^{(l)}$ is a learnable weight matrix which learns the linear transformation relationships of features.
- $\mathbf{b}^{(l)}$ is a bias term.

### 3.3.2. Batch-wise Graph Processing for Efficient Training

**Algorithm 3:** Batch-wise GCN Encoder with Multi-Time-Step Aggregation (for $F = 1$)

**Input:** Batch input $\mathbf{X}_{\text{batch}} \in \mathbb{R}^{B \times N \times T \times F}$; Graph list $\mathcal{G}_{\text{batch}}$.

**Output:** Node features $\mathbf{O_G} \in \mathbb{R}^{B \times N \times T \times F}$

**if** $F \neq 1$ **then**
  **Warning:** Feature dimension $F \neq 1$. Multi-time-step aggregation optimization is not applicable.
  **Terminate execution**
**end**
**for** $i \in \{0, 1, \ldots, B-1\}$ **do**
  $G_i \leftarrow \mathcal{G}_{\text{batch}}[i]$
  Offset node indices by $i \cdot N$ to avoid conflicts
**end**
Merge all $\mathbf{G_i}$ into a single mega graph $\mathbf{G_{mega}}$ with shifted node indices
$\mathbf{x_{in}} \leftarrow \mathbf{X}_{\text{batch}}.squeeze(-1).reshape(B \cdot N, T)$ `// Treat` $T$ `as the feature dimension`
$\mathbf{x_1} \leftarrow \text{GCN}_1(\mathbf{x_{in}}, \mathbf{G_{mega}})$
$\hat{\mathbf{x}}_\mathbf{1} \leftarrow \text{FC}_1(\mathbf{x_1})$
$\mathbf{x_2} \leftarrow \text{GCN}_2(\hat{\mathbf{x}}_\mathbf{1}, \mathbf{G_{mega}})$
$\hat{\mathbf{x}}_\mathbf{2} \leftarrow \text{FC}_2(\mathbf{x_2})$
$\mathbf{x_3} \leftarrow \text{GCN}_3(\hat{\mathbf{x}}_\mathbf{2}, \mathbf{G_{mega}})$
$\hat{\mathbf{x}}_\mathbf{3} \leftarrow \text{FC}_3(\mathbf{x_3})$
$\mathbf{O_G} \leftarrow \hat{\mathbf{x}}_\mathbf{3}.unsqueeze(-1)$ `// Retrieve feature dimension`
**return** $\mathbf{O_G}$

In this thesis, a batch-wise graph processing method is proposed to reduce the computational cost. During the model training phase, the dataset is divided into batches randomly, which consist of multiple data samples. Each data sample refers to a sliding window corresponding to a unique graph structure (see section 5.1.3). However, GCN typically processes only a single graph at a time, resulting in a significant computational cost when handling large batch sizes. To overcome this limitation, the individual graphs corresponding to data samples in a batch are fused into a mega graph, where each original graph maintains structural independence by applying an index shift to prevent overlapping node indices. Consequently, the GCN model can efficiently process all data samples in a mini-batch simultaneously.

Additionally, the sequential information cannot be effectively captured into GCN since it typically performs convolution operations for each time step separately. This process dramatically increases the computational load when numerous historical time steps are processed independently. To overcome this issue, the input data can be reshaped to incorporate the temporal dimension as a feature dimension if the number of input features is equal to one. For instance, given an input tensor $\mathbf{X} \in \mathbb{R}^{B \times N \times T \times F}$, in which $B$ denotes the number of data samples, $N$ is the number of nodes in the mega graph, $T$ is the number of the historical time

steps and $F$ indicates the number of features, the tensor can be squeezed to $\mathbf{X} \in \mathbb{R}^{B \times N \times T}$ if $F = 1$. As a result, the temporal dimension $T$ takes the place of the feature dimension, which enables the model to process the features across all time steps simultaneously.

To better illustrate the computational load reduction process, Algorithm 3 presents the procedure of graph merging and spatial dependencies encoding with 3 GCN hidden layers.

## 3.4. Temporal Dependencies Encoding Module

To encode the temporal dynamics of urban traffic, this research employs the classic RNN model and its two variants for comparison. In this thesis, these models have the same approach to processing sequence data. For example, with an input tensor $\mathbf{X} \in \mathbb{R}^{B \times N \times T \times F}$, in which $B$ denotes the number of data samples, $N$ is the number of nodes in the mega graph, $T$ is the number of the historical time steps and $F$ indicates the number of features, the tensor can be reshaped to $\mathbf{X} \in \mathbb{R}^{(B \times N) \times T \times F}$. Accordingly, the models treat each node across batches as an individual sequence sample, which contains a number of $F$ feature dimensions with a temporal length of $T$.

### 3.4.1. RNN

The classic RNN architecture consists of three main components: observations (inputs), hidden states, and outputs. A directed link is established along the hidden states across the time steps to capture the temporal dependencies in the input sequence data, shown in Figure 7. With a given sequence data $\mathbf{X} = \{x_1, x_2, x_3, ..., x_T\}$, the observation at time step $t$ is $x_t$, the hidden state in the RNN model is $\mathbf{h_t}$, and the output is $\mathbf{O_t}$. Then, the RNN can be explained with Equations 3.12 [35]. The core of the RNN process involves recurrently updating the historical hidden state with current input, which enables the model to learn the temporal dynamics within the data sequence.

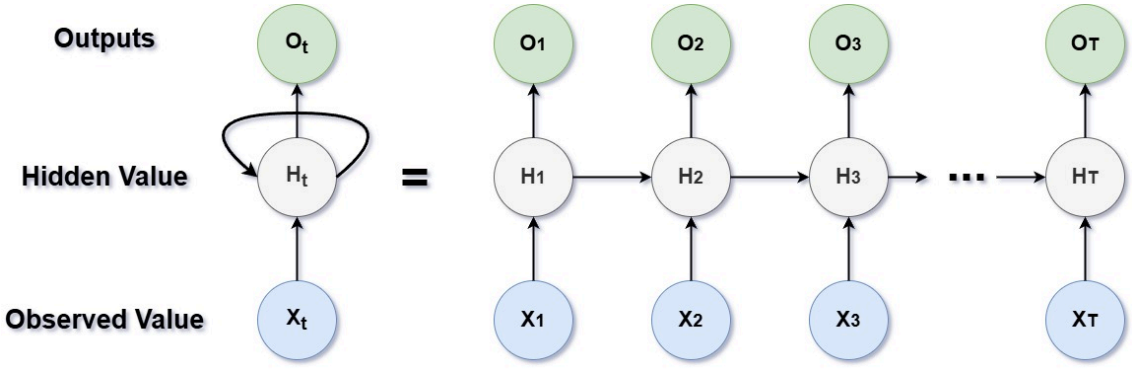



**Figure 7** Classic RNN Architecture

$$\mathbf{h_t} = \sigma_h(\mathbf{W_{hh}}\mathbf{h_{t-1}} + \mathbf{W_{hx}}\mathbf{x_t} + \mathbf{b_h}) \tag{3.12a}$$

$$\mathbf{O_t} = \sigma_o(\mathbf{W_{ho}}\mathbf{h_t} + \mathbf{b_o}) \tag{3.12b}$$

**Where:**

- $\mathbf{W}$ refers to a learnable weighted matrix.
- $\sigma$ is an activation function.
- $\mathbf{b}$ represents a bias term.

### 3.4.2. LSTM

However, RNN suffers from gradient vanishing and exploding problems during backpropagation due to repeated application of the chain rule when the sequence is long. To overcome these limitations, an updated variation of RNN is proposed, called LSTM, which modifies the RNN structures by adding a gate mechanism in the hidden layers. Specifically, three gates are constructed in LSTM cells, including input gate ($\mathbf{I_t}$), forget gate ($\mathbf{F_t}$), and output gate ($\mathbf{O_t}$), visualized in Figure 8. Besides, the LSTM introduces a cell state module, which works with the gates to selectively retain and abandon information. The cell state works as a memory unit which updates its stored information by element-wise addition and multiplication rather than repeated matrix multiplication. Thus, this process significantly stabilizes the gradient and mitigates the gradient vanishing and exploding problem. At a current time step $x_t$, the LSTM hidden layer workflow can be illustrated in Equations 3.13 [19].

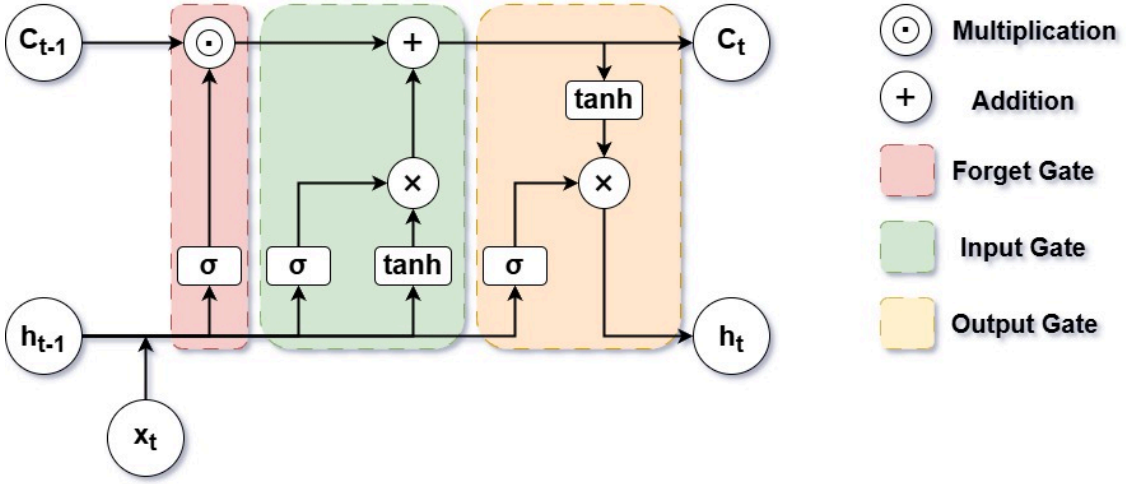



**Figure 8** LSTM Cell Architecture

$$\mathbf{I_t} = \sigma(\mathbf{x_t}\mathbf{W_{xi}} + \mathbf{h_{t-1}}\mathbf{W_{hi}} + \mathbf{b_i}) \tag{3.13a}$$

$$\mathbf{F_t} = \sigma(\mathbf{x_t}\mathbf{W_{xf}} + \mathbf{h_{t-1}}\mathbf{W_{hf}} + \mathbf{b_f}) \tag{3.13b}$$

$$\mathbf{O_t} = \sigma(\mathbf{x_t}\mathbf{W_{xo}} + \mathbf{h_{t-1}}\mathbf{W_{ho}} + \mathbf{b_o}) \tag{3.13c}$$

$$\widetilde{\mathbf{C}}_\mathbf{t} = tanh(\mathbf{x_t}\mathbf{W_{xc}} + \mathbf{h_{t-1}}\mathbf{W_{hc}} + \mathbf{b_c}) \tag{3.13d}$$

$$\mathbf{C_t} = \mathbf{F_t} \odot \mathbf{C_{t-1}} + \mathbf{I_t} \odot \widetilde{\mathbf{C}}_\mathbf{t} \tag{3.13e}$$

$$\mathbf{h_t} = \mathbf{O_t} \odot tanh(\mathbf{C_t}) \tag{3.13f}$$

**Where:**

- $\mathbf{I_t}$, $\mathbf{F_t}$, and $\mathbf{O_t}$ are the input gate, forget gate, and output gate, respectively.
- $\widetilde{\mathbf{C}}_\mathbf{t}$ and $\mathbf{C_t}$ are the cell input and current cell state.
- $\mathbf{W}$ is the learnable weighted matrix.
- $\mathbf{b}$ is the bias term.
- $\sigma$ is the activation function.

### 3.4.3. GRU

Even though LSTM is a capable model for time series prediction tasks, its complex structure causes a relatively high computational load, which limits its practical applications. As another alternative to RNN, GRU is a simplified structure that reduces the number of parameters but retains the essential gate mechanism to avoid gradient vanishing and exploding [36]. In short, the GRU framework extends the RNN hidden layers by additionally consists of the reset gate ($\mathbf{R_t}$), update gate ($\mathbf{Z_t}$), and a candidate hidden state ($\widetilde{\mathbf{h}}_\mathbf{t}$), shown in Figure 9. The reset gate re-evaluates the hidden state from the previous time step by selectively filtering the past crucial information that contributes to the candidate's hidden state. The update gate determines which information in the hidden state is carried forward with input at the current time step. Eventually, the hidden state at the current time step can be updated by jointly processing the output from these components. At a current time step $x_t$, the GRU hidden layer workflow can be illustrated in Equations 3.14 [35].

$$\mathbf{R_t} = \sigma(\mathbf{x_t}\mathbf{W_{xr}} + \mathbf{h_{t-1}}\mathbf{W_{hr}} + \mathbf{b_r}) \tag{3.14a}$$

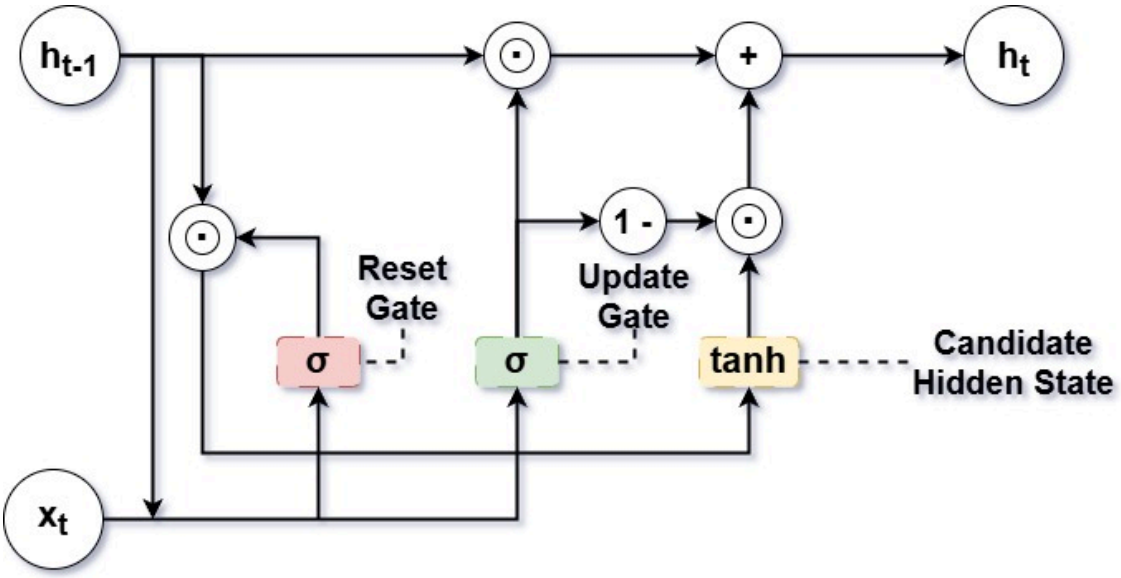



**Figure 9** GRU Cell Architecture

$$\mathbf{Z_t} = \sigma(\mathbf{x_t}\mathbf{W_{xz}} + \mathbf{h_{t-1}}\mathbf{W_{hz}} + \mathbf{b_z}) \tag{3.14b}$$

$$\widetilde{\mathbf{h}}_\mathbf{t} = tanh(\mathbf{x_t}\mathbf{W_{xh}} + (\mathbf{R_t} \odot \mathbf{h_{t-1}})\mathbf{W_{hh}} + \mathbf{b_h}) \tag{3.14c}$$

$$\mathbf{h_t} = \mathbf{Z_t} \odot \mathbf{h_{t-1}} + (1 - \mathbf{Z_t}) \odot \widetilde{\mathbf{h}}_\mathbf{t} \tag{3.14d}$$

**Where:**

- $\mathbf{R_t}$ and $\mathbf{Z_t}$ are the reset gate and update gate, respectively.
- $\widetilde{\mathbf{h}}_\mathbf{t}$ and $\mathbf{h_t}$ are the candidate hidden state and current hidden state, respectively.
- $\mathbf{W}$ is the learnable weighted matrix.
- $\mathbf{b}$ is the bias term.
- $\sigma$ is the activation function.

## 3.5. Spatio-Temporal Dependencies Fusion Decoding Module

As aforementioned, two encoders independently encode the spatial and temporal dependencies inherent in the data. However, these dependencies are extracted separately so that the joint spatio-temporal representation is not fully investigated. Thus, Equation 3.15a fuses their

outputs via element-wise averaging after their dimensions match with each other through FC in the Spatial Dependencies Encoding Module, which facilitates a balanced integration without introducing bias or information dilution. Subsequently, an RNN decoder with the same structure as the temporal dependencies encoder is introduced to further extract the temporal dependencies in the fused latent representation, shown in Equation 3.15b. The decoder can enable the model to effectively leverage both spatial and temporal dependencies so that the model's robustness and performance can be improved.

$$\mathbf{x_{fuse}} = \frac{1}{2}\left(\mathbf{O_G} + \mathbf{O_R}\right) \tag{3.15a}$$

$$\mathbf{O_{fuse}} = f_{RNN-Decoder}(\mathbf{x_{fuse}}) \tag{3.15b}$$

**Where:**

- $\mathbf{x_{fuse}}$ is the fused latent representation of spatio-temporal dependencies.
- $\mathbf{O_G},\ \mathbf{O_R} \in \mathbb{R}^{(B\times N)\times T\times 1}$ refers to the output from the spatial and temporal encoder, respectively.
- $\mathbf{O_{fuse}} \in \mathbb{R}^{(B\times N)\times T\times 1}$ denotes the final predictive output of the model.

## 3.6. Spatio-Temporal Dependencies Encoder-Decoder Architecture

Figure 10 visualizes the proposed ST-GCN Encoder-Decoder framework, which consists of three main blocks: spatial blocks, temporal blocks, and spatio-temporal blocks.

Generally, the spatial and temporal information is initially processed by the corresponding blocks for data represented in a graph format, such as a grid-based or pixel-based layout. The preprocessed node features help to assist the sequence-based graph construction. Then, the encoder modules can individually capture spatial and temporal dependencies with the hybrid graph and node features as inputs. Eventually, the encoder output is fused and passed through the decoder to produce final predictions.

The temporal-related block is responsible for dealing with traffic sequence features such as traffic flow and traffic speed. During data preprocessing, the nodes with extremely low average features are deemed as outliers and filtered out to reduce the model training cost while stabilizing the prediction performance. Furthermore, features are pre-normalized before feeding them into the temporal encoder to alleviate the impact of outliers so that the

model convergence can be enhanced. The classical RNN model and its variants are adopted as temporal encoders for their satisfactory ability to model data dynamics.

