Chair of Transportation Systems Engineering
TUM School of Engineering and Design
Technical University of Munich

Fast adaptation of spatio-temporal traffic prediction to temporal pattern shift

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 Univ.-Prof. Dr. Constantinos Antoniou
M.Eng. Cheng Lyu
M.Sc. Ningkang Yang
Chair of Transportation Systems Engineering

Submitted by Zhaofan Chen
Arcisstrasse 21
80333 München

Submitted on München, 23.10.2024

Abstract

Understanding shifts in temporal patterns due to unprecedented events is crucial for enabling quick adjustments in traffic strategies. However, capturing these temporal pattern shifts with conventional statistical methods proves challenging, especially for cities with limited data. This thesis proposes a deep-learning-based feature extractor designed to capture temporal pattern shifts in source cities, with knowledge transfer facilitated to a target city via domain adversarial. The feature extractor consists of the Position-aware Graph Neural Network (PGNN) and the 1D Convolutional Neural Network (1D CNN). The PGNN, which takes distance-weighted aggregation over anchor sets, can capture the position-specific spatial correlations of traffic volumes in the road network, while 1D CNN is for fusing the spatial correlations over different time steps. To evaluate the effectiveness of proposed architecture, a case study, which transfers a traffic forecasting model pretrained on Antwerp and Bangkok to Barcelona, is conducted. Experiments in source cities show that proposed feature extractor can effectively capture temporal pattern shift and demonstrates a strong robustness against the impact of pandemic compared to baseline models. Meanwhile, the model maintains a high forecasting accuracy when the feature extractor is transferred to the target city.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

Traffic prediction plays a crucial role in Intelligent Transportation Systems (ITS), enabling authorities to efficiently allocate limited resources to meet traffic demands and mitigate potential economic losses. Specifically, traffic prediction involves forecasting traffic status, such as speed, volume and density, across a road network for a specified future time interval using statistical models. Since the traffic conditions at different time points can be viewed as a sequence of discrete, chronologically ordered data points, traffic prediction is inherently a spatio-temporal forecasting task.

Considering temporal aspect, autocorrelation, seasonal features and trend features would jointly decide the current state. Autocorrelation can be briefly explained as the influences of previous time steps on current step. Seasonal and trend features mean periodical and long-term patterns of the sequence. Correspondingly, traditional statistical models including Autoregressive Integrated Moving Average (ARIMA) and Vector Autoregressive (VAR) Model could help us to predict the macroscopic traffic volume within a certain time scale. However, these models need an assumption that data should be stationary, meaning that data disrupted by emergent event is not suitable.

Spatially, at a certain time step, a node would be influenced greatly by its vicinity more than distant nodes. Such a spatial heterogeneity could help us to understand the diffusion process such as traffic congestion spread in the road network and detect critical nodes where the traffic volumes would be significantly more than other nodes in the network. Spatial statistical models such as Geographically Weighted Regression (GWR) and Spatially Weighted Regression (SWR) could assist us to quantify the spatial heterogeneity between target node and the rest nodes in the network. Specifically, the regression coefficients between target node and its neighbors would be larger than the counterpart between target node and others. Conversely, their performance might be compromised under public emergency because normal travel patterns would be disturbed by the policies of the authority.

Comprehensively, traditional time-series and spatial statistical models might not be competent for prediction task especially considering the disruption from large-scale emergent event. For example, extreme weather condition may reduce the traffic speed rapidly but increase the traffic density dramatically. Such a sudden change does not satisfy the assumption of conventional time-series statistical model. To capture the potential spatial-temporal features behind the disrupted data, data-drive method would be a suitable alternative. Many machine-learning and deep-learning based approaches already proved data-driven method's huge potential in digging out sophisticated spatial-temporal relationship during normal period but few studies involved exploring shifting patterns under emergent event. Also, the amount and quality of data would exert a great impact on the model performance. The cities with sufficient training data usually could ensure data diversity so models can fully explore spatial-temporal dynamic patterns behind it. By contrast, cities with newly-built infrastructures might have limited data which would only provide limited spatial-temporal patterns and model might show a poor performance in these cities due to possible underfitting.

To address the problems mentioned above, we design a deep-learning based model combined with transfer-learning technique in this thesis. Specifically, on several source cities where sufficient training data exists, a Graph Neural Network (GNN) will be firstly trained, followed by a convolutional operation acting on temporal dimension to fuse the spatial and temporal features of road-network nodes jointly. The outputs will act as the inputs of a predictor that predicts the future traffic status of source cities.

After pretraining on source cities, a domain-adversarial transfer learning will be performed between a domain classifier and a feature extractor, which includes a pretrained GNN and a pretrained 1D CNN, using data from both the source cities and the target city. The domain classifier attempts to correctly identify whether the patterns provided by the feature extractor belong to the target or source city, while the feature extractor aims to cheat the classifier. This approach enables the feature extractor to learn domain-invariant knowledge from both datasets.

1.2. Objectives

The objectives of this thesis cover following three parts:

Firstly, this thesis will explore how the public emergency disrupts the traffic patterns and the traffic predicting model.

Secondly, we aim to design an architecture that can fuse the spatial and temporal patterns of traffic volume together and capture the temporal pattern shift behind them so we can measure the disruption of emergency on traffic patterns.

Thirdly, we will come up with a transferrable framework, which can transfer the knowledge learned from source cities to the target city via domain adversarial, to solve the data-scarcity problem of the target city.

1.3. Contributions

This thesis's contribution includes three parts:

Firstly, this study utilizes the PGNN to extract the spatial patterns of road network. Unlike common GNNs perform node embedding through computational graph, the PGNN assigns each node a unique embedding by aggregating the information from different anchor sets. Such an information aggregating mechanism can help us to distinguish the nodes in a symmetrical graph like road network.

Secondly, this study uses the 1D CNN to capture temporal pattern shift across different time steps. Instead of extracting spatial and temporal patterns respectively and concatenating them together, the 1D convolution operation can fuse the interactions between each node and anchor nodes at different time steps simultaneously.

Thirdly, the domain adversarial is adopted to transfer the temporal-pattern shift knowledge of source cities to the target city and solve the data scarcity problem of the target city.

1.4. Thesis Outline

This thesis includes eight chapters. The rest sections are: Chapter 2 reviews relevant literatures and encapsulate all methodologies applied in these literatures. Chapter 3 illustrates the mathematical theories and model architecture used in this thesis in detail. Chapter 4 introduces the backgrounds of source cities and target city, describing data features. Chapter 5 analyzes and discusses the results of proposed model and baseline models. Chapter 6 summarizes the contributions of this study and shows the prospects for future research.

2. Literature Review

2.1. Traditional Traffic Prediction

Traffic prediction can be briefly defined as leveraging historical traffic data of road network, usually combined with external information such as weather condition, to predict the traffic status in the future. Here, the traffic data or traffic status can be traffic density, speed, traffic volume and other indicators reflecting the usage rate of transport infrastructures.

Traffic prediction method can be mainly divided into two categories, namely model-driven and data-driven method (Zhao et al. 2020). Model-driven method utilizes physical or traffic flow theories to predict the traffic status of road network so this method would have a strong interpretability and be more understandable than data-driven method. Some representatives like cell transmission model (Wei, Cao, and Sun 2013), queuing theory model (Xu et al. 2014) and microscopic fundamental diagram model (Xu, He, and Sha 2013) can forecast traffic status figuratively. However, traffic status could not be simply regarded as an interaction among velocity, traffic density and traffic volume. In reality, external factors including authority's policies, weather and public emergencies also can exert impacts on traffic status significantly. These interferences might even cause normal travel patterns to become unstable rapidly and such a circumstance is beyond the expertise of most model-driven approaches.

Data-driven method can be explained as that model would be firstly trained on massive amounts of historical data and the trained model takes never-seen-before input to predict traffic status using the knowledge acquired from training data (Zhao et al. 2020). Compared with model-driven method, it usually does not need prior knowledge and any assumption. This trait makes data-driven method more flexible while facing different traffic scenarios. Unlike model-driven method that it clearly defines the relationship between traffic features via empirical equation or prior knowledge, data-driven method would learn each feature's contribution and the relationship among features automatically. It seems

that data-driven method is more efficient than model-driven method concerning feature processing.

Moreover, based on the concept of data-driven method, it knows that model performance is sensitive to data quality and quantity. Specifically, if input data is not pre-processed and fed into model directly, model might learn the knowledge from both normal data and noises. That could cause model converge difficultly and perform unstably. Unlike model-driven method, huge amounts of effective data could be a prerequisite of excellent model performance and when it faces data scarcity, data-driven method might underperform model-driven method. Table 1 encapsulates the differences between model-driven and data-driven method.