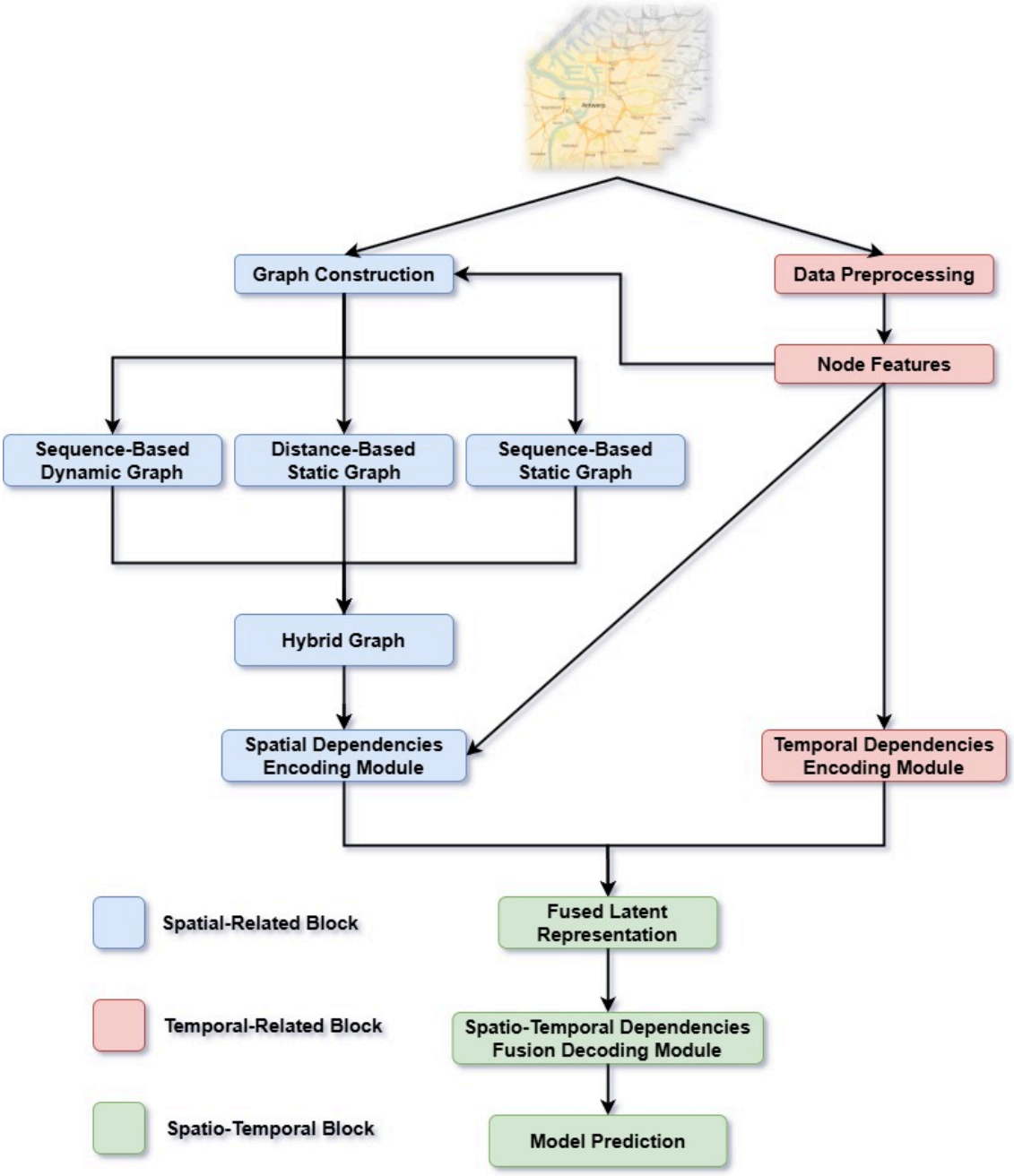



**Figure 10** Proposed ST-GCN Encoder-Decoder Framework

The spatial-related blocks enable the model to extract spatial dependencies from the graph. In this framework, three distinct graphs are generated based on node-level similarity and spatial distance. Then, a hybrid graph is integrated based on three separate graphs, which allows the model to investigate long-term variation trends, short-term emergent changes, and spatial correlation. Therefore, the spatial encoder with a hybrid graph can learn more informative and fine-grained spatial correlations from traffic data.

The spatio-temporal block is designed to jointly model spatial and temporal dependencies

and effectively learn their complex interaction. The latent representation of spatio-temporal dependencies can be obtained by fusing the outputs of encoders. As this thesis focuses on time series forecasting, the RNN structure is utilized in this decoder to preserve temporal continuity. Eventually, the Spatio-Temporal Dependencies Fusion Decoding Module further enhances the model performance to exploit the dynamic temporal patterns in the data by effectively leveraging both spatial and temporal contextual information.

## 3.7. Baseline Models

To validate the effectiveness of the proposed framework, the baseline models are constructed following the concept of dual-branch ablation design. Specifically, the dual-branch ablation design verifies the necessity and effectiveness of several key components within the proposed ST-GCN framework as well as the model robustness:

1. The inclusion of the GCN model for spatial dependencies modeling;
2. The encoder-decoder architecture;
3. The contribution of each graph pattern;
4. The impact of RNN variants for temporal modeling;
5. Model robustness under emergent public events;

Table 8 provides the names and explanations of each module used in the dual-branch ablation experiment for better clarity of the baseline model structural design. Figure 11 shares a format example of how these components and scenarios are constructed. Notably, a cascaded structure is employed in the comparative experiments against the encoder-decoder framework, as visualized in Figure 12. Therein, the encoder and decoder modules are removed. Instead, the graph data initially passes through the GCN model, followed by an RNN model to generate the predictions. The primary aim of Group 5 is to test the robustness of RT components under different public events. Table 9 summarizes the scenarios of public events, which are categorized as before the public event, during the public event, and the recovery period. Typically, the traffic data pattern is stable and predictable before the emergent public event, while the pattern can be highly irregular during the event due to unexpected disruptions. Eventually, the data pattern can return to a normal pattern during the recovery period. Accordingly, these three stages are designed to test model robustness under the effect of unexpected event eruptions.

**Table 8** Baseline Model Components

| **Component** | **Descriptions** |
|---|---|
| RNN | Recurrent Neural Network |
| LSTM | Long Short-Term Memory |
| GRU | Gated Recurrent Unit |
| GCN | Graph Convolution Network |
| Dis | Distance-Based Static Graph |
| RT | Real-time DTW for Sequence-Based Dynamic Graph |
| HT | Historical DTW for Sequence-Based Static Graph |
| Cas | Cascaded model framework |
| En | Encoder |
| De | Decoder |

**Table 9** Scenarios of Public Events

| **Scenarios** | **Descriptions** |
|---|---|
| BE | Before emergent public events |
| EPE | During emergent public events |
| RP | Recovery Period |

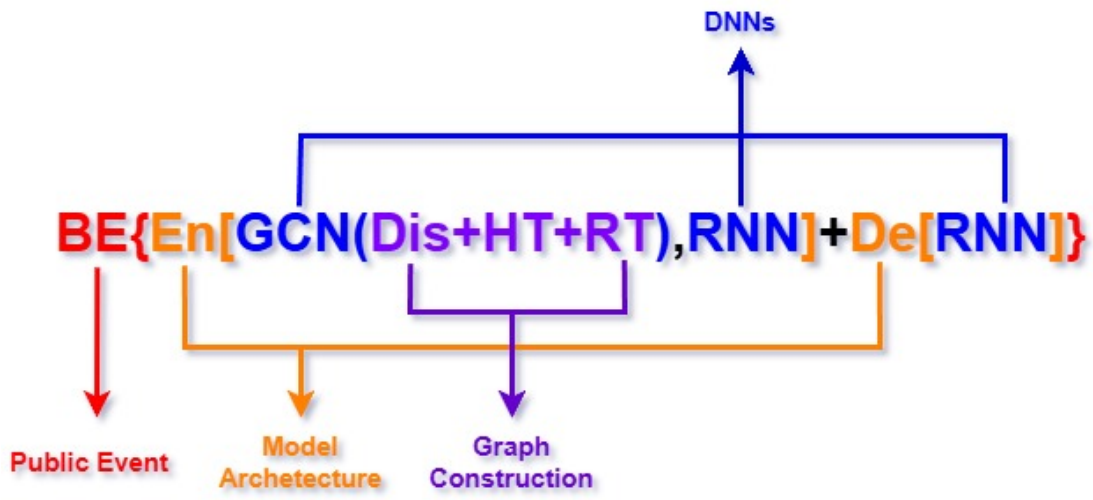



**Figure 11** Formalized Composition of Model Structure

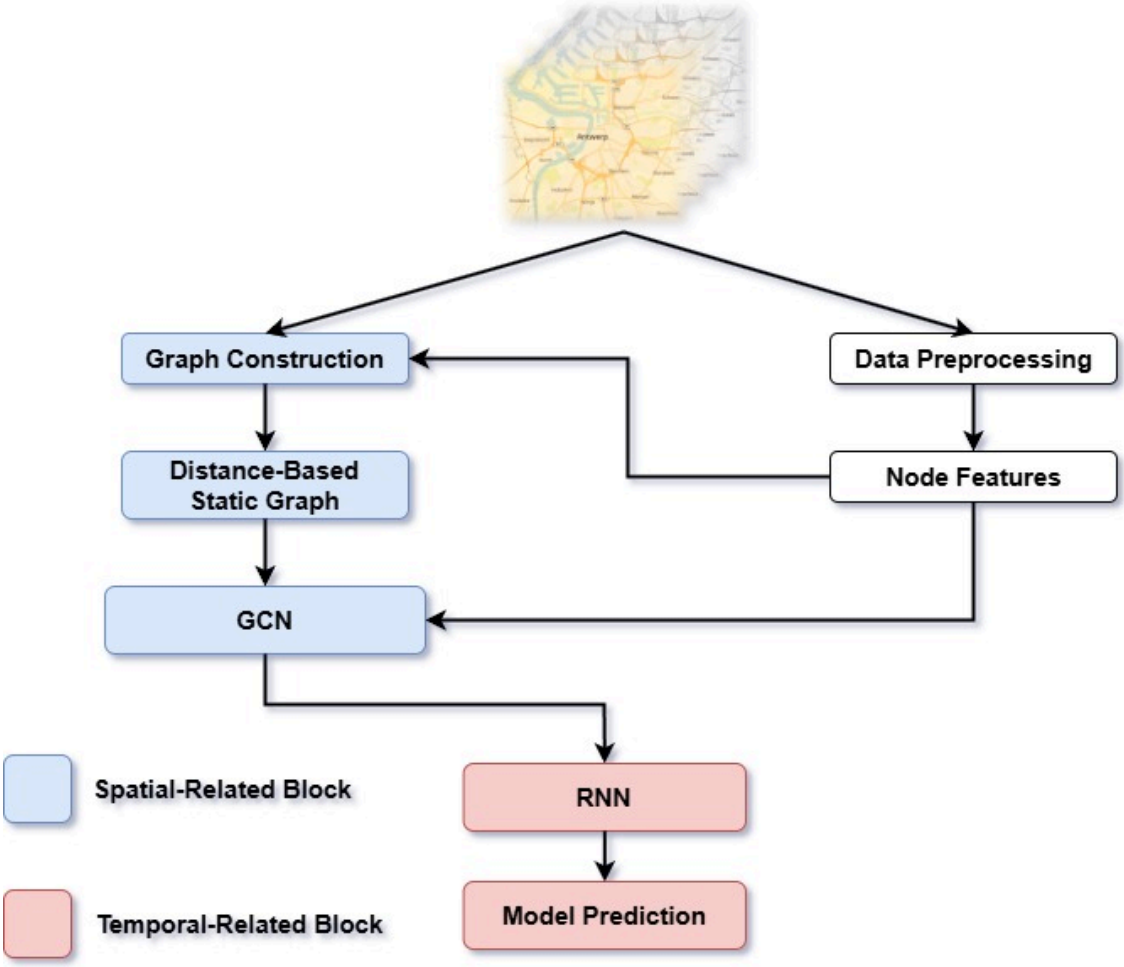



**Figure 12** ST-GCN Cascaded Framework

Model Group 1: Necessity and effectiveness check of the GCN model.

- RNN
- Cas[GCN(Dis),RNN]

Model Group 2: Necessity and effectiveness check of encoder-decoder architecture.

- Cas[GCN(Dis),RNN]
- En[GCN(Dis),RNN] + De[RNN]

Model Group 3: Necessity and effectiveness check of the contribution of each graph pattern.

- En[GCN(Dis),RNN] + De[RNN]
- En[GCN(Dis+RT),RNN] + De[RNN]
- En[GCN(Dis+HT),RNN] + De[RNN]
- En[GCN(Dis+HT+RT),RNN] + De[RNN]

Model Group 4: Necessity and effectiveness check of the contribution of RNN variants for temporal modeling.

- En[GCN(Dis+HT+RT),RNN] + De[RNN]
- En[GCN(Dis+HT+RT),GRU] + De[GRU]
- En[GCN(Dis+HT+RT),LSTM] + De[LSTM]

Model Group 5: Model robustness check under emergent public events.

- BE{En[GCN(Dis+HT+RT),RNN] + De[RNN]}
- BE{En[GCN(Dis+HT),RNN] + De[RNN]}
- EPE {En[GCN(Dis+HT+RT),RNN] + De[RNN]}
- EPE{En[GCN(Dis+HT),RNN] + De[RNN]}
- RP{En[GCN(Dis+HT+RT),RNN] + De[RNN]}
- RP{En[GCN(Dis+HT),RNN] + De[RNN]}

## 3.8. Evaluation Metrics

To comprehensively evaluate the model's performance, five evaluation metrics are employed, which target different aspects of prediction error between the ground truth $y_i$ and predicted value $\hat{y}_i$ among $n$ samples. The following contents present the selected evaluation metrics along with their descriptions and formulas.

1. Mean Absolute Error (MAE): It determines the overall performance of a model by calculating the average absolute difference between the ground truth and the predicted value, shown in Equation 3.16.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.16}$$

2. Mean Squared Error (MSE): It is more sensitive to large errors than MAE since there is a severe penalty due to the squaring operation, shown in Equation 3.17.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.17}$$

3. Coefficient of Determination ($R^2$): It evaluates the model fitness by calculating how much variance can be explained by the model, shown in Equation 3.18. Note the $\bar{y}_i$ refers to the average value of the ground truth.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{3.18}$$

4. Mean Absolute Percentage Error (MAPE): It removes the influence of data units by computing relative errors and expresses the errors directly through percentages, which is easy to interpret, as shown in Equation 3.19.

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{3.19}$$

5. Symmetric Mean Absolute Percentage Error (SMAPE): MAPE is sensitive to extremely low ground truth and zero value, which leads to infinite or disproportionately large errors. Accordingly, SMAPE refines the MAPE formula by adjusting the denominator, shown in Equation 3.20.

$$SMAPE = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2} \tag{3.20}$$

In short-term traffic state prediction tasks, the nodes may exhibit extremely low values or even zero. Therefore, this thesis strategically selects evaluation metrics at the node-level and the network-level to comprehensively assess the model performance. Specifically, the MSE and MAE are chosen at the node-level to avoid potential biases in evaluating the overall performance of the model. In contrast, the $R^2$, MAPE, and SMAPE are employed at the network-level to evaluate the model's ability to explain the global traffic variation patterns. Accordingly, at the network-level, the influence of units is eliminated, and the model accuracy can be intuitively interpreted by percentage-based error.

# 4. Case Study

## 4.1. Study Area

In order to evaluate the performance of the proposed framework, this thesis conducts carefully designed experiments based on real-world data, which was released by an online mapping company [37]. The original data was collected in Antwerp, the second largest city in Belgium, and a vital intersection of transportation networks in Europe due to its well-developed railway and highway systems. Figure 13 shows the network structure of Antwerp.

**Figure 13** Antwerp Traffic Network Structure

Generally, the development of the transport system has served as a foundation for urban growth in European capital cities [38]. Antwerp is a capital city renowned for its historical heritage, modern industry, and global ports. This city is located in the capital region of Brussels, which is near the largest coastal freight seaport in Northwest Europe. Besides, the capital lies in the intersection of several key highway and railway corridors, so there is a high intensity of transport activity all year round [38].

However, Antwerp has experienced severe traffic congestion over the last ten years, despite the high intensity of transport activity that has contributed to its economic growth. Several studies have been conducted to quantify the economic loss owing to traffic congestion hotspots in Antwerp. A recent survey has revealed that approximately €5.9 billion in economic losses are caused, placing Antwerp sixth among the European cities for long-term traffic congestion [39]. More recently, the highest recorded level of the average congestion rate on main roads was marked to be 464 hours-kilometers on 28 February 2024, leading to €439 million economic loss. The extreme level of congestion was mainly triggered by unexpected factors, such as snowfall and protests [40].

Consequently, it is essential for real-time traffic control to construct an effective short-term traffic prediction model that is capable of responding to emergency situations.

## 4.2. Data Description

### 4.2.1. Spatial Representation of the Road Network

The original dataset was collected from January to June 2019 and 2020, covering the COVID-19 pandemic. It is represented as a grid-based image, in which each cell corresponds to a $100m \times 100m$ spatial unit, maintaining a resolution of $495 \times 436$ pixels, shown in Figure 14. The node-level traffic features for one day are represented by a tensor with a shape of $(1, 288, 495, 436, 8)$, in which the second dimension denotes the aggregation of 5-minute intervals while the third and fourth dimension indicates the height and width of the graph. Thus, there are $495 \times 436$ nodes in this graph. Specifically, there are eight channels in the last dimension, which store the average traffic volume and speed in four directions (Northeast, Southeast, Southwest, and Northwest).

In addition to the traffic feature data, a static HDF5 file is also provided, which contains a $(9, 495, 436)$ tensor. The first channel of the first dimension is a gray-scale map which has the same resolution as Figure 14 while the rest of the channels indicate the cell neighbor connection in eight directions, including Northeast, East, Southeast, South, Southwest, West, Northwest, and North. Figure 15 presents the gray-scale map of Antwerp with a radial traffic network structure, which consists of several trunk roads connecting the urban and suburban areas. The gray-scale map is transformed from satellite imagery, in which the nodes are assigned an intensity value within the range of 0 to 255. The nodes are brighter with higher grayscale values, which indicates a wider road segment.

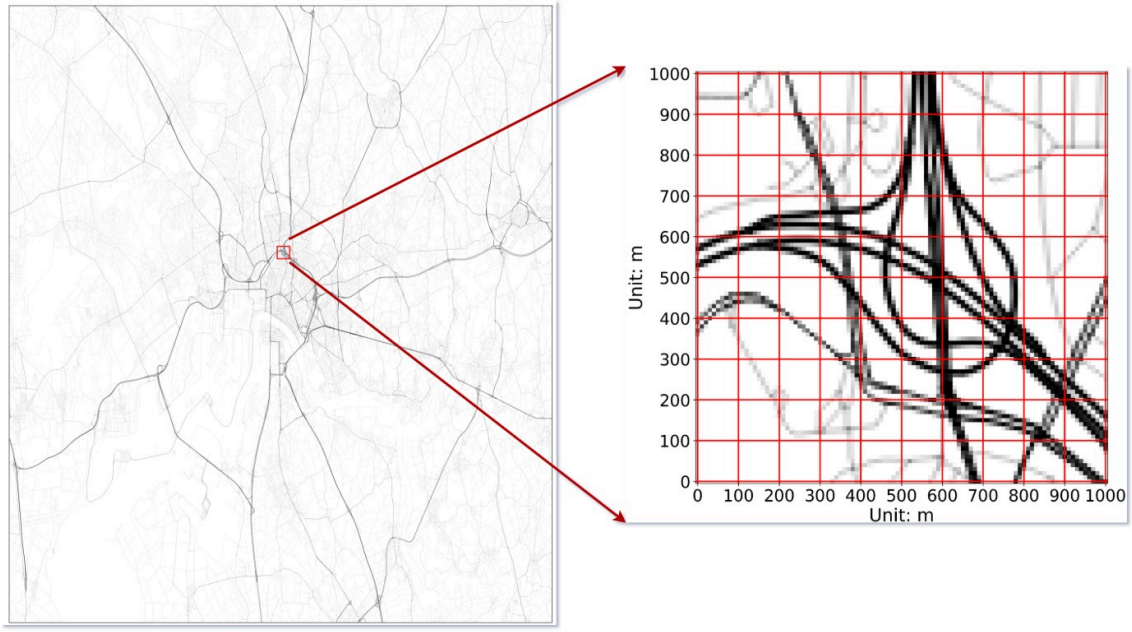



**Figure 14** Antwerp Traffic Network Graph Cells

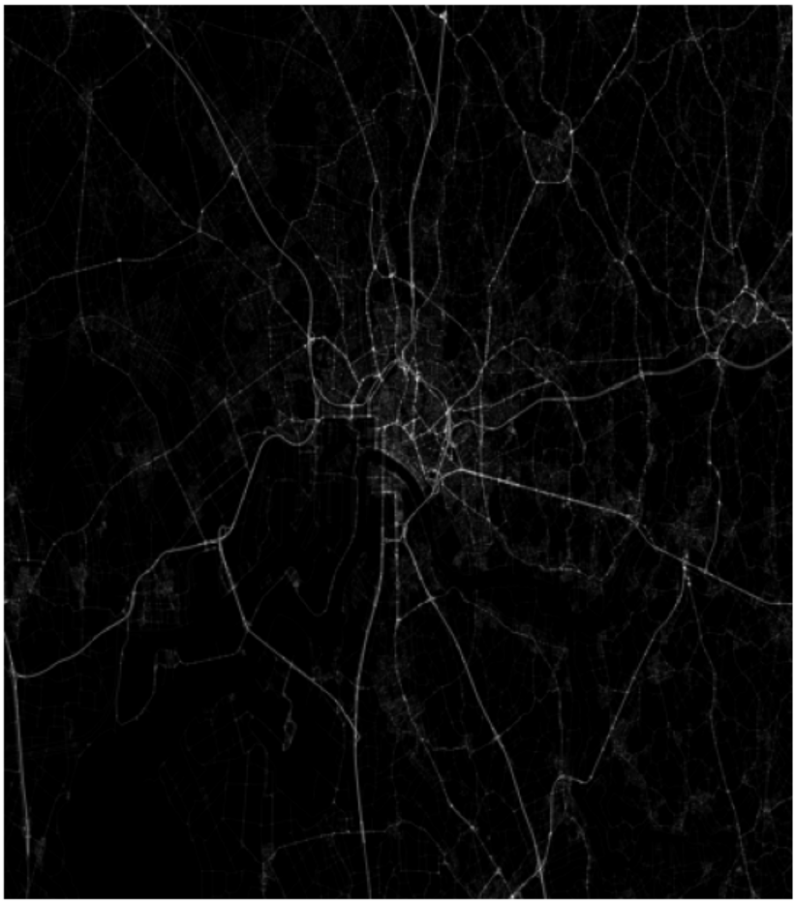


**Figure 15** Antwerp Gray-Scale Map

### 4.2.2. Temporal Patterns of Urban Traffic Volume

As aforementioned, the traffic features are represented by a tensor with the shape of $(1, 288, 495, 436, 8)$. Since the primary target of this thesis is short-term traffic volume prediction, the speed features are therefore discarded. Then, the node-level traffic volume is aggregated to a network-wide traffic volume in four directions shown in Figure 16.

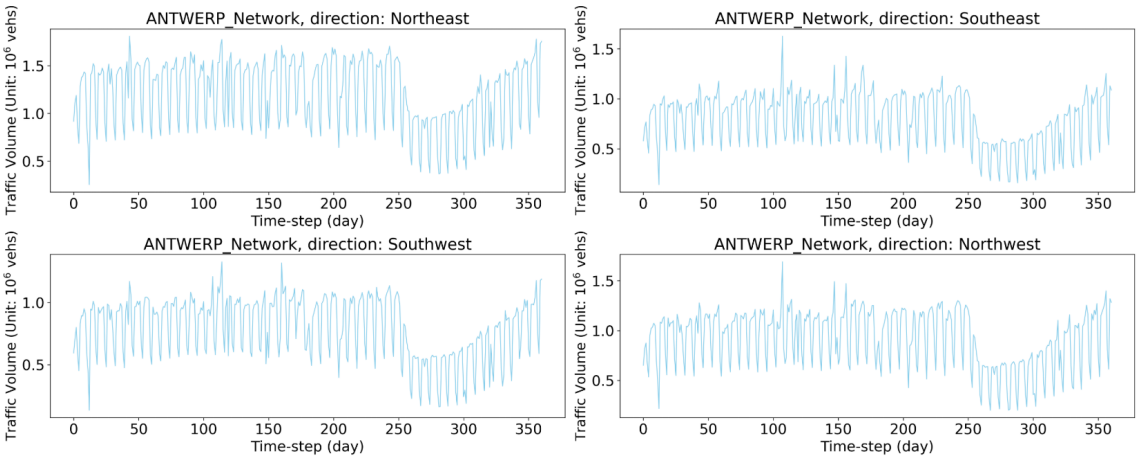



**Figure 16** Antwerp Traffic Volume in Four Distinct Directions

The temporal variation patterns in four directions are largely consistent, despite the numerical differences in traffic volumes. Therefore, to save the computational cost and better visualize the overall annual trend of urban traffic volume, the traffic volumes in all directions are aggregated within each node. Subsequently, node-level traffic volumes are summed to produce the network-wide traffic volume. The detailed aggregation steps are mathematically defined in Equation 4.1. Consequently, the total traffic volume of the traffic network from January to June in both 2019 and 2020 can be represented by a tensor $(\mathbf{Q_d})$ with a shape of $(361, 1)$.

$$\mathbf{Q_d} = \sum_{t=1}^{288} \sum_{i=1}^{495} \sum_{j=1}^{436} \sum_{k \in \{0,2,4,6\}} \mathbf{X_{t,i,j,k}} \tag{4.1}$$

**Where:**

- $\mathbf{X} \in \mathbb{R}^{361 \times 288 \times 495 \times 436 \times 8}$ represents the node-level traffic features collected over 361 days, where each dimension corresponds to day, time interval, grid height, grid width, and feature channel, respectively.

Figure 17 visualizes the traffic flow variation derived from the calculated tensor $\mathbf{Q_d}$. The curve in sky blue represents the traffic flow, while the shaded red, yellow, and green intervals correspond to the pre-pandemic (before March 2020), pandemic (March to April 2020), and recovery period (from mid-April 2020 onwards), respectively. In general, the curve exhibits a fluctuating pattern with regular peaks and valleys due to the existence of weekdays and weekends. It is worth stressing that the traffic volume was relatively steady before the pandemic. However, a significant decline in traffic volume was detected during the pandemic

period, which could be due to the implementation of quarantine restrictions. During the recovery period, even though the traffic flow began to recover, it generally remained lower than the level observed before the pandemic.

The short-term traffic prediction task is particularly challenging due to the difficulty of capturing reliable temporal patterns and unexpected factors. Thus, to better evaluate the model's robustness under certain unexpected events, this thesis selects data samples from the aforementioned three distinct periods. Besides, the sampling duration for each period should be kept short when developing short-term traffic flow prediction models. Therefore, the following three representative periods are chosen (shown in Figure 17):

1. Before pandemic: $16^{th}$–$30^{th}$ March 2019
2. During pandemic: $16^{th}$–$30^{th}$ March 2020
3. Recovery period: $16^{th}$–$30^{th}$ April 2020

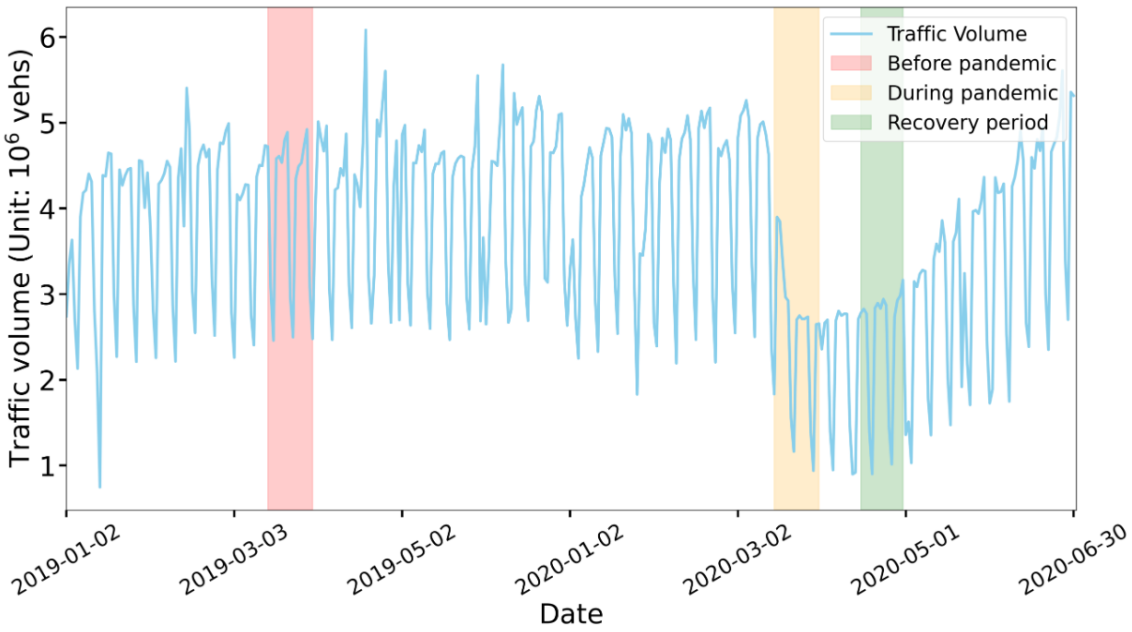



**Figure 17** Temporal Trends of Traffic Volume in Antwerp

In the literature review section, numerous studies have concluded that traffic flow inherently has strong temporal dependencies, so previous traffic patterns can significantly influence future traffic states. The inconsistencies in temporal patterns can have a negative impact on model prediction performance. Therefore, it is essential to analyze the traffic pattern immediately before each study interval. Figure 18 to Figure 20 visualize the average traffic pattern between the research interest periods and their respective preceding half-month time windows, which share the same scale for consistent comparison. Initially, the 5-minute intervals are transformed into 30-minute interval aggregation since the former is too short and produces a large number of zero values, which reduces the interpretability of the heatmaps. Therefore, these graphs are the representations of the average traffic volume per 30-minute interval. Additional work is limiting the data samples between $0^{th}$ to $98.5^{th}$ percentile to re-

move the outliers, which can distort the clarity.

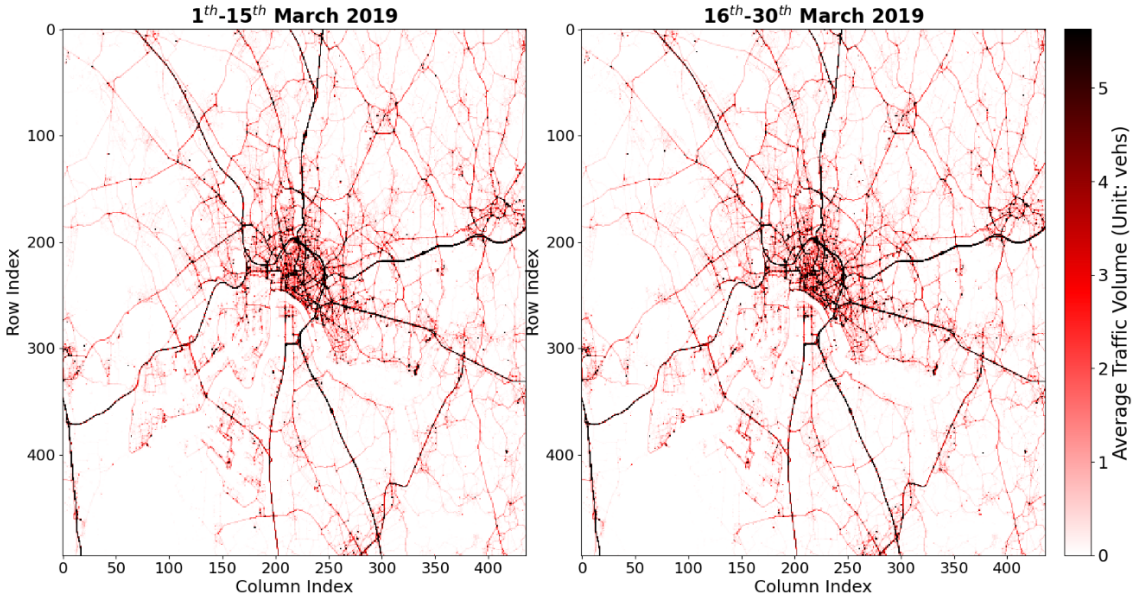



**Figure 18** Traffic Pattern Comparison in March 2019

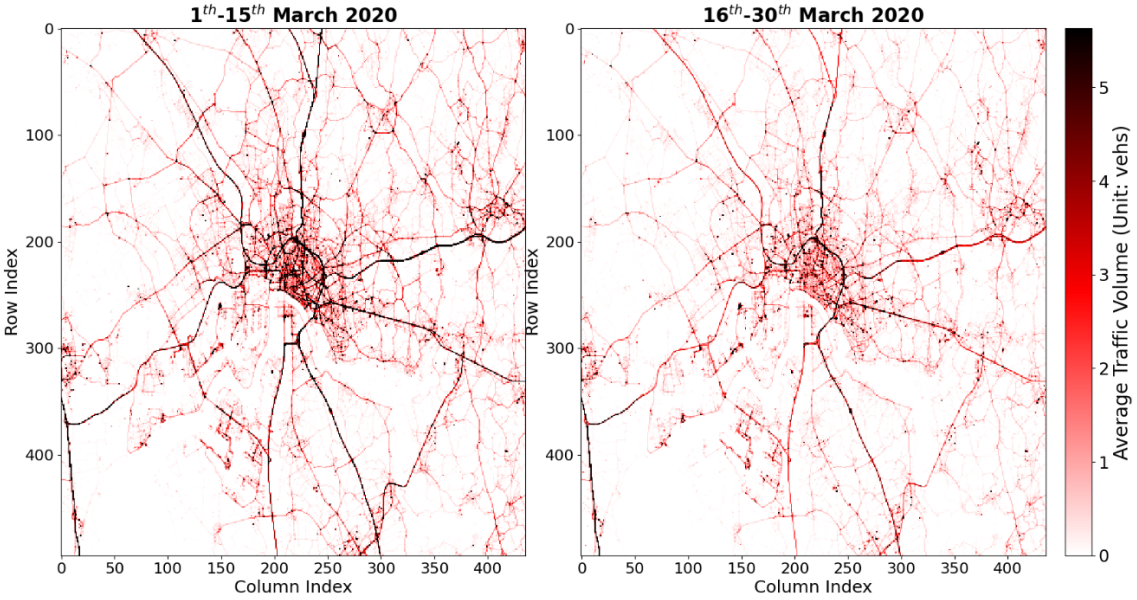



**Figure 19** Traffic Pattern Comparison in March 2020

The comparison figures reveal notable traffic pattern differences between the pandemic period and its preceding half-month, while there is relatively little change during the pre-pandemic and recovery periods. More precisely, a substantial drop in traffic volume within the Antwerp traffic network can be found during the pandemic period. Several dramatic reductions can be discovered on the arterial roads that connect the urban and suburban areas. The decrease in commuting between districts can be attributable to the lockdown policies forcing people to work at home. As a result, traffic volume within the urban network also declined significantly.

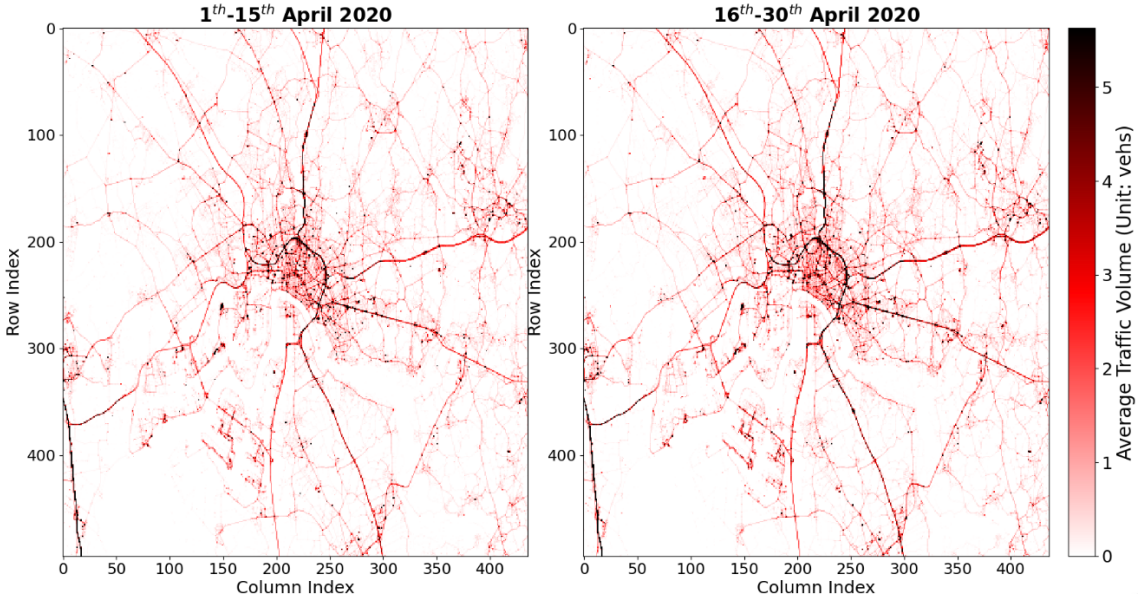



**Figure 20** Traffic Pattern Comparison in April 2020

A large number of white areas exist in those heatmaps, meaning that a part of the region has zero traffic. To better discuss the data distribution, Table 10 presents a statistical summary of node-level average traffic volumes for each research period of interest. The tables reveal that a minimum of 75% of nodes experience extremely low traffic or even zero traffic. Furthermore, it is worth stressing that the average traffic volume on the main roads is relatively lower than expected due to the inclusion of nighttime hours when there is minimal traffic activity. Interestingly, the highest traffic volume is found during the pandemic period, which is double that of the other two periods. One of the possible explanations is that some of the zones are restricted due to the quarantine policies, which force vehicles to use the same available roads so that an abnormal highest value is observed.