Table 1 Differences between model-driven and data-driven prediction method

	Model-driven Method	**Data-driven Method**
Modelling Approach	Based on prior/ professional knowledge	Based on the knowledge behind historical data
Explainability	Understandable	Difficult to understand for humans
Computational Complexity	High complexity and suitable for small-scale missions	Depend on data amount, large data amount leads to a high complexity
Sensitivity to Noise	Show a relatively strong robustness against noises	Sensitive and usually need a data preprocess before training
Sensitivity to Data Amount	Unsensitive	Sensitive and data scarcity can cause overfitting problem
Feature Extraction	Rely on professional theories and experience	Automatically learn the interaction mechanism between features but feature engineering can be carried out before training
Real-time Ability	Difficult to reflect real-time situation accurately	Achievable if real-time data is available

2.2. Spatio-temporal Traffic Prediction Models

Spatial-temporal prediction means a prediction mission that considers the shifting patterns on temporal dimension and the spatial correlation on spatial dimension simultaneously. This mission is applied widely in meteorological, geographical and transport field (Shi et al. 2015). Shi et al. (2015) firstly came up with a ConvLSTM architecture to forecast the precipitation given a local region. Zhang, Zheng, and Qi (2017) proposed Spatio-Temporal Residual Networks (ST-ResNet) for flow prediction at grid level. They firstly divided flow data into 3 parts along temporal dimension: closeness, period and trend. Each part was fed into a ConvResNet architecture respectively. Finally, the outputs of 3 ConvResNet architectures were fused as prediction results. Wang et al. (2018) proposed a RegionTrans method for crowd-flow prediction. They firstly partitioned city map into equal-sized grid level and pretrained a ConvLSTM in a source city. In target city, the loss function covered 2 parts, regression loss and the divergence between each target-city grid and its mapped source-city grid. Instead of using ConvLSTM architecture merely, Zhang et al. (2020) used a Multi-Graph Convolutional Network (MGCN) and a ConvLSTM to extract spatial and temporal dependency separately. The proposed architecture outperformed common ConvLSTM-based models.

However, the drawbacks of ConvLSTM and its derivatives limit their performances on traffic prediction:

1. ConvLSTM could be applied for image-shaped data only considering 2D convolution mechanism and actual road network's shape usually is not regular. If raw data is not rectangular-shaped image data, it will be transformed into the shape that ConvLSTM needs and this problem might hinder ConvLSTM's wide application in traffic prediction.

2. 2D convolution operation can loss some information between a node and its neighbors. During convolution operation, the receptive field of filter is fixed, indicating that each node only considers the influences from fixed-number neighbors. In reality, the neighbor amount of each node is not constant.

3. ConvLSTM shows a relatively weak robustness against graph structure. In each receptive field, filter does the same convolution operation and does not consider the

heterogeneity among different positions, ignoring connectivity difference between a node and its neighbors.

4. ConvLSTM can loss spatial information of graph. The topological structures of graph such as community and loop reflect the connectivity between nodes but 2D convolution operation could not process such topological information but node-based features merely.

5. ConvLSTM does not consider edge properties and directionality of a graph. 2D convolution operation does not consider the influence of edge weight on information flow and the diffusion direction of information.

A road network can be abstracted as a graph. A graph, depicted as $G = (V, E)$, usually consists of 2 basic elements: nodes and edges. Nodes are expressed as a set $V = \{v_1, v_2, v_3, \cdots, v_n\}$ and each node has corresponding attribute values. Edges are used to connect nodes in a graph and reflect the connectivity between nodes. Edges are written as a set $E = \{e_1, e_2, e_3, \cdots, e_n\}$ and an edge $e = (u, v)$ means it connects node u and v. It mainly has 5 criteria for graph classification:

1. Edge directionality. Graphs are classified as undirected and directed graph. Undirected graph means that any pair of nodes connected by an edge have a symmetrical relationship. In a directed graph, edges show directionality. Edge $e = (u, v)$ indicates that the start and end node of e are node u and v. If it has no connection from v to u, edge e will be a one-way connection.

2. Edge weight. A graph with weighted edges is called weighted graph while the graph only with unweighted edges is unweighted graphs. In weighted graph, edge weight's magnitude reflects the correlation strength between 2 nodes and the design of weight magnitude usually depends on professional knowledge. The weight values in an unweighted graph are the same for all edges.

3. Amounts of nodes and edges. If actual edge amounts are much less than maximum possible edge amounts, $E \ll V^2$, graph will be sparse graph. If actual edge amounts are close to maximum possible edge amounts, $E \approx V^2$, graph will be classified as dense graph.

4. Node attributes. All edges or nodes in a graph belong to the same type or have the same attributes, the graph would be homogeneous. A heterogeneous graph means that its edges or nodes could belong to different categories.

5. Connectivity. A connected graph can be defined as that in an undirected graph, any two nodes can find a path to link them together. Disconnected graph refers that a graph exists at least one pair of nodes without path to join them. Strongly connected graph would be that if any node can reach other others via a directed path in a directed graph.

In traffic prediction mission, the road network can be treated as a directed-weighted graph. Nodes can be intersections of main roads, tollbooth, tunnels, etc. Edges usually are the roads to link these nodes together and edge weights can be the reciprocal of road length, directionality being designed based on road network's actual situation.

Since road network is a subtype of graph, many studies utilized GNN-based approaches to dig out spatial corelations of road network. According to GNN categories used in prediction, it can be mainly categorized into six categories:

Table 2 The characteristics of different GNNs

	Model Characteristics
Recurrent GNN (RGNN) based	Updates node features in spatial dimension iteratively
Graph Auto Encoder (GAE) based	Encoder embeds each node based on its attributes and topological structure in the graph. Decoder reconstructs a graph via the embeddings from encoder.
Graph Convolution Network (GCN) based	Aggregate each node's all neighbors' attributes
GraphSAGE based	Sample partial neighbors and aggregate their attributes
Graph Attention (GAT) based	Assign each neighbor an attention score to reflect its importance to target node. Attention scores adjust during training

Yu, Yin, and Zhu (2018) proposed a Spatio-Temporal Graph Convolutional Networks (STGCN) to replace regular convolution-recurrent-unit structure. They took two temporal gated convolution layers, between which a GCN layer was placed, to extract spatio-temporal correlations. Zhao et al. (2020) designed a Temporal Graph Convolutional Network (T-GCN) so that spatial and temporal dependencies were extracted by a GCN and Gated Recurrent Unit (GRU) sequentially. Considering that Recurrent Neural Network (RNN) and LSTM might be not competent for long-term sequence prediction, Wu et al. (2019) adopted a dilated causal convolution to expand the receptive field for temporal convolution, whose outputs were fed into a GCN layer. Unlike previous studies using two separate deep-learning modules to learn spatial and temporal dependencies, Song et al. (2020) proposed a Spatial-Temporal Synchronous Graph Convolutional Network (STSGCN) to learn spatio-temporal patterns simultaneously. They firstly connected all nodes with themselves at adjacent timesteps to form a localized spatio-temporal graph signal matrix. Then, GCN layers processed the matrix and an aggregation layer fused GCN's outputs via cropping operation. Liu (2022) proposed to extract short and long-term temporal characteristics. They took instantaneous-fusion and long-term-dependency module to catch short-term and long-term signals separately. Liu, Cao, and Dong (2023) divided input sequences along time axis into odd-index and even-index sequence and utilized a Dynamic Graph Convolution Network (DGCN) to process them separately and fused two sequences together.

Since spatial-temporal dependencies might be different among regions, Geng et al. (2019) proposed a Spatio-temporal Multi-Graph Convolution Network (ST-MGCN). In each region, it firstly divided features into three types: neighborhood, functional similarity and connectivity. They utilized Contextual Gated Recurrent Neural Network (CGRNN) to process each feature globally. Then, a multi-graph convolution was performed to learn the correlation among regions. Bai et al. (2020) argued that directly leveraging predefined graph and adjacent matrix for graph convolution might not fully reflect the dynamic correlations between nodes at different time steps so they proposed an Adaptive Graph Convolutional Recurrent Network (AGCRN). Namely, a Node Adaptive Parameter Learning (NAPL) module firstly learned node-level features for each time step and a Data Adaptive Graph Generation (DAGG) module constructed a new graph based on NAPL's outputs.