**Table 10** Statistical Overview of Average Traffic Flow in Three Periods of Interest

| **Statistics (Unit: vehs)** | **Before Pandemic** | **Pandemic** | **Recovery Period** |
|---|---|---|---|
| Minimum Value | 0 | 0 | 0 |
| $25^{th}$ Percentile | 0 | 0 | 0 |
| Mean Value | 0.389 | 0.247 | 0.235 |
| Median Value | 0.001 | 0.001 | 0.001 |
| $75^{th}$ Percentile | 0.099 | 0.081 | 0.085 |
| Max | 104.5 | 222.45 | 105.289 |

# 5. Result and Discussion

## 5.1. Data Preprocessing

### 5.1.1. Node Filtering

In section 4.2.2, it has been discussed that most of the nodes in the Antwerp traffic network graph are redundant since their average traffic volume is extremely low or even zero. These redundant nodes can act as noise as there is hardly a learnable pattern, resulting in destabilizing the model performance and costing too much computational load. Therefore, a primary task is to eliminate those outliers as much as possible to improve data quality. The node filtering process is based on the benchmark graph from March 2019. In other words, the same filtered graph structure is applied across all three periods of interest. Because employing a consistent graph structure enhances the comparability of the model robustness test across different periods.

To effectively extract the road network in the gray-scale map (Figure 15), nodes with an intensity value greater than 30 are selected to exclude those minor roads or even those corresponding to areas without roads. Subsequently, the rest of the nodes are filtered according to traffic volume, retaining those within $30^{th}$ to $100^{th}$ percentile. Further work is done by removing the remaining isolated nodes within the graph since they provide little information to the GCN model but increase the computational load. These steps ensure the exclusion of low-traffic noises and save the high-traffic nodes, which are the representation of the transportation network in Antwerp.

As a result, 5524 nodes are selected with the distribution shown in Figure 21. In the figure, the white nodes represent the valid nodes after screening. It can be found that most of the arterial roads are preserved, although there are some discontinuities along them. Table 11 presents the statistical summary of node-level average traffic volumes after screening for comparison, which proves the effectiveness of screening out redundant nodes. The statistical summary further proves that the remaining valid nodes are the representation of vehicle arterial roads with certain traffic activity.

Notably, the maximum traffic volume in the pandemic and recovery period is much smaller than the values before filtering. This is because the graph structure is constructed based on the traffic volume data from the benchmark period of 2019. Therefore, the exclusion of certain high-traffic nodes can exist due to the variations in traffic patterns across different time periods.

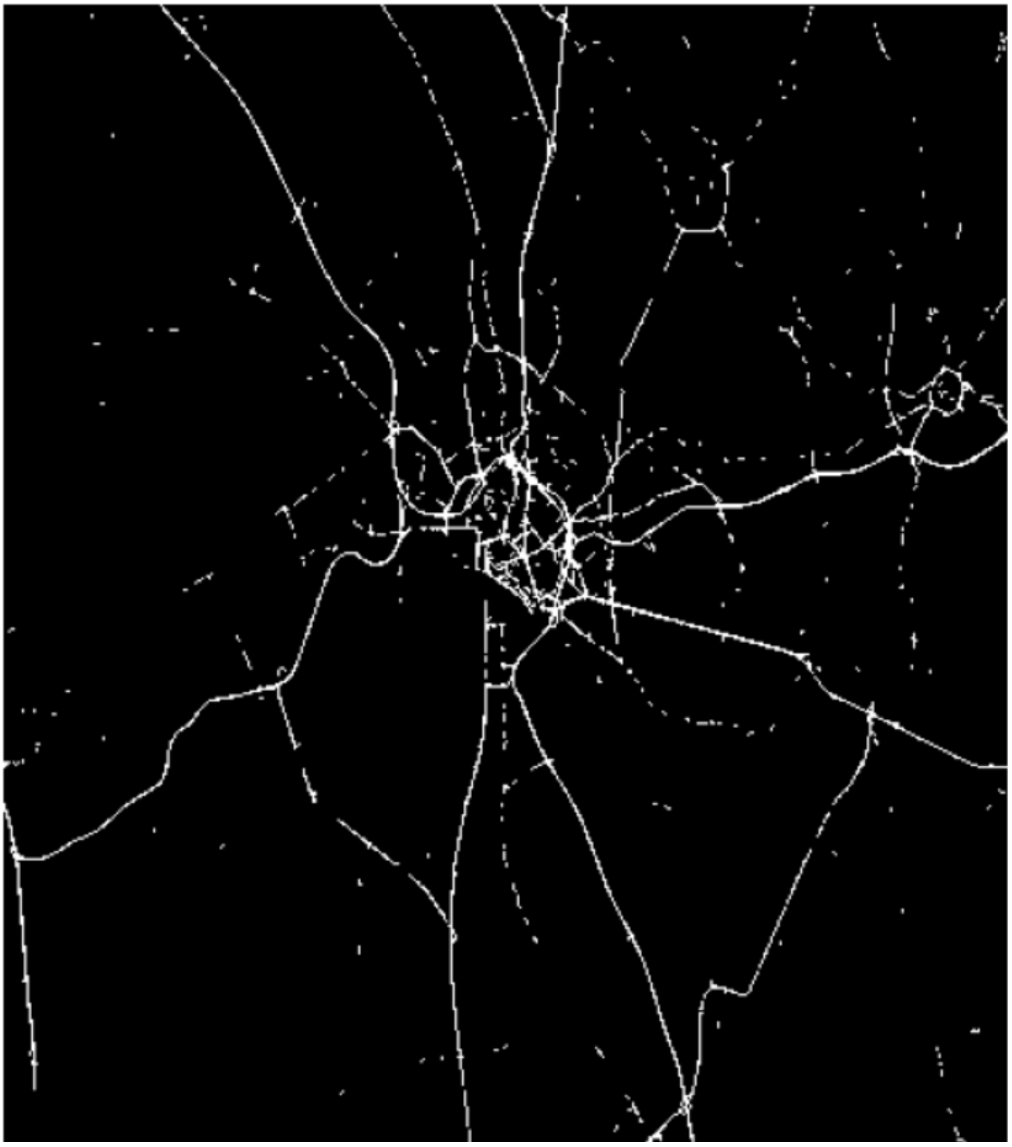


**Figure 21** Traffic Network After Node Filtering

**Table 11** Statistical Overview of Average Traffic Flow After Screening

| | **Before Pandemic** | **Pandemic** | **Recovery Period** |
|---|---|---|---|
| **Minimum Value** | 1.760 | 0.010 | 0.0010 |
| **25$^{th}$ Percentile** | 3.230 | 1.593 | 1.578 |
| **Mean Value** | 7.520 | 3.643 | 3.619 |
| **Median Value** | 5.370 | 2.603 | 2.540 |
| **75$^{th}$ Percentile** | 9.090 | 4.218 | 4.094 |
| **Max** | 104.500 | 57.392 | 69.597 |

### 5.1.2. Graph Construction

After node filtering, the remaining valid nodes are deemed as the representation of road segments. Therefore, the node connections can be established by Equation 3.2, which links each node with its spatially adjacent neighbors and itself. However, the spatial information of first-order connectivity can be limited to be aggregated since each node corresponds to a $100m \times 100m$ pixel. Thus, this thesis employed Equation 3.3 to extend the connectivity with a maximum allowable 5-step shortest path so that the effectiveness of information propagation in the GCN model can be enhanced.

The distance-based static graph is constructed by Equation 3.4. The parameter $\sigma$ is empirically set to 2.3 in a balanced manner with the given maximum path length of five steps. This value allows a smooth decay of weight values with respect to distance so that reasonable spatial relationships can be propagated in the GCN model. Table 12 presents the edge weights corresponding to different path lengths. Once the Gaussian distance weight is determined, the edge weights are then normalized by Equation 3.5.

**Table 12** Edge Weight in Distance-Based Static Graph

| **Path Length** | **Weight** |
|---|---|
| 0 | 1.000 |
| 1 | 0.910 |
| 2 | 0.685 |
| 3 | 0.427 |
| 4 | 0.220 |
| 5 | 0.094 |

As for the sequence-based dynamic and static graphs, the temperature parameters in Equation 3.8 and Equation 3.7 are empirically set to 5, which ensures the edge weight scales are aligned with those calculated from the distance-based graph. This setting prevents information dilution or dominance from any individual source when fusing them together.

Eventually, the fused graph has 5524 nodes with 90292 edges. It is worth stressing that the fused graphs are directed because of the application of the Softmax normalization. More precisely, the edge and the neighbor node vary from node to node so that the edge weight after normalization is asymmetrical. Consequently, the two adjacent nodes can share the same edge but different edge weights depending on the direction.

### 5.1.3. Sample Generation

As aforementioned in section 4.2.2, the time interval is re-aggregated into 30-minute bins to reduce the number of zero-value observations. Only the volume features are retained

from the raw data since the focus of this thesis is traffic flow prediction. Additionally, to save computational costs, traffic volumes in four directions in each node are aggregated because of the similar temporal pattern they exhibited. Consequently, the feature tensor for each of the three analysis periods is in the shape of $(720, 5524, 1)$, where the first dimension denotes the number of time steps.

Then, the data samples can be generated based on sliding window techniques with a pre-processed feature tensor $\mathbf{X} \in \mathbb{R}^{720\times5524\times1}$. Specifically, starting from time step $t$, a sliding window is represented as a sequence pair sample $\mathbf{X^{(t)}}, \mathbf{Y^{(t)}}$, which contains the historical sequence as input and the future sequence as ground truth. To generate the next sample, the window is shifted forward by $L$ time steps. In this thesis, the $L = 1$ is used to generate samples to improve the model sensitivity of short-term dynamics. Equation 5.1 mathematically formulates the sliding window technique.

$$\mathbf{D} = \left\{ \left( \mathbf{X^{(i)}}, \mathbf{Y^{(i)}} \right) \;\middle|\; i = 0, 1, \ldots, T - H_t - F_t \right\} \tag{5.1}$$

**Where:**

- The historical window $\mathbf{X^{(t)}} = \{x_{t-H_t}, x_{t-H_t+1}, x_{t-H_t+2}, ..., x_{t-1}\}$ has the length of $H_t$ time steps.
- The future window $\mathbf{Y^{(t)}} = \{y_t, y_{t+1}, y_{t+2}, ..., y_{t+F_{t-1}}\}$ has the length of $F_t$ time steps.
- In this thesis, the historical and future time steps are both set as 6.
- In this thesis, $H_t = 6$, and $F_t = 6$, the number of sample is $T - H_t - F_t + 1 = 709$.

After generating the data sample based on the sliding window techniques, the dataset is partitioned into three groups: a training group (60%), a validation group (20%), and a test group (20%). The training group is used to optimize the model parameters, such as learnable weight matrices, and minimize the training loss. The validation group is adopted to monitor the model's generalization ability and the existence of overfitting or underfitting since it is not involved in the model optimization. The test group is set to provide the final evaluation of the model's performance.

Before feeding the data samples into the model, they should be normalized to accelerate the model convergence and stabilize the training process. In this thesis, data standardization is applied within each sliding window pair, respectively, instead of globally. Moreover, to avoid information leakage, the standardization parameters used in the historical window, such as $\mu$ and $\sigma$, are recorded and applied to the corresponding future window. This local normalization strategy is employed based on the following three reasons:

1. The z-score standardization is employed in place of Min-Max normalization to mitigate the impact of outliers. Equation 5.2 illustrates the z-score standardization.

2. Enable the model to focus on short-term temporal patterns.

3. Local standardization reduces the influence of long-term trends and potential outliers at certain time steps.

**Algorithm 4:** Data Sample Generation (Step Size = 1)

**Input:** Filtered traffic data $\mathbf{X} \in \mathbb{R}^{T \times N \times 1}$; history window length $H = 6$; future window length $F = 6$; Group division ratio: $\alpha$, $\beta$, $\gamma$

**Output:** $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$, $(\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}})$, $(\mathbf{X}_{\text{test}}, \mathbf{Y}_{\text{test}})$

$S \leftarrow T - H - F + 1$ `// Total number of samples`

Initialize lists $X\_all \leftarrow [\,]$, $Y\_all \leftarrow [\,]$

**for** $i \in \{0, 1, \ldots, S-1\}$ **do**

  $X^{(i)} \leftarrow \mathbf{D}[i : i + H, :]$ `// Historical Sequence`

  $Y^{(i)} \leftarrow \mathbf{D}[i + H : i + H + F, :]$ `// Future Sequence`

  Append $X^{(i)}$ to $X\_all$; Append $Y^{(i)}$ to $Y\_all$

**end**

`Shuffle` and `Split` $X\_all, Y\_all$ into training ($\alpha$), validation ($\beta$), and test ($\gamma$).

**foreach** $(x, y)$ *in training group* **do**

  Compute mean $\mu$, std $\sigma$ of $x$

  $x' \leftarrow (x - \mu)/\sigma$ `// Z-score normalization`

  $y' \leftarrow (y - \mu)/\sigma$

  Append $x', y'$ to $\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}$

**end**

**foreach** $(x, y)$ *in validation group* **do**

  Compute mean $\mu$, std $\sigma$ of $x$

  $x' \leftarrow (x - \mu)/\sigma$

  $y' \leftarrow (y - \mu)/\sigma$

  Append $x', y'$ to $\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}}$

**end**

**foreach** $(x, y)$ *in test group* **do**

  Compute mean $\mu$, std $\sigma$ of $x$

  $x' \leftarrow (x - \mu)/\sigma$

  Append $x'$ to $\mathbf{X}_{\text{test}}$; Append $y$ (unscaled) to $\mathbf{Y}_{\text{test}}$

**end**

**return** $\mathbf{X}_{train}, \mathbf{Y}_{train}, \mathbf{X}_{val}, \mathbf{Y}_{val}, \mathbf{X}_{test}, \mathbf{Y}_{test}$

$$z = \frac{x - \mu}{\sigma} \tag{5.2}$$

**Where:**

- $x$, $\mu$, and $\sigma$ are the traffic flow value, mean value in the window, and standard deviation in the window, respectively.

Furthermore, it is worth stressing that the dual-branch ablation experiments of the model are executed based on the filtered nodes from the benchmark period of March 2019. The best-performing model combination is then applied to the subsequent pandemic and recovery periods to evaluate model robustness.

## 5.2. Model Preparation

### 5.2.1. Dropout Layer

One of the common issues in DNNs is overfitting, which denotes that the model is capable of producing satisfactory predictions with training data or a dataset that is highly similar to the training data, while failing to perform well with unseen data. To address this problem, this thesis employs the dropout layers, which were initially proposed by [41]. Namely, the dropout layers can randomly deactivate some neurons by setting their output to 0 and preventing them from backward propagation. Therefore, the model can be more robust in terms of a variety of data patterns. However, the settings of dropout layers can be tricky as too many layers or too much of a dropout rate can severely damage the model's performance. A mathematical formulation of the dropout mechanism is presented in Equation 5.3.

$$\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{r}, \quad r_i \sim Bernoulli(p) \tag{5.3}$$

**Where:**

- $x$ is the original output from the model.
- $r$ denotes the mask value (1 or 0), representing whether the output is saved or abandoned.
- The $r$ is generated followed by a Bernoulli distribution with the dropout ratio equal to $p$

### 5.2.2. Training Strategy

In DNNs, the gradient explosion or vanishing can occur if the weight matrices are not properly initialized. Specifically, a DNN transforms the input by passing it through multiple hidden layers, in which the linear transformation and non-linear activation are applied, shown in

Equation 5.4. However, for a given multi-layer DNN, the gradient updates should follow the chain rule in backpropagation, illustrated in Equation 5.5. Therein, the type of activation function, prediction data pattern, and weight matrices control the gradient magnitude of each layer so that they play a critical role in avoiding gradient explosion or vanishing.

$$\mathbf{h^{(l)}} = f^{(l)}(\mathbf{z^{(l)}}) = f^{(l)}\left(\mathbf{W^{(l)}}\mathbf{h^{(l-1)}} + \mathbf{b^{(l)}}\right) \tag{5.4}$$

$$\frac{\partial Loss}{\partial x} = \frac{\partial Loss}{\partial h^{(L)}} \cdot \prod_{l=1}^{L}\left(f'^{(l)}(\mathbf{z^{(l)}}) \cdot \mathbf{W^{(l)}}\right) \tag{5.5}$$

**Where:**

- $\mathbf{h^{(l)}}$ refers to the hidden state at $l^{th}$ layer.
- $\mathbf{W^{(l)}}$ is the learnable weight matrices at $l^{th}$ layer.
- $\mathbf{b^{(l)}}$ represents the bias term at $l^{th}$ layer.
- $f$ denotes the activation function, such as Rectified Linear Unit (ReLU).

To address this issue, this thesis employs Kaiming initialization [42], whose idea is to control the data pattern of the model prediction close to the ground truth so that the gradient magnitude can be kept at a stable and reasonable scale. Equation 5.6 derives the expected variance of the weighted matrix according to this expectation. In this thesis, assuming the data follows a normal distribution, the ReLU activation function (Equation 5.6b) is utilized in this thesis so that the expected variance should be adjusted to half since all negative values are set to zero.

$$Var[\mathbf{y}] = n \times Var[w\mathbf{x}] = n \times Var[w] \times \sigma^2 \tag{5.6a}$$

$$ReLU(\mathbf{x}) = max(0, \mathbf{x}) \tag{5.6b}$$

$$Var[\mathbf{y}] = \sigma^2 \Rightarrow Var[w] = \frac{1}{2n} \quad (after\ ReLU) \tag{5.6c}$$

**Where:**

- $n$ is the number of samples.
- $\sigma$ denotes the standard deviation.