In spired by self-attention mechanism which is widely used in Natural Language Processing (NLP), Computer Vision (CV) and time-series analysis, some studies leveraged self-attention mechanism to explore the spatio-temporal correlation in traffic data. Guo et al. (2022) utilized attention mechanism to extract spatial and temporal dependencies sequentially. Then, a GCNConv block was used to make further process. Zheng et al. (2019) proposed a Graph Multi-Attention Network (GMAN). The core section of GMAN was ST-Attention block. In the block, a spatial and a temporal attention processed spatial and temporal patterns of input firstly and their outputs fused through gated fusion. Huang et al. (2024) proposed a Dynamic Spatio-Temporal Graph Transformer Network (DST-GTN) to catch varying need of nodes under different spatio-temporal scenarios. Input's spatio-temporal dynamics and daily/weekly-periodic features were caught by a dynamic spatio-temporal (Dyn-ST) embedding and a temporal identity embedding module respectively. A temporal transformer module then processed the concatenated embeddings. Finally, Dynamic Spatio-Temporal Module (DSTM) output prediction results. Specifically, DSTM included a Dynamic Spatio-Temporal Graph Generator (DSTGG) and a Node Frequency Learning Spatio-temporal Graph Convolution Network (NFLSTGCN). DSTGG was used to construct a global spatio-temporal graph and NFLSTGCN was used to extract the demands of each node for local and global information.

H. Liu et al. (2023) proposed an Attention-based Spatio-Temporal Graph Convolutional Recurrent Network (ASTGCRN). They substituted original Multi-layer Perceptron (MLP) layer in Gated Recurrent Unit (GRU) with Adaptive Graph Convolution Network (AGCN). To solve the problem of GRU not competent for long-term prediction, GRU was followed by an attention layer.

2.3. Transfer Learning

Transfer learning can be defined as a machine-learning method that a model utilizes the knowledge learned from source task to promote its performance on target task (Pan and Yang 2010). Transfer learning can be categorized into three types: inductive, transductive and unsupervised transfer learning. Inductive transfer learning means that the model

transfers the knowledge obtained from source task to target task given conditions that source and target task are different but target task would be supervised-learning. By contrast, transductive transfer learning requires that source and target task are the same but their data distributions are different and it usually has few or no labels in target data. The last category, unsupervised learning, means that both source and target data are without labels; model utilizes source-domain knowledge to boost its performance on source task.

The strength of transfer learning can be summarised in following three points:

1. Transfer learning can reduce training cost. Usually, a random initialized deep-learning model would take lots of time and computation resources to reach convergence. If it deploys a model that already pretrained on the source data to target data, the model would converge quickly.

2. Transfer learning can promote model's generalization ability. If a deep-learning model is directly trained on limited target data, overfitting could occur easily. When model starts training on target data with a set of pretrained parameters, the knowledge extracted from source task could help effectively mitigate overfitting.

3. Transfer learning can keep model relatively stable while training on target data. On target dataset, a randomly initialized model might be unstable especially during early-training stage. By comparison, the parameters of a pretrained model usually are within a feasible range and their gradient values' variance would be smaller.

Therefore, transfer-learning strategy would be promising in extracting city-invariant spatial-temporal correlation and predicting traffic status in data-scarcity cities. Zhang et al. (2019) proposed a transductive transfer-learning strategy considering both data scarcity in target task and the similarity among cellular traffic. They firstly clustered cellular dataset of source task. Secondly, they took the Spatial–Temporal Cross-domain neural Network (STCNet) trained by last cluster as the initial state for next-cluster training so diversity and similarity of different clusters were preserved. These pretrained STCNet models were finetuned on target data lastly. Li et al. (2020) proposed a Maximum Mean Discrepancy (MMD) strategy. During training, the loss function of predictor covered both

regression loss and the distance between source and target data. Huang et al. (2021) partitioned target city's road network into subgraphs and did the same on source city. They firstly trained predictor using source-city data at subgraph level and deployed pre-trained model on target-city subgraphs. Li et al. (2021) proposed a transfer-learning method on feature space. They decomposed traffic flow by Empirical Mode Decomposition (EMD) and performed a Transfer Component Analysis (TCA) in feature space to minimize the discrepancy between target and source data. Wang et al. (2022) proposed a Deep Attentive Adaptation Network model named ST-DAAN to transfer source-data knowledge to target data for crowd flow prediction. Firstly, a spatio-temporal knowledge transfer was conducted on feature space via MMD. Then, a global spatial attention mechanism was performed to explore the correlations between each source cell and all target cells. To extract the knowledge at different spatial scales, Mo and Gong (2023) proposed a multi-granular transfer-learning strategy. In source cities, feature embeddings were extracted at different spatial scales and a knowledge distiller was used to capture spatial-temporal patterns. In target city, feature embeddings were matched with the most suitable multi-granular meta-knowledge via attention mechanism.