After backpropagation, this thesis employs Adaptive Moment Estimation (Adam) to update model parameters. Specifically, this optimizer can adaptively tune the learning rate by leveraging the first-order and second-order moments of the gradients to adjust the model parameters. Besides, a reduction patience parameter is set to reduce the learning rate to further optimize model performance. This process is illustrated in Equation 5.7 [43].

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{5.7a}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{5.7b}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{5.7c}$$

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{5.7d}$$

$$\text{if } \nabla Loss \approx 0 \text{ for } p \text{ epochs} \quad \Rightarrow \quad \alpha \leftarrow \gamma \cdot \alpha \tag{5.7e}$$

**Where:**

- $m_t$ and $v_t$ are the first-order and second-order moment at current step $t$, respectively.
- $g_t$ is the current gradient at current step $t$.
- $\beta_1$ and $\beta_2$ are the decay rate parameter.
- $\hat{m}_t$ and $\hat{v}_t$ are the corrected moments.
- $\theta_{t+1}$ and $\theta_t$ are the updated and current model parameter, respectively.
- $\epsilon$ is utilized to avoid zero denominators.
- $\gamma$ is the reduced ratio of the learning rate.

During the training process, the model learns the data characteristics by minimizing the loss in the training group, while the validation group is used to monitor the model's generalization

ability. Therefore, overfitting can occur when the model continuously modifies its parameters with the training group, as the validation group does not directly participate in the updating process. To address the limitation, this thesis leverages the early stopping technique, which allows the model to record the validation loss of each epoch. Once there is no further improvement in validation loss after several consecutive epochs, the training will be paused, and the model will be assigned parameters corresponding to the epoch with the best validation performance. Algorithm 5 presents the procedure of the Early Stopping technique.

**Algorithm 5:** Early Stopping during Model Training

**Input:** Number of epochs $E$; patience threshold $P$; training group $X_{\text{train}}$; validation group $X_{\text{val}}$

**Output:** Trained model with best validation performance

```
Initialize model parameters randomly
best_val_loss ← ∞
epochs_no_improve ← 0
for epoch ← 1 to E do
    Train the model on X_train
    Compute validation loss val_loss on X_val
    if val_loss < best_val_loss then
        best_val_loss ← val_loss
        Save current model parameters as best_model
        epochs_no_improve ← 0
    else
        epochs_no_improve ← epochs_no_improve + 1
        if epochs_no_improve ≥ P then
            break                                  // Early stopping triggered
        end
    end
end
Restore model parameters from best_model
return Trained model
```

### 5.2.3. Group 1 Model hyperparameters: Effectiveness of the GCN Model

The aim of this group is to prove the effectiveness of the GCN model. The comparative models included in this group are RNN and Cas[GCN(Dis),RNN]. Table 13 and Table 14 illustrate the hyperparameters of each model framework. Note that the Layer Index column denotes the data flows through the model component during the forward propagation process.

**Table 13** Group 1 Architectural Hyperparameters of Models

| **Model** | **DNNs Components** | **Layer Index** | **Layer Type** | **Hidden Size / Dropout Rate** |
|---|---|---|---|---|
| RNN | RNN | 1 | hidden layer | 128 |
| | | 2 | hidden layer | 256 |
| | | 3 | hidden layer | 128 |
| | | 4 | hidden layer | 1 |
| | | 5 | Dropout layer | 0.1 |
| Cas [GCN(Dis), RNN] | GCN | 1 | hidden layer | 32 |
| | RNN | 1 | hidden layer | 64 |
| | | 2 | hidden layer | 128 |
| | | 3 | hidden layer | 256 |
| | | 4 | hidden layer | 128 |
| | | 5 | hidden layer | 64 |
| | | 6 | hidden layer | 1 |
| | | 7 | Dropout layer | 0.2 |

**Table 14** Group 1 Training Hyperparameters of Models

| | **RNN** | **Cas[GCN(Dis),RNN]** |
|---|---|---|
| **Starting Learning Rate** | 0.0005 | 0.0005 |
| **Annealing learning rate** | 0.5 | 0.5 |
| **Minimum Learning Rate** | 0.000001 | 0.000001 |
| **Learning Rate Reduction Patience** | 3 | 3 |

Continued on next page

**Table 14** Continued from Previous Page

| | **RNN** | **Cas[GCN(Dis),RNN]** |
|---|---|---|
| **Early Stopping Patience** | 10 | 10 |
| **Batch Size** | 4 | 4 |
| **Number of Epochs** | 300 | 300 |

### 5.2.4. Group 2 Model hyperparameters: Effectiveness of the Encoder-Decoder Structure

The aim of this group is to prove the effectiveness of the Encoder-Decoder Structure. The comparative models included in this group are Cas[GCN(Dis),RNN] and En[GCN(Dis),RNN] + De[RNN]. Table 15 and Table 16 illustrate the hyperparameters of each model framework. Note that the Layer index column denotes the data flows through the model component during the forward propagation process.

**Table 15** Group 2 Architectural Hyperparameters of Models

| **Model** | **DNNs Components** | **Layer Index** | **Layer Type** | **Hidden Size / Dropout Rate** |
|---|---|---|---|---|
| Cas [GCN(Dis), RNN] | Check Table 13 | | | |
| En[GCN(Dis), RNN] + De[RNN] | GCN Encoder | 1 | hidden layer | 64 |
| | | 2 | FC | 6 |
| | | 3 | hidden layer | 64 |
| | | 4 | FC | 6 |
| | | 5 | hidden layer | 64 |
| | | 6 | FC | 6 |
| | RNN Encoder | 1 | hidden layer | 64 |

Continued on next page

**Table 15** Continued from Previous Page

| Model | DNNs Components | Layer Index | Layer Type | Hidden Size / Dropout Rate |
|---|---|---|---|---|
| En[GCN(Dis), RNN] + De[RNN] | RNN Encoder | 2 | hidden layer | 128 |
| | | 3 | hidden layer | 256 |
| | | 4 | hidden layer | 128 |
| | | 5 | hidden layer | 64 |
| | | 6 | hidden layer | 1 |
| | | 7 | dropout layer | 0.1 |
| | RNN Decoder | 1 | hidden layer | 64 |
| | | 2 | hidden layer | 128 |
| | | 3 | hidden layer | 256 |
| | | 4 | hidden layer | 128 |
| | | 5 | hidden layer | 64 |
| | | 6 | hidden layer | 1 |
| | | 7 | dropout layer | 0.1 |

**Table 16** Group 2 Training Hyperparameters of Models

| | Cas[GCN(Dis),RNN] | En[GCN(Dis),RNN] + De[RNN] |
|---|---|---|
| **Starting Learning Rate** | 0.0005 | 0.0005 |
| **Annealing learning rate** | 0.5 | 0.5 |
| **Learning Rate Reduction Patience** | 3 | 3 |
| **Early Stopping Patience** | 10 | 10 |

Continued on next page

**Table 16** Continued from Previous Page

| | **Cas[GCN(Dis),RNN]** | **En[GCN(Dis),RNN] + De[RNN]** |
|---|---|---|
| **Batch Size** | 4 | 16 |
| **Number of Epochs** | 300 | 300 |

### 5.2.5. Group 3 Model hyperparameters: Impact of Different Graph Patterns

The aim of this group is to prove the effectiveness of different graph patterns. The comparative models included in this group are: En[GCN(Dis),RNN] + De[RNN], En[GCN(Dis+RT),RNN] + De[RNN], En[GCN(Dis+HT),RNN] + De[RNN], and En[GCN(Dis+HT+RT),RNN] + De[RNN]. Table 17 to Table 19 illustrate the hyperparameters of each model framework. Note that En[GCN(Dis),RNN] + De[RNN] and En[GCN(Dis+HT),RNN] + De[RNN] have the same structure, while En[GCN(Dis+RT),RNN] + De[RNN] and En[GCN(Dis+HT+RT),RNN] + De[RNN] have the same structure. These hyperparameter settings can improve the comparability of models. Besides, the Layer index column denotes the data flows through the model component during the forward propagation process.

**Table 17** Group 3 Architectural Hyperparameters of Models

| **Model** | **DNNs Components** | **Layer Index** | **Layer Type** | **Hidden Size / Dropout Rate** |
|---|---|---|---|---|
| En[GCN(Dis), RNN] + De[RNN]<br>En[GCN(Dis+HT), RNN] + De[RNN] | Check Table 15 | | | |
| En[GCN(Dis+RT), RNN] + De[RNN]<br>En[ GCN(Dis+RT+HT), RNN] + De[RNN] | GCN Encoder | 1 | hidden layer | 128 |
| | | 2 | FC | 128 |
| | | 3 | hidden layer | 256 |
| | | 4 | FC | 256 |

Continued on next page

**Table 17** Continued from Previous Page

| Model | DNNs Components | Layer Index | Layer Type | Hidden Size / Dropout Rate |
|---|---|---|---|---|
| En[GCN(Dis+RT), RNN] + De[RNN]<br><br>En[ GCN(Dis+RT+HT), RNN] + De[RNN] | GCN Encoder | 5 | hidden layer | 64 |
| | | 6 | FC | 6 |
| | RNN Enecoder | 1 | hidden layer | 64 |
| | | 2 | hidden layer | 128 |
| | | 3 | hidden layer | 256 |
| | | 4 | hidden layer | 512 |
| | | 5 | hidden layer | 256 |
| | | 6 | hidden layer | 64 |
| | | 7 | hidden layer | 1 |
| | | 8 | dropout layer | 0.1 |
| | RNN Decoder | 1 | hidden layer | 64 |
| | | 2 | hidden layer | 128 |
| | | 3 | hidden layer | 256 |
| | | 4 | hidden layer | 512 |
| | | 5 | hidden layer | 256 |
| | | 6 | hidden layer | 64 |
| | | 7 | hidden layer | 1 |
| | | 8 | dropout layer | 0.1 |

**Table 18** Group 3 Training Hyperparameters of Models Part A

| | **En[GCN(Dis),RNN] + De[RNN]** | **En[GCN(Dis+HT),RNN] + De[RNN]** |
|---|---|---|
| **Starting Learning Rate** | 0.0005 | 0.0005 |
| **Annealing learning rate** | 0.5 | 0.5 |
| **Learning Rate Reduction Patience** | 3 | 3 |
| **Early Stopping Patience** | 10 | 10 |
| **Batch Size** | 16 | 16 |
| **Number of Epochs** | 300 | 300 |

**Table 19** Group 3 Training Hyperparameters of Models Part B

| | **En[GCN(Dis+RT),RNN] + De[RNN]** | **En[GCN(Dis+HT+RT), RNN] + De[RNN]** |
|---|---|---|
| **Starting Learning Rate** | 0.0005 | 0.0005 |
| **Annealing learning rate** | 0.5 | 0.5 |
| **Learning Rate Reduction Patience** | 3 | 3 |
| **Early Stopping Patience** | 10 | 10 |
| **Batch Size** | 4 | 4 |
| **Number of Epochs** | 300 | 300 |

### 5.2.6. Group 4 Model hyperparameters: Comparison of RNN Variants

The aim of this group is to investigate the effectiveness of RNN variants. The comparative models included in this group are En[GCN(Dis+HT+RT),RNN] + De[RNN], En[GCN(Dis+HT +RT), GRU] + De[GRU], and En[GCN(Dis+HT+RT),LSTM] + De[LSTM]. Table 17 to Table 19 illustrate the hyperparameters of each model framework. These models have a similar architecture. These hyperparameter settings can improve the comparability of models.

**Table 20** Group 4 Architectural Hyperparameters of Models

| Model | DNNs Components | Layer Index | Layer Type | Hidden Size / Dropout Rate |
|---|---|---|---|---|
| En[ GCN(Dis+RT+HT), RNN] + De[RNN]<br>En[ GCN(Dis+RT+HT), LSTM] + De[LSTM]<br>En[ GCN(Dis+RT+HT), GRU] + De[GRU] | GCN Encoder | 1 | hidden layer | 128 |
| | | 2 | FC | 128 |
| | | 3 | hidden layer | 256 |
| | | 4 | FC | 256 |
| | | 5 | hidden layer | 64 |
| | | 6 | FC | 6 |
| | RNN Encoder<br>LSTM Encoder<br>GRU Encoder | 1 | hidden layer | 64 |
| | | 2 | hidden layer | 128 |
| | | 3 | hidden layer | 256 |
| | | 4 | hidden layer | 512 |
| | | 5 | hidden layer | 256 |
| | | 6 | hidden layer | 64 |
| | | 7 | hidden layer | 1 |
| | | 8 | dropout layer | 0.1 |
| | RNN Decoder<br>LSTM Decoder<br>GRU Decoder | 1 | hidden layer | 64 |
| | | 2 | hidden layer | 128 |
| | | 3 | hidden layer | 256 |
| | | 4 | hidden layer | 512 |
| | | 5 | hidden layer | 256 |
| | | 6 | hidden layer | 256 |

Continued on next page

**Table 20** Continued from Previous Page

| Model | DNNs Components | Layer Index | Layer Type | Hidden Size / Dropout Rate |
|---|---|---|---|---|
| | RNN Decoder | 7 | hidden layer | 1 |
| | LSTM Decoder<br>GRU Decoder | 8 | dropout layer | 0.1 |

**Table 21** Group 4 Training Hyperparameters of Models Part A

| | En[GCN(Dis+HT+RT),RNN]+ De[RNN] | En[GCN(Dis+HT+RT),LSTM]+ De[LSTM] |
|---|---|---|
| **Starting Learning Rate** | 0.0005 | 0.0005 |
| **Annealing learning rate** | 0.5 | 0.5 |
| **Learning Rate Reduction Patience** | 3 | 3 |
| **Early Stopping Patience** | 10 | 10 |
| **Batch Size** | 16 | 16 |
| **Number of Epochs** | 300 | 300 |

**Table 22** Group 4 Training Hyperparameters of Models Part B

| | En[GCN(Dis+HT+RT),GRU] + De[GRU] |
|---|---|
| **Starting Learning Rate** | 0.0005 |
| **Annealing learning rate** | 0.5 |
| **Learning Rate Reduction Patience** | 3 |
| **Early Stopping Patience** | 10 |
| **Batch Size** | 4 |

Continued on next page

**Table 22** Continued from Previous Page

| | **En[GCN(Dis+HT+RT),GRU] + De[GRU]** |
|---|---|
| **Number of Epochs** | 300 |

### 5.2.7. Group 5 Model hyperparameters: Model Robustness Check

The aim of this group is to evaluate the robustness of the proposed model framework, En[GCN (Dis+HT+RT),RNN] + De[RNN], in the context of the COVID-19 pandemic scenario. The comparison model, En[GCN(Dis+HT),RNN] + De[RNN], is set as a baseline variant, which adopts a static graph structure. Models are evaluated over three temporal phases: before the pandemic, during the pandemic, and during the recovery period. The hyperparameter settings are consistent with the other groups, shown in Table 17 to Table 20.

## 5.3. Experiment Result and Analysis

In this section, the effectiveness of the GCN model, encoder-decoder architecture, graph variants, RNN variants, and model robustness are discussed. Namely, they are discussed in three aspects: global performance, time step-wise forecasting accuracy, and spatial error distribution. Note that MAPE is computed only for instances where the ground truth is non-zero to avoid division errors. Besides, the relatively low ground truth in short-term traffic volume prediction can lead to an illusion of high MAPE and SMAPE. Therefore, MSE and MAE are employed at the node level to quantify the prediction errors directly. In contrast, the network-level accuracy is presented with MAPE, SMAPE, and $R^2$ since the scale of total ground truth across the time steps is unknown so that the relative accuracy can better illustrate the model performance.

1. Global Performance: The predictive accuracy of models is compared using node-level and network-level evaluation metrics tables, which focus on the overall performance across all time steps of all data samples.

2. Time Step-Wise Forecasting Accuracy: The average model performance is evaluated at each time step to reveal the temporal progression of prediction accuracy at the network level.

3. Spatial Error Distribution: The spatial error distribution is evaluated through node-level heatmaps at the first and last time steps, which highlights the network local difference and reveals the model's robustness at later stages of the prediction horizon.

### 5.3.1. Group 1: Effectiveness of the GCN Model

Table 23 reports the evaluation metrics of group 1 models at node-level. The result illustrates that the RNN model has slightly lower MSE and MAE than the Cas[GCN(Dis),RNN] model, meaning that the RNN model shows marginally superior performance in predicting the traffic flow in most nodes. However, its performance falls much behind the Cas[GCN(Dis),RNN] model in terms of the traffic volume prediction at the network-level, shown in Table 24. More precisely, the Cas[GCN(Dis),RNN] model can explain the data variance more accurately for its higher $R^2$, and lower MAPE and SMAPE. This is attributed to its better prediction capability for high-traffic nodes, which play a dominant role in overall network traffic volume. Accordingly, it can be concluded that the Cas[GCN(Dis),RNN] model has better global performance.

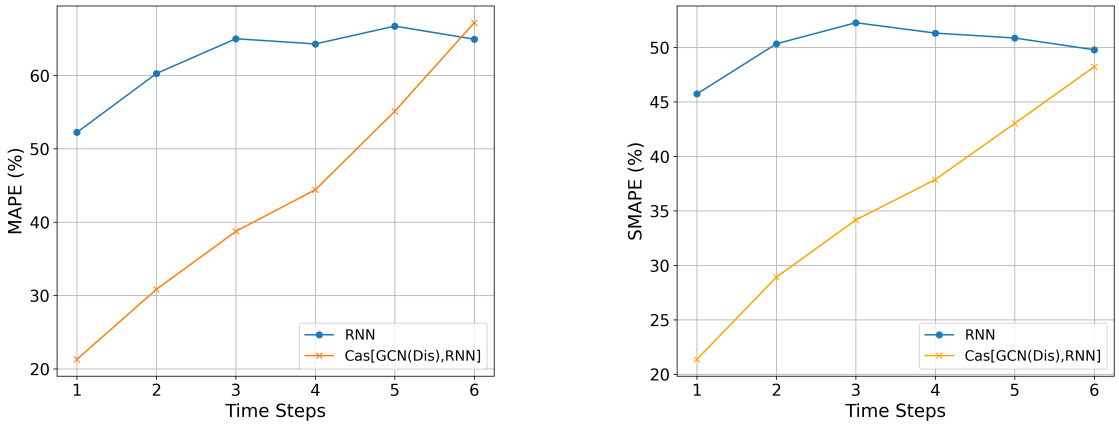
**Table 23** Group 1 Evaluation Metrics at Node-Level

| | **MSE** | **MAE** |
|---|---|---|
| **RNN** | 157.764 | 5.587 |
| **Cas[GCN(Dis),RNN]** | 162.281 | 5.829 |

**Table 24** Group 1 Evaluation Metrics at Network-Level

| | $R^2$ | **MAPE** | **SMAPE** |
|---|---|---|---|
| **RNN** | 0.404 | 62.23% | 50.05% |
| **Cas[GCN(Dis),RNN]** | 0.621 | 42.93% | 35.59% |

Figure 22a and Figure 22b present the time step-wise loss of both models. Both models show a decrease in model performance from the first time step to the last time step, which is reasonable for weaker correlations between the later time steps to the input historical sequence. The results show that the RNN model has relatively high error across all time steps, with SMAPE, for instance, starting from approximately 45% to 50%. However, its performance remains stable at the later time steps, indicating a certain degree of resilience. In contrast, Cas[GCN(Dis),RNN] performs much better than RNN at most time steps. Moreover, it has much superior performance during the first four time steps, meaning that the model has better short-term prediction performance despite a sharp drop in performance, approximately 46% for MAPE and 26% for SMAPE. Accordingly, it can be concluded that Cas[GCN(Dis),RNN] demonstrates superior performance across time steps.




(a) Time Step-Wise MAPE of Group 1

(b) Time Step-Wise SMAPE of Group 1

**Figure 22** Comparison of Time Step-Wise MAPE and SMAPE for Group 1

Figure 23 pictures the node-level model performance at the first and last time step. The Cas[GCN(Dis),RNN] gives better predictions at the first time step, particularly for most arterial roads. However, as aforementioned, its performance deteriorates significantly over time steps, falling behind the performance of the RNN models at the last time step. The poor predictions are primarily concentrated in urban areas, in which the road topology is complicated. Therefore, the poor capability can be due to the inadequacy of the distance-based static adjacency matrix in representing complex topology, so the model is potentially misled.

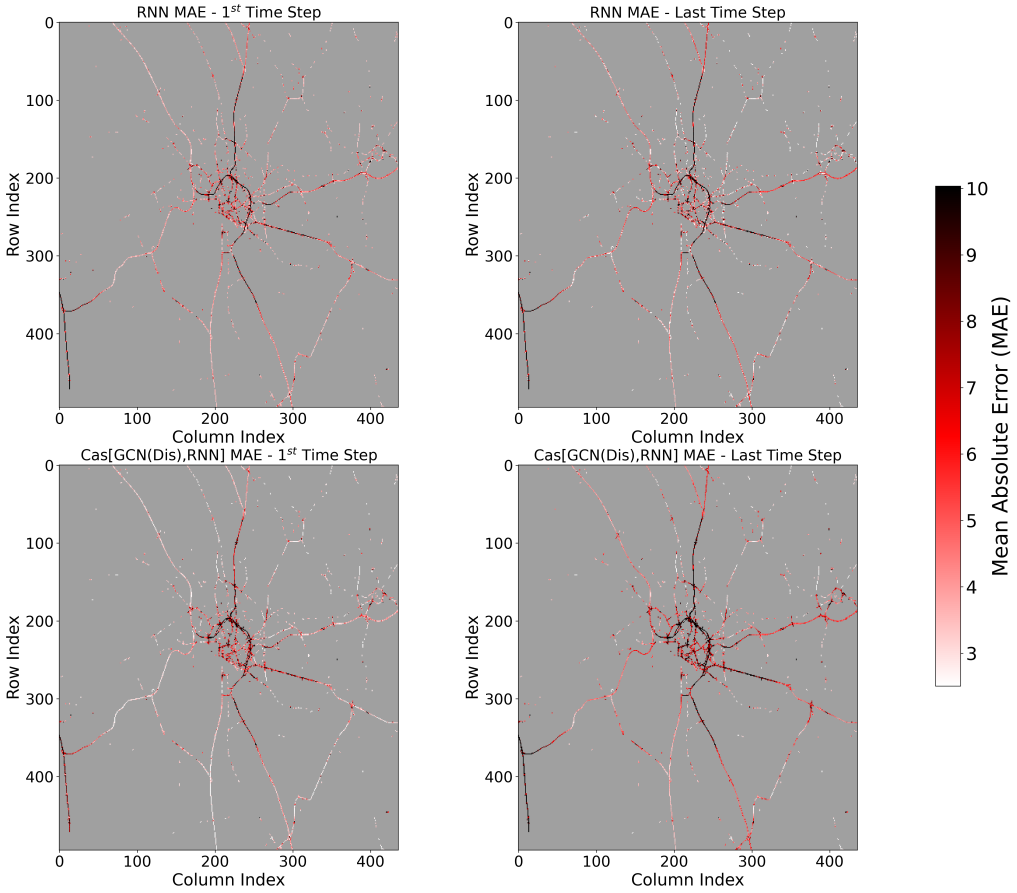



**Figure 23** Spatial Error Distribution of Group 1

Another potential issue can be error accumulation. More precisely, the inherent temporal structure within the data is disrupted after the spatial aggregation process in the GCN model, leading to a distorted temporal representation. The processed data are subsequently fed into the RNN model, in which the hidden state can propagate and even amplify errors across time steps. Accordingly, a sharp drop in model performance across time steps can be observed.

Eventually, it can be concluded that the inclusion of the GCN model is crucial to enhance the model performance by additionally investigating the spatial information. However, the distance-based static adjacency matrix is too simplistic to represent the urban network structure. The model performance is also limited at later future time steps, meaning a weak temporal pattern extraction ability. Accordingly, the graph structure should be more sophisticated to capture intricate topology characteristics to improve model performance. Additional components can be included to enhance the model's capability of learning temporal dependencies.

### 5.3.2. Group 2: Effectiveness of the Encoder-Decoder Structure

Table 25 reports the evaluation metrics of group 2 models at the node-level. The result presents a distinct accuracy improvement if the cascaded structure is replaced by an encoder-decoder structure for the lower MSE and MAE. These results indicate that the encoder-decoder structure not only produces superior predictions across the data samples but also can effectively decrease the number of large errors. Additionally, this model can explain the network-level data variance more accurately for its higher $R^2$ and lower MAPE and SMAPE, shown in Table 26. Accordingly, it can be concluded that the En[GCN(Dis),RNN] + De[RNN] model has better global performance.

**Table 25** Group 2 Evaluation Metrics at Node-Level

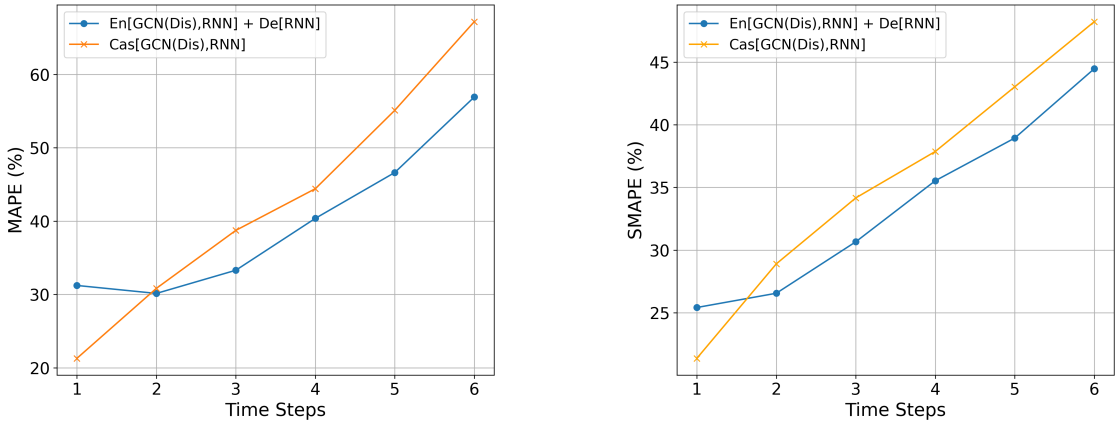
| | **MSE** | **MAE** |
|---|---|---|
| **Cas[GCN(Dis),RNN]** | 162.281 | 5.829 |
| **En[GCN(Dis),RNN] + De[RNN]** | 147.521 | 5.076 |

**Table 26** Group 2 Evaluation Metrics at Network-Level

| | $R^2$ | **MAPE** | **SMAPE** |
|---|---|---|---|
| **Cas[GCN(Dis),RNN]** | 0.621 | 42.93% | 35.59% |
| **En[GCN(Dis),RNN] + De[RNN]** | 0.656 | 38.60% | 32.83% |

Figure 24a and Figure 24b present the time step-wise loss of both models. A clear decrease in model performance can be observed from the first step to the last step, which is reasonable for weaker correlations between the later time steps to the input historical sequence. Both figures present a better model performance of the Cas[GCN(Dis),RNN] model

at the first time step with approximately 5% higher in MAPE and SMAPE compared with the performance of the En[GCN(Dis),RNN] + De[RNN] model. This can be attributed to the fact that the spatio-temporal representations are not effectively fused by taking the average, meaning that the influence of two dependencies can vary across the time steps. However, the En[GCN(Dis),RNN] + De[RNN] model can produce predictions with less loss across the rest of the time steps. In other words, the En[GCN(Dis),RNN] + De[RNN] model has about 20% performance drop, which is lower than the 35% performance drop from the Cas[GCN(Dis),RNN] model, meaning that the encoder-decoder structure has better resilience to increasing prediction horizons.




(a) Time Step-Wise MAPE of Group 2

(b) Time Step-Wise SMAPE of Group 2

**Figure 24** Comparison of Time Step-Wise MAPE and SMAPE for Group 2

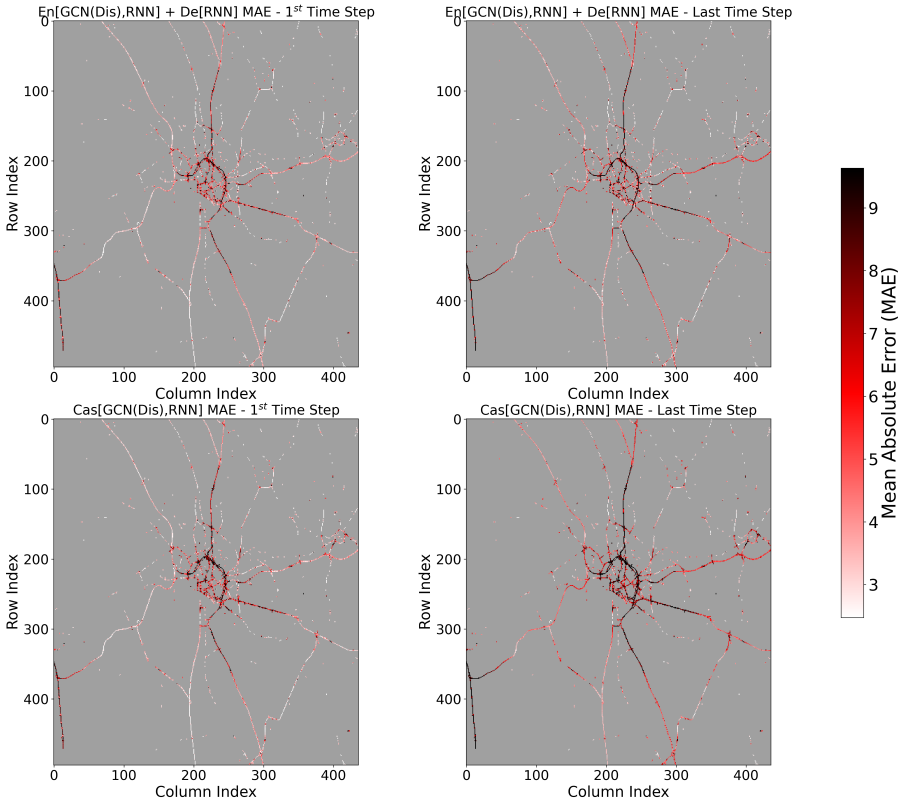



**Figure 25** Spatial Error Distribution of Group 2

From the spatial error distribution heat map (Figure 25), a clear error reduction of node-level loss can be found around urban areas at both time steps. This result proves the effectiveness of the encoder-decoder structure in enhancing the model capability of learning temporal patterns inside the data sequence by successfully fusing spatial and temporal dependencies.

Eventually, it can be concluded that the encoder-decoder structure can be more efficient than the cascaded structure for its superior ability to fuse spatial and temporal dependencies and further explore the temporal dependencies by adding the RNN learning component of hidden traffic temporal patterns.

### 5.3.3. Group 3: Impact of Different Graph Constructions

Table 27 and Table 28 present the evaluation metrics at the node-level and network-level, respectively. Among these graph variations, the model employs a hybrid graph composed of Dis, RT, and HT, which achieves the most superior performance. This is followed by the model with the graph hybridized from Dis+ RT and Dis + HT. Additionally, En[GCN(Dis),RNN] + De[RNN] yields the lowest performance among all other models. Interestingly, the experiment results illustrate that the RT component contributes the most to the model performance, while only a slight improvement of HT can be observed. The difference can be attributed to the aim of this task. Short-term traffic volume prediction is highly correlated with current temporal dynamics, which can be efficiently captured by RT. In contrast, the distinct spatio-temporal characteristics can be diluted or outdated in HT since they are derived from the long-range intervals in the preceding time window of interest. Accordingly, it can be concluded that the En[GCN(Dis+HT+RT),RNN] + De[RNN] demonstrates the best global performance.

**Table 27** Group 3 Evaluation Metrics at Node-Level

| | **MSE** | **MAE** |
|---|---|---|
| **En[GCN(Dis),RNN] + De[RNN]** | 147.521 | 5.076 |
| **En[GCN(Dis+RT),RNN] + De[RNN]** | 144.615 | 4.836 |
| **En[GCN(Dis+HT),RNN] + De[RNN]** | 147.072 | 5.076 |
| **En[GCN(Dis+HT+RT),RNN] + De[RNN]** | 144.023 | 4.819 |

**Table 28** Group 3 Evaluation Metrics at Network-Level

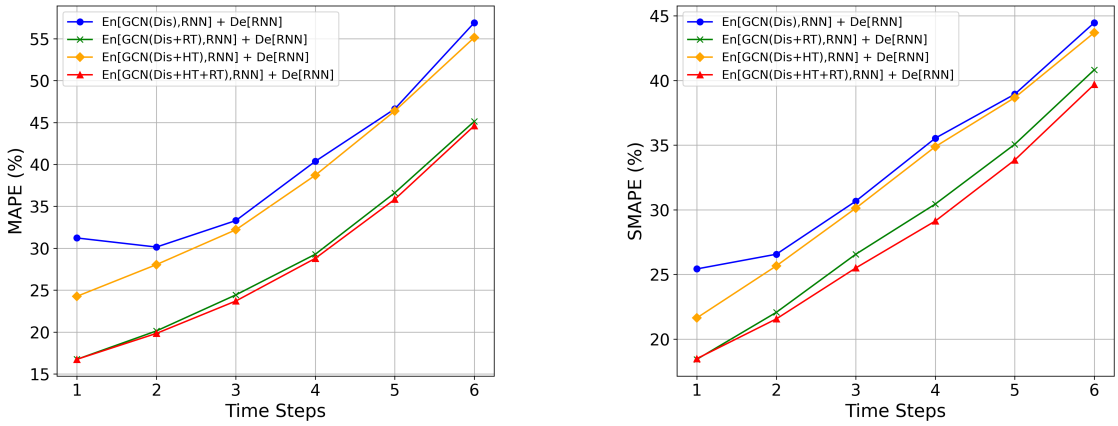
| | $R^2$ | **MAPE** | **SMAPE** |
|---|---|---|---|
| **En[GCN(Dis),RNN] + De[RNN]** | 0.656 | 38.60% | 32.83% |
| **En[GCN(Dis+RT),RNN] + De[RNN]** | 0.684 | 28.73% | 28.90% |

Continued on next page

**Table 28** Continued from Previous Page

| | $R^2$ | **MAPE** | **SMAPE** |
|---|---|---|---|
| **En[GCN(Dis+HT),RNN] + De[RNN]** | 0.657 | 37.46% | 32.45% |
| **En[GCN(Dis+HT+RT),RNN] + De[RNN]** | 0.693 | 28.26% | 28.04% |

Figure 26a and Figure 26b present the time step-wise loss of models. All models show a similar decreasing trend and magnitude. Those models incorporated with the RT component have similar performance with approximately a minimum of 15% in MAPE and SMAPE. A 25% performance drop of those models with RT component can be observed, resulting in an approximately maximum of 45% in MPAE and 40% in SMAPE. Nevertheless, the HT component still offers a slight improvement, meaning that additional spatial-temporal patterns can be learned from this graph.




(a) Time Step-Wise MAPE of Group 3

(b) Time Step-Wise SMAPE of Group 3

**Figure 26** Comparison of Time Step-Wise MAPE and SMAPE for Group 3

From the spatial error distribution heat maps (Figure 27 and Figure 28), four models exhibit similar performance on the node-level prediction. Specifically, they all have relatively poor predictions in urban areas but better predictions in the arterial ways. The differences in the results are hard to distinguish because of their similar MSE and MAE at the node-level.

Eventually, it can be concluded that the En[GCN(Dis+HT+RT),RNN] + De[RNN] model, which incorporates a distance-based static graph, a sequence-based static graph, and sequence-based dynamic graphs, has the best prediction performance.