Wu et al. (2022) devised a Generative Adversarial Network (GAN) based approach. A predictor was firstly pretrained using source data. When predictor was transferred to target data, target data passed a generator. The generated data was sent to both the pre-trained predictor and a discriminator whose inputs included the generated data and source data. Namely, the discriminator was designed to judge whether generator can generate domain-invariant features using target data while the predictor was used to check whether these domain-invariant features are helpful for prediction mission. Tang et al. (2022) also proposed an adversarial-based strategy. An encoder and a domain classifier were designed to extract city-invariant topological features from the road networks of both source and target cities and a predictor was pretrained using source data simultaneously. When the architecture was deployed on target city, an extra private encoder was connected to it to provide target-city road network's private information. Zhang et al. (2022) proposed a Conditional GAN (CGAN) based architecture. During pretraining, generator input a random noise vector and traffic demands to generate the traffic distribution of source cities. Also, traffic demands of source cities were clustered. When pre-trained architecture was transferred on target city, to promote generalization ability, a

cluster matching regularizer was added to loss function so the distance between each target data point and its corresponding cluster was minimized.

Research gaps are listed below considering all studied mentioned above:

1. Most studies were focused on the prediction mission during normal period while few studies (Tsai et al. 2021) involved extracting spatial-temporal patterns considering the impact of public emergency.

2. Most GNN-based architectures were deployed on graph-structural data directly but grid-based data seems more common and few GNN-based studies utilized grid-based data.

3. Traditional GNN's expressive power limitation usually is not considered especially when it faces positional-aware task.

3. Methodology

3.1. Problem Definition

In this study, we perform traffic prediction using graph structure. The road network of one city is simplified as a graph $g = (V, E)$. $V = \{v_1, v_2, \cdots, v_n\}$ represents the critical nodes in the network and $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \cdots, \{v_{n-1}, v_n\}\}$ represents the edges connecting these critical nodes.

Then, we divide the traffic volumes of road network g into t parts with the same-length time interval and the traffic volumes are expressed as $\{X_1^g, X_2^g, \cdots, X_t^g\}$, where $X_i^g = \{x_{i,1}, x_{i,2}, \cdots, x_{i,n}\}$ represents the traffic volumes for n critical nodes at time interval t.

Given the road networks of different source cities $G_s = \{g_1, g_2, \cdots, g_s\}$ with sufficient volume data and one target city road network g_t with limited volume data, we aim to design a spatio-temporal fusion predicting model, which transfers the temporal-pattern shift knowledge of source cities to the target city, to predict the traffic volume $\{X_{t+1}^{g_t}, X_{t+2}^{g_t}, \cdots, X_{t+l}^{g_t}\}$ of the target city using its limited historical volume data $\{X_{t-k_t}^{g_t}, X_{t-k_t+1}^{g_t}, \cdots, X_t^{g_t}\}$ and the historical volume data $\{X_{t-k_s}^{g_i}, X_{t-k_s+1}^{g_i}, \cdots, X_t^{g_i}\}$ of different source cities $G_t = \{g_1, g_2, \cdots, g_s\}$, where $k_t \ll k_s$.

3.2. Proposed Architecture

The proposed architecture comprises three components: a feature extractor, a domain classifier, and a predictor. The feature extractor integrates a PGNN and a 1D CNN. The predictor is a Recurrent Neural Network (RNN) or LSTM, while the domain classifier is a MLP. In the pretraining phase, the feature extractor and predictor are pretrained on traffic volume data from the source cities. Subsequently, a domain adversarial training stage is conducted, involving the pretrained feature extractor, pretrained predictor, and a

randomly initialized domain classifier, utilizing volume data from both the source cities and the target city. Figure 1 displays the entire procedure.