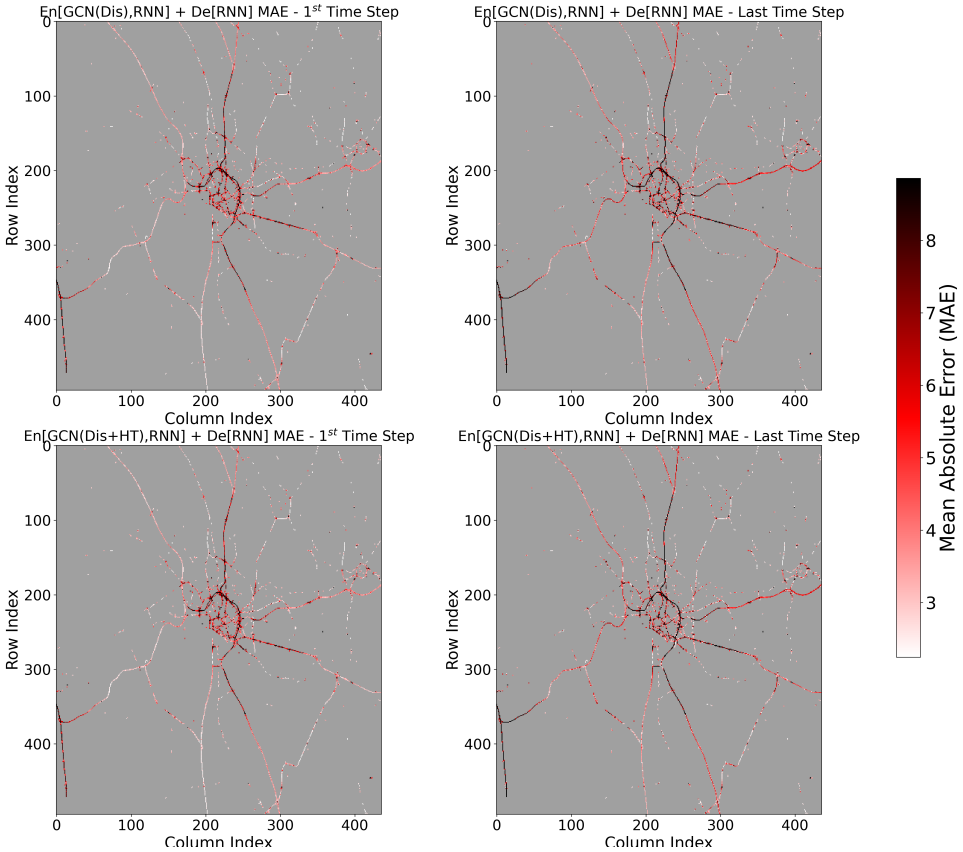



**Figure 27** Spatial Error Distribution of Group 3 Part A

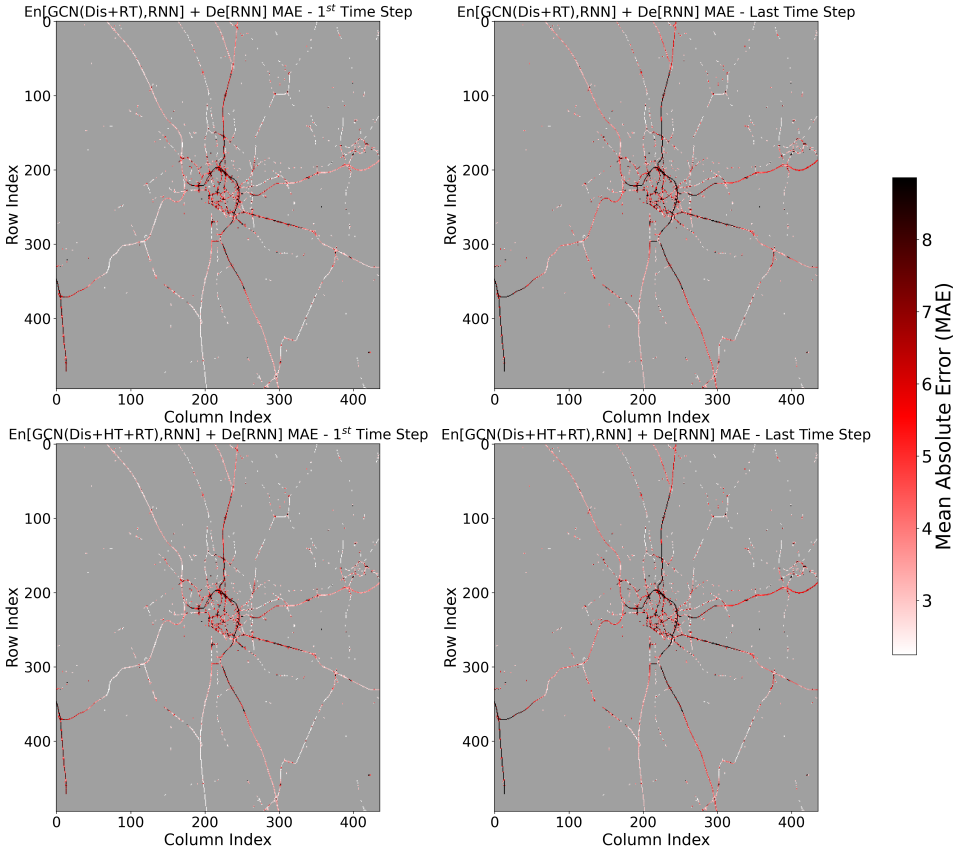



**Figure 28** Spatial Error Distribution of Group 3 Part B

### 5.3.4. Group 4: Comparison of RNN Variants

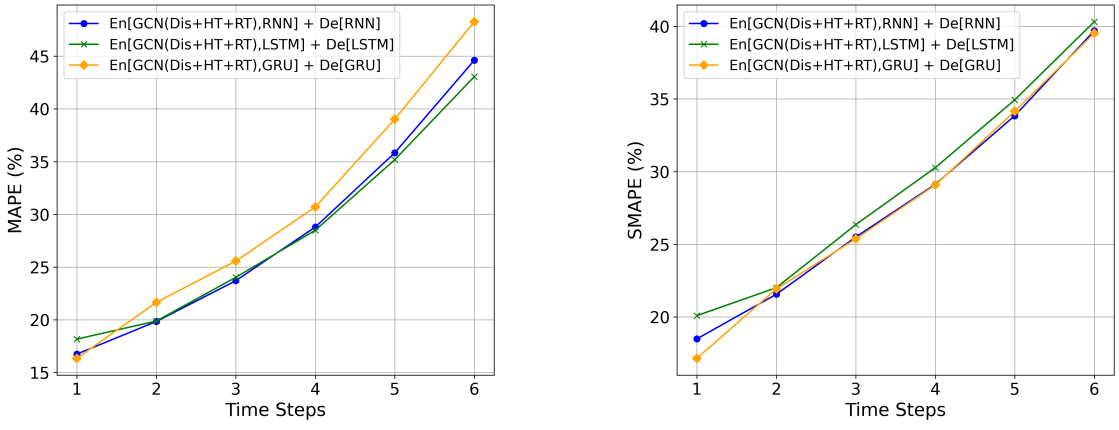
Table 29 and Table 30 present the evaluation metrics at the node-level and network-level, respectively. The result shows that all three variants have similar performance in terms of the node-level and the network-level. Their similar performance is further proved in Figure 29a and Figure 29b. All three models have nearly identical trends of a decreasing pattern in MAPE and SMAPE across the time steps. In addition to this, they even have similar minimum and maximum errors. Moreover, the nearly identical pattern of spatial error distribution is shown in Figure 30 and 31. Eventually, it can be concluded that these models have nearly identical performance. However, the En[GCN(Dis+HT+RT),RNN] + De[RNN] model is regarded as the best model among them for its lower model complexity and computational cost.

**Table 29** Group 4 Evaluation Metrics at Node-Level

| | **MSE** | **MAE** |
|---|---|---|
| **En[GCN(Dis+HT+RT),RNN] + De[RNN]** | 144.023 | 4.819 |
| **En[GCN(Dis+HT+RT),GRU] + De[GRU]** | 143.599 | 4.870 |
| **En[GCN(Dis+HT+RT),LSTM] + De[LSTM]** | 146.303 | 4.870 |

**Table 30** Group 4 Evaluation Metrics at Network-Level

| | $R^2$ | **MAPE** | **SMAPE** |
|---|---|---|---|
| **En[GCN(Dis+HT+RT),RNN] + De[RNN]** | 0.693 | 28.26% | 28.04% |
| **En[GCN(Dis+HT+RT),GRU] + De[GRU]** | 0.709 | 30.26% | 27.87% |
| **En[GCN(Dis+HT+RT),LSTM] + De[LSTM]** | 0.6817 | 28.13% | 29.00% |




(a) Time Step-Wise MAPE of Group 4

(b) Time Step-Wise SMAPE of Group 4

**Figure 29** Comparison of Time Step-Wise MAPE and SMAPE for Group 4

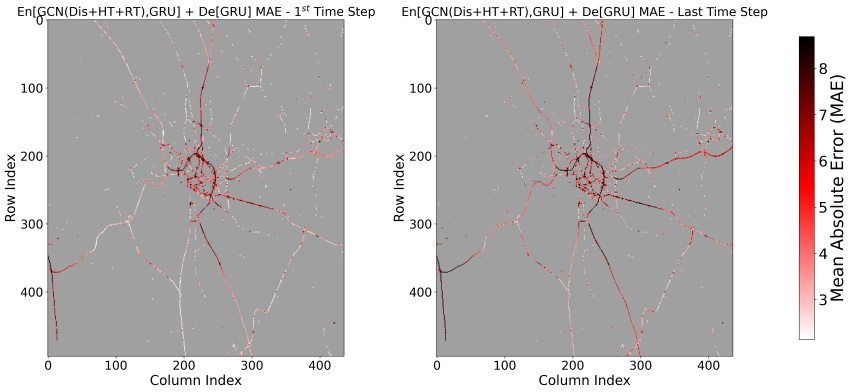



**Figure 30** Spatial Error Distribution of Group 4 Part A

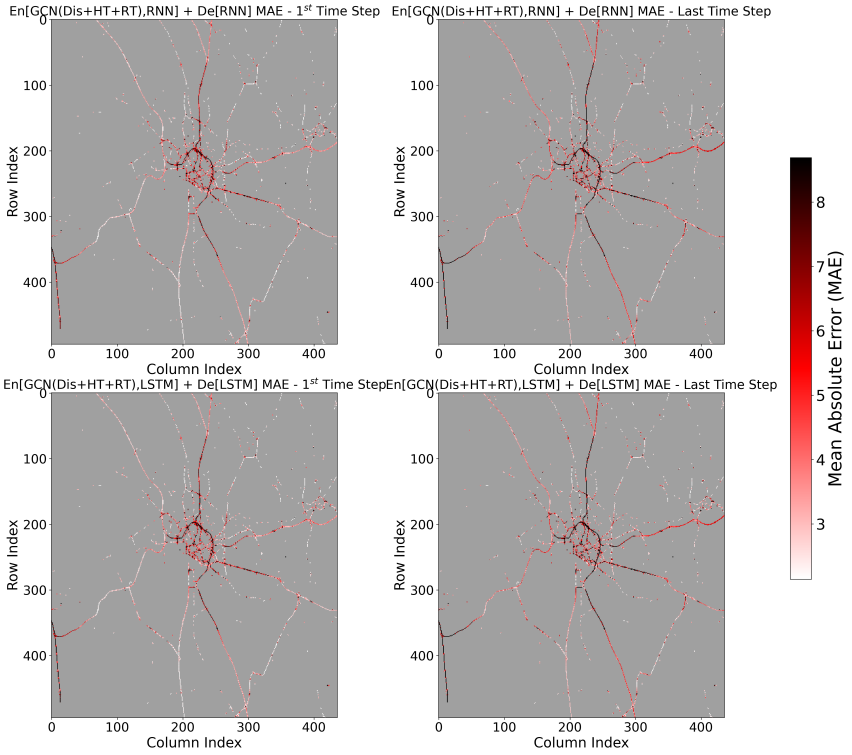



**Figure 31** Spatial Error Distribution of Group 4 Part B

### 5.3.5. Group 5: Model Robustness Check

Table 31 and Table 32 present the evaluation metrics at the node-level and network-level, respectively. In section 5.3.3, it has been discussed that the BE{En[GCN(Dis+HT+RT),RNN] + De[RNN]} model demonstrates superior performance than BE{En[GCN(Dis+HT),RNN] + De[RNN]} for its critical component RT. This relative improvement is still valid during the pandemic and recovery period. However, an obvious increase in MAPE and SMAPE is detected at the network level. The potential reason is that the graph structure was constructed based on the traffic volume data from March 2019. The graph structure, such as connectivity, can be changed during different time periods due to external factors, such as the pandemic quarantine zones. During the pandemic event and the recovery period, the performance gap between the two comparison models is significantly larger than that observed before the pandemic event. A potential reason can lie in the considerable variability in traffic patterns during these periods, which limits the model's performance as the long-term trend captured by HT

can mislead the model under dynamic conditions. Nevertheless, the relative improvement still demonstrates the effectiveness of sequence-based dynamic graphs structured and constructed by DTW.

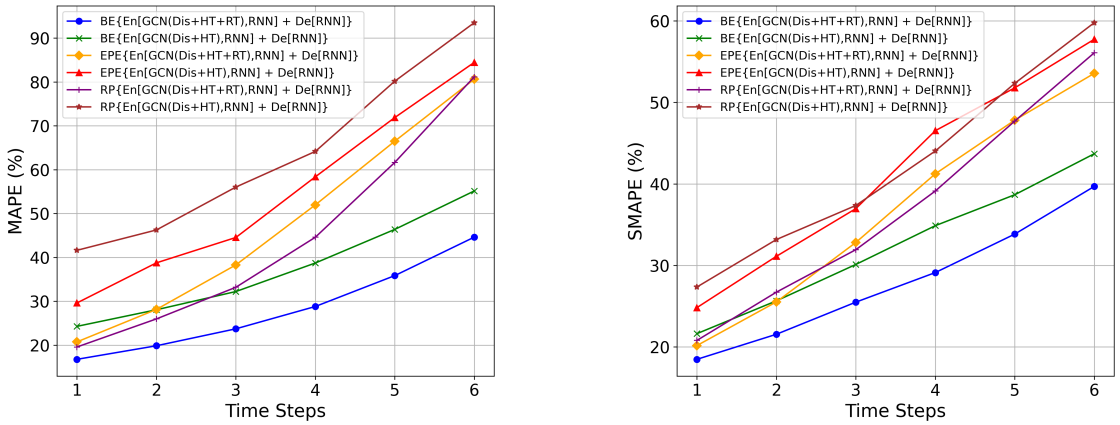
**Table 31** Group 5 Evaluation Metrics at Node-Level

| | **MSE** | **MAE** |
|---|---|---|
| **BE{En[GCN(Dis+HT+RT),RNN] + De[RNN]}** | 144.023 | 4.819 |
| **BE{En[GCN(Dis+HT),RNN] + De[RNN]}** | 147.072 | 5.076 |
| **EPE {En[GCN(Dis+HT+RT),RNN] + De[RNN]}** | 58.962 | 2.801 |
| **EPE{En[GCN(Dis+HT),RNN] + De[RNN]}** | 60.321 | 2.918 |
| **RP{En[GCN(Dis+HT+RT),RNN] + De[RNN]}** | 42.604 | 2.779 |
| **RP{En[GCN(Dis+HT),RNN] + De[RNN]}** | 44.609 | 3.000 |

**Table 32** Group 5 Evaluation Metrics at Network-Level

| | $R^2$ | **MAPE** | **SMAPE** |
|---|---|---|---|
| **BE{En[GCN(Dis+HT+RT),RNN] + De[RNN]}** | 0.693 | 28.26% | 28.04% |
| **BE{En[GCN(Dis+HT),RNN] + De[RNN]}** | 0.657 | 37.46% | 32.45% |
| **EPE {En[GCN(Dis+HT+RT), RNN] + De[RNN]}** | 0.652 | 47.70% | 36.86% |
| **EPE{En[GCN(Dis+HT),RNN] + De[RNN]}** | 0.550 | 54.60% | 41.50% |
| **RP{En[GCN(Dis+HT+RT),RNN] + De[RNN]}** | 0.657 | 44.35% | 37.07% |
| **RP{En[GCN(Dis+HT),RNN] + De[RNN]}** | 0.510 | 63.62% | 42.34% |

Figure 32a and Figure 32b present the time step-wise loss of models. It is clearly shown in the figures that those models incorporating the RT component consistently keep their superior performance despite the performance drop during the pandemic and recovery period. Moreover, the increase in performance remains around 10% across the prediction horizon.




(a) Time Step-Wise MAPE of Group 5

(b) Time Step-Wise SMAPE of Group 5

**Figure 32** Comparison of Time Step-Wise MAPE and SMAPE for Group 5

Figure 33 to Figure 35 present the spatial error distribution heat maps in three different research periods. In each period, the two models exhibit similar performance on the node-level prediction. Particularly, large errors are concentrated around urban areas, but accurate predictions are made in arterial ways. These results are reasonable in that the differences are hard to distinguish because they have similar MSE and MAE at the node level.

Eventually, it can be concluded that the proposed framework has a certain robustness to the public emergent event since it can still retain its relative increase in performance.

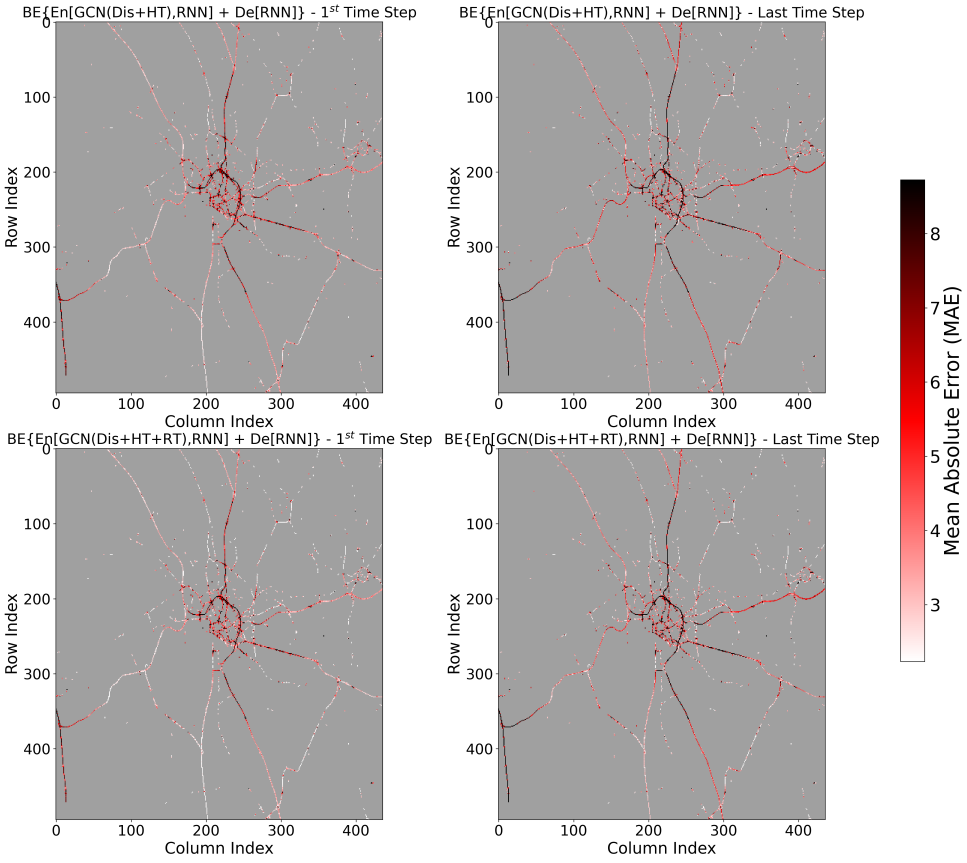



**Figure 33** Spatial Error Distribution of Group 5 (Before Pandemic)

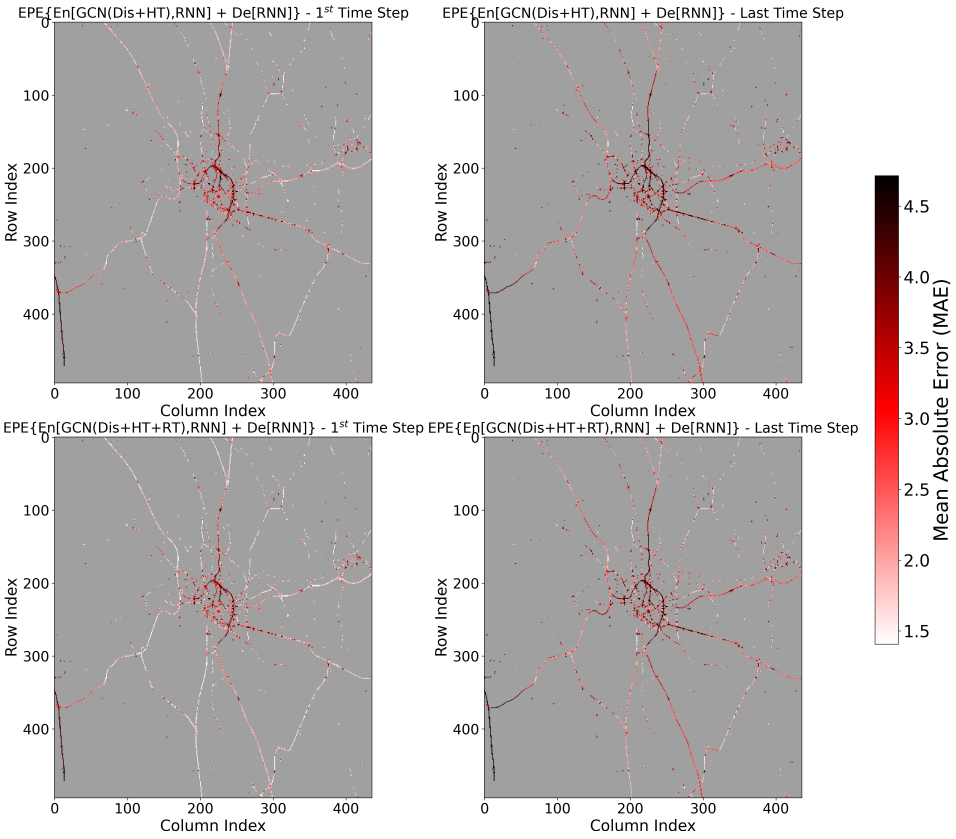



**Figure 34** Spatial Error Distribution of Group 5 Part B (During Pandemic)

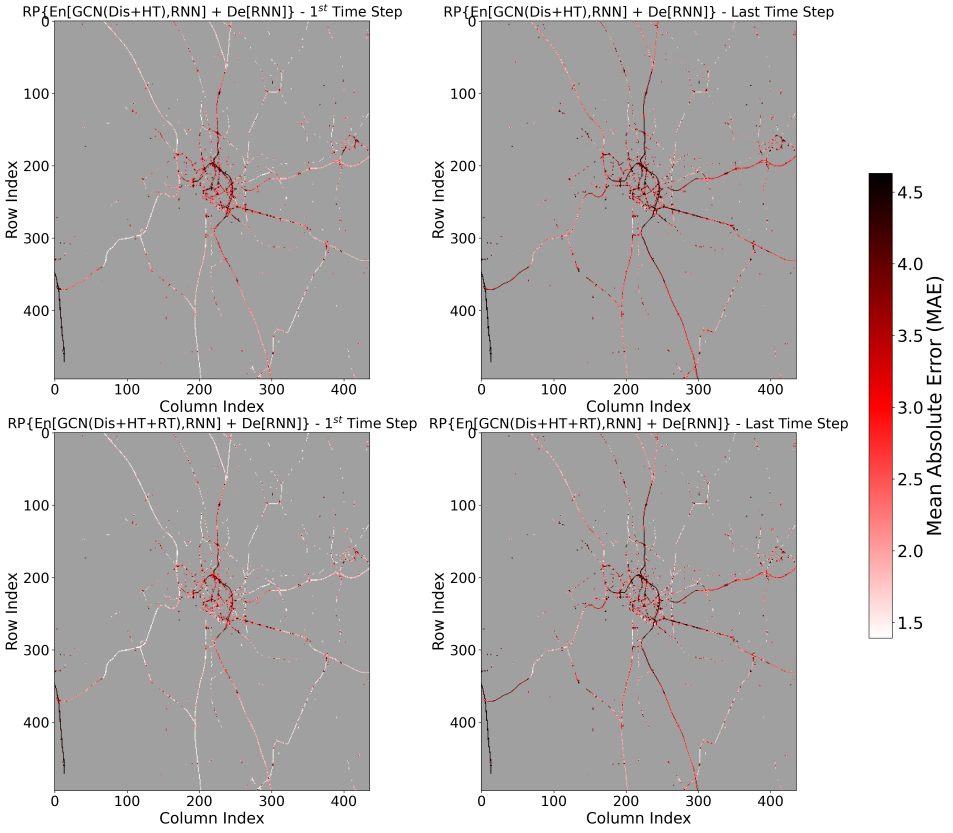



**Figure 35** Spatial Error Distribution of Group 5 Part B (Recovery Period)

# 6. Conclusion

## 6.1. Research Contributions

This thesis proposes a novel ST-GCN model with a hybrid dynamic adjacency matrix, which is constructed based on Gaussian distance weights and DTW node similarity. Specifically, the proposed model framework fused the distance-based static adjacency matrix, sequence-based static adjacency matrix, and sequence-based dynamic adjacency matrix as a hybridized graph that contains physical topology, long-term traffic state pattern, and short-term traffic state characteristics. Furthermore, an encoder-decoder structure is designed to enhance the model's capability of learning hidden temporal dynamics in the data sequence. In short, the data samples are passing through GCN and RNN encoders, which can capture the spatial and temporal dependencies, respectively. Then, their outputs are fused and fed into an RNN decoder to further extract the temporal relationships in the hidden representations.

Following the principle of dual-branch ablation, a carefully designed experiment is executed using a real-world dataset to evaluate the proposed model framework. Namely, the necessity and effectiveness of four key components are verified, including the inclusion of the GCN model, the encoder-decoder architecture, the different graph patterns, and the variants of RNN models. In addition to this, the model's robustness is further tested during the COVID-19 pandemic and recovery period. The performances of these models are evaluated at node-level and network-level with evaluation metrics, including MSE, MAE, $R^2$, MAPE, and SMAPE. Particularly, the MAPE and SMAPE can give an illusion of relatively high output error with low ground truth values due to the nature of the short-term prediction task. Therefore, the node-level errors are illustrated with MSE and MAE as the statistics analysis has presented the data scale, while the $R^2$, MAPE, and SMAPE are employed at network-level to compare the relative loss within the entire network.

The performances of models are evaluated through three aspects, including global performance, time step-wise forecasting accuracy, and spatial error distribution. Therein, the average performance across all time steps of all data samples for each model is discussed. Moreover, this thesis localizes prediction errors across individual time steps and explores error differences in their spatial distribution. Eventually, the experiment results validate the effectiveness of each component within the proposed En[GCN(Dis+HT+RT),RNN] + De[RNN] model. In particular, the inclusion of the GCN model significantly improves the model's prediction ability by capturing spatial dependencies through Dis. Compared with Cas, the complicated structure of ED can prove its effectiveness in extracting hidden temporal patterns within the input data sequence. HT and RT play a role in extracting the long-term and short-term traffic volume variation trend. Furthermore, RT contributes to much more performance improvement than HT, owing to the nature of short-term prediction tasks, which reflect real-time

traffic scenarios and are more beneficial than relying on historical long-term trends.

The proposed model framework is capable of handling traffic volume prediction in urban areas under a variety of circumstances. The following three applications are discussed to support real-world transport management:

1. Regional traffic accident response system: To detect sudden traffic accidents, [44] introduced Dijkstra and Depth-First Search to effectively extract the correlation between the traffic network accident sections and the nearby area spatial characteristics. In this thesis, the proposed model framework with additional sequence-based region similarity can also provide transport organizers with an effective alternative for emergency response planning. The DTW algorithm enables the real-time traffic state detection in upstream and downstream road segments so that the model can deliver an early warning by identifying the potential accident regions, which supports traffic management measures.

2. Dynamic signal control system: [45] proposed a regional traffic signal control method based on the back-pressure algorithm and DNNs to construct a multi-section traffic signal coordination control model, which is capable of optimizing vehicle arrival times at an intersection so that the congestion rate can be reduced. In this thesis, the proposed ST-GCN model captures spatio-temporal dependencies within the entire traffic network, making it possible to optimize the signal timing more precisely across the entire network. For example, it can shorten the green light duration on trunk roads with low traffic volume while extending the green light duration on branch roads. Accordingly, the travel efficiency of vehicles can be enhanced by replacing the fixed timing strategy.

3. Real-time routing optimization for public transportation: Public transportation, especially buses, can reduce the traffic efficiency within a road segment due to their large size and limited speed. [46] proposed a restrictive inheritance-based heuristic algorithm to solve the real-time vehicle routing problem. However, this algorithm struggles with its limited empirical parameters and execution time cost. As an improvement, the real-time routing strategies can be produced in a timely and precise manner based on the surrounding traffic state evaluation from the proposed model framework so that operation efficiency can be improved.

## 6.2. Research Limitations

1. During the node filtering process, the nodes are directly screened by the gray-scale map and the range of the traffic flow, resulting in several scattered irrelevant nodes left in the graph. Some of the critical nodes, such as those around city centers, are removed, which limits the model ability to predict complicated urban network areas.

2. When determining the edge weights, the $\sigma$ and temperature parameters $\tau$ are selected empirically, which can introduce a subjective bias to the adjacency matrices.

3. Graph structures are fused by averaging. This process may fail to fully represent the traffic dynamic states since the influence of these graphs can vary across time steps.

4. The model performance can reduce approximately 10%, although it can retain its relative increase compared with the base model. This indicates insufficient generalization capability in various data patterns.

## 6.3. Future Research Potential Directions

1. The node filtering process can be updated to preserve the complex graph structure, including the arterial roads and urban center networks, while eliminating those scattered nodes.

2. Instead of fusing the graph structure by taking the mean, further research can incorporate learnable fusion weights parameters to enable the model to adaptively balance the contributions of different graph structures.

3. Learnable fusion parameters can be introduced into the fusion process of encoder outputs to flexibly adapt to the dynamic influence across time steps.

4. In future work, transfer learning can be introduced to apply the model to different cities or different time periods within a single city to enhance the model generalization capability.

# Bibliography


[1] W. Zhang, R. Yao, Y. Yuan, X. Du, L. Wang, and F. Sun, “A traffic-weather generative adversarial network for traffic flow prediction for road networks under bad weather,” *Engineering Applications of Artificial Intelligence*, vol. 137, no. 109125, 2024.

[2] Y. Xu, X. Cai, E. Wang, W. Liu, Y. Yang, and F. Yang, “Dynamic traffic correlations based spatio-temporal graph convolutional network for urban traffic prediction,” *Information Sciences*, vol. 621, pp. 580–595, 2023.

[3] F. Li, J. Feng, H. Yan, G. Jin, D. Jin, and Y. Li, “Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution,” *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 1, 2023.

[4] R. Zhong, B. Hu, F. Wang, Y. Feng, Z. Li, X. Song, Y. Wang, S. Lou, and J. Tan, “Multi-factor embedding GNN-based traffic flow prediction considering intersection similarity,” *Neurocomputing*, vol. 620, no. 129193, 2025.

[5] F. Chen, X. Sun, Y. Wang, Z. Xu, and W. Ma, “Adaptive graph neural network for traffic flow prediction considering time variation,” *Expert Systems with Applications*, vol. 255, no. 124430, 2024.

[6] M. Li and Z. Zhu, “Spatial-temporal fusion graph neural networks for traffic flow forecasting,” in *The AAAI Conference on Artificial Intelligence*, vol. 35, pp. 4189–4196, 2020.

[7] W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao, and X. Feng, “Multi-range attentive bicomponent graph convolutional network for traffic forecasting,” in *The AAAI Conference on Artificial Intelligence*, vol. 11, 2019.

[8] C. Zheng, X. Fan, S. Pan, H. Jin, Z. Peng, Z. Wu, C. Wang, and P. S. Yu, “Spatio-temporal joint graph convolutional networks for traffic forecasting,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 1, 2024.

[9] Y. Liu, T. Feng, S. Rasouli, and M. Wong, “ST-DAGCN: A spatiotemporal dual adaptive graph convolutional network model for traffic prediction,” *Neurocomputing*, vol. 601, no. 128175, 2024.

[10] L. Zhang, H. Luan, and J. Zhan, “Stabilization of stop-and-go waves in vehicle traffic flow,” *IEEE Transactions on Automatic Control*, vol. 69, no. 7, pp. 4583–4597, 2024.

[11] S. Wang, C. Shao, J. Zhang, Y. Zheng, and M. Meng, “Traffic flow prediction using bi-directional gated recurrent unit method,” *Urban Informatics*, vol. 1, no. 16, 2022.

[12] S. V. Kumar and L. Vanajakshi, “Short-term traffic flow prediction using seasonal arima model with limited input data,” *European Transport Research Review*, vol. 7, no. 21, 2015.

[13] K. Holden, “Vector auto regression modeling and forecasting,” *Journal of Forecasting*, vol. 14, pp. 159–166, 1995.

[14] X. Luo, L. Niu, and S. Zhang, “An algorithm for traffic flow prediction based on improved SARIMA and GA,” *KSCE Journal of Civil Engineering*, vol. 22, pp. 4107–4115, 2018.

[15] B. Sun, W. Cheng, P. Goswami, and G. Bai, “Short-term traffic forecasting using self-adjusting k-nearest neighbours,” *IET Intelligent Transport Systems*, vol. 12, pp. 41–48, 2018.

[16] Y. Wang, C. Jing, S. Xu, and T. Guo, “Attention based spatiotemporal graph attention networks for traffic flow forecasting,” *Information Sciences*, vol. 607, pp. 869–883, 2022.

[17] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, “Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction,” *Sensors*, vol. 17, no. 818, 2017.

[18] J. Ye, J. Zhao, K. Ye, and C. Xu, “How to build a graph-based deep learning architecture in traffic domain: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 5, 2022.

[19] Q. Chen, W. Wang, X. Huang, and L. Hai Ning, “Attention-based recurrent neural network for traffic flow prediction,” *Journal of Internet Technology*, vol. 21, no. 3, pp. 831–839, 2020.

[20] J. Ye, J. Zhao, K. Ye, and C. Xu, “Multi-STGCnet: A graph convolution based spatial-temporal framework for subway passenger flow forecasting,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2020.

[21] Y. Liu, S. Rasouli, M. Wong, T. Feng, and T. Huang, “RT-GCN: Gaussian-based spatiotemporal graph convolutional network for robust traffic prediction,” *Information Fusion*, vol. 21, no. 102078, 2024.

[22] H. Wang, R. Zhang, X. Cheng, and L. Yang, “Hierarchical traffic flow prediction based on

spatial-temporal graph convolutional network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16137–16147, 2022.

[23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018.

[24] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A graph multi-attention network for traffic prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 1234–1241, 2020.

[25] C. Chen, K. Li, S. G. Teo, X. Zou, K. Wang, J. Wang, and Z. Zeng, "Gated residual recurrent graph neural networks for traffic prediction," in *AAAI Conference on Aetificial Intelligence*, vol. 23, pp. 16137–16147, 2022.

[26] Z. Cui, L. Lin, Z. Pu, and Y. Wang, "Graph markov network for traffic forecasting with missing data," *Transportation Research Part C: Emerging Technologies*, vol. 117, no. 102671, 2020.

[27] S. Guo, Y. Lin, N. Feng, and C. S. H. Wang, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, no. 101, pp. 922–929, 2019.

[28] Y. Xiao and Y. Yin, "Hybrid LSTM neural network for short-term traffic flow prediction," *Information*, vol. 10, no. 3, 2019.

[29] S. Wang, J. Zhao, C. Shao, C. Dong, and C. Yin, "Truck traffic flow prediction based on LSTM and GRU methods with sampled gps data," *IEEE Access*, vol. 8, pp. 208158–208169, 2020.

[30] S. Lan, Y. Ma, W. Haung, W. Wang, H. Yang, and P. Li, "DSTAGNN: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting," in *Proceedings of the 39th International Conference on Machine Learning* (K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, eds.), vol. 162 of *Proceedings of Machine Learning Research*, pp. 11906–11917, PMLR, 17–23 Jul 2022.

[31] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, IJCAI-2018, p. 3634–3640, International Joint Conferences on Artificial Intelligence Organization, July 2018.

[32] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word

recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 43–49, 1978.

[33] T. Kim, J. Park, J. Yoo, J. M. Ha, and B. D. Youn, "Enhancing gearbox fault diagnosis under phase estimation errors: A dynamic time warping and blind deconvolution approach," *Journal of Sound and Vibration*, vol. 598, no. 118851, 2025.

[34] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.

[35] I. D. Mienye, T. G. Swart, and G. Obaido, "Recurrent neural networks: A comprehensive review of architectures, variants, and applications," *Information*, vol. 15, no. 9, 2024.

[36] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[37] HERE, "Sample map data for students," 2021. Accessed: 2025-01-15.

[38] G. Buroni, Y.-A. Le Borgne, G. Bontempi, and K. Determe, "Cluster analysis of on-board-unit truck big data from the brussels capital region," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2074–2079, 2018.

[39] INRIX, "Inrix reveals congestion at germany's worst traffic hotspots to cost drivers €48 billion over the next decade," 2016. Accessed: 2025-04-03.

[40] The Brussels Times, "Record-breaking traffic jams on flemish roads in january," *The Brussels Times*. Accessed: 2025-04-03.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[42] kaiming He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

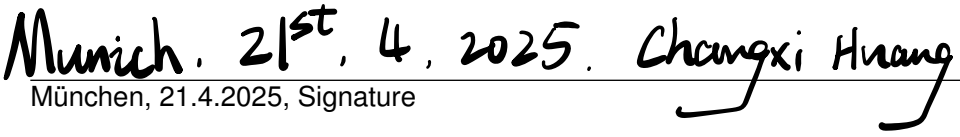
[44] X. Xu, X. Hu, Y. Zhao, X. Lv, and A. Aapaoja, "Urban short-term traffic speed prediction with complicated information fusion on accidents," *Expert Systems with Applications*, vol. 224, no. 119887, 2023.

[45] C. Wu, S. Zhou, C. Fan, and L. Qi, “A method for short-term traffic flow prediction and control on urban road,” in *2021 7th International Conference on Big Data Computing and Communications (BigCom)*, pp. 263–270, 2021.

[46] G. You, “Sustainable vehicle routing problem on real-time roads: the restrictive inheritance-based heuristic algorithm,” *Sustainable Cities and Society*, vol. 79, no. 103682, 2022.

# Declaration

I hereby confirm that the presented thesis work has been done independently and using only the sources and resources as are listed. This thesis has not previously been submitted elsewhere for purposes of assessment.




München, 21.4.2025, Signature