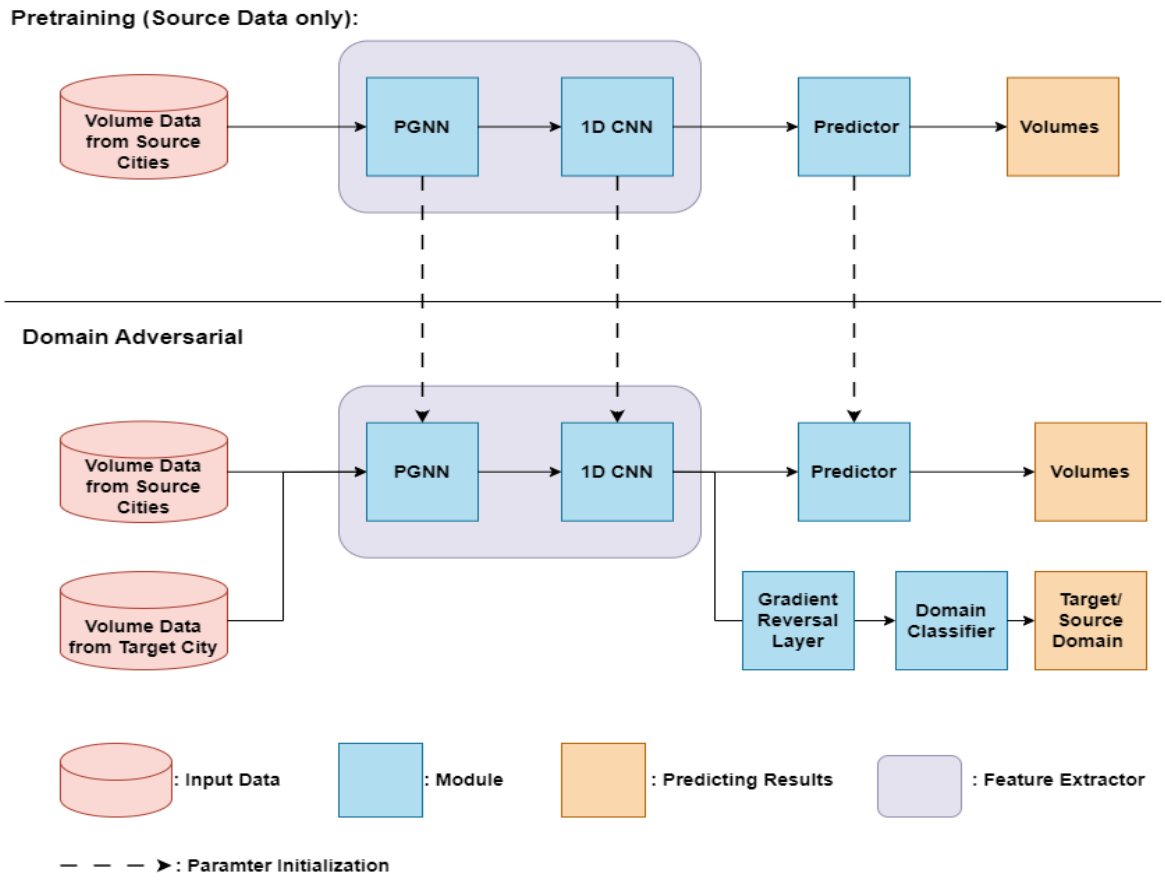

Figure 1 Proposed architecture in this thesis

If it directly trains entire architecture using source and target data, the architecture might perform unstably and difficult to converge. On the one hand, such a deep-layer feature extractor might converge slowly or have relatively large fluctuations due to possible gradient disappearance or gradient distortion. On the other hand, the proposed architecture is adversarial-based: domain classifier acts as a role to hinder feature extractor to learn city-specific feature and feature extractor learns to cheat it via studying city-invariant patterns. If it directly starts a domain adversarial for a randomly initialized feature extractor, feature extractor might focus on learning how to cheat domain classifier but neglect whether these patterns are useful for prediction task. Therefore, feature extractor was

pretrained on source data firstly and it ensures that feature extractor already can capture useful patterns for prediction before domain adversarial begins.

Feature extractor is composed of two modules: one PGNN and one 1D CNN. The reason why it connects those two modules sequentially can be illustrated by Figure 2.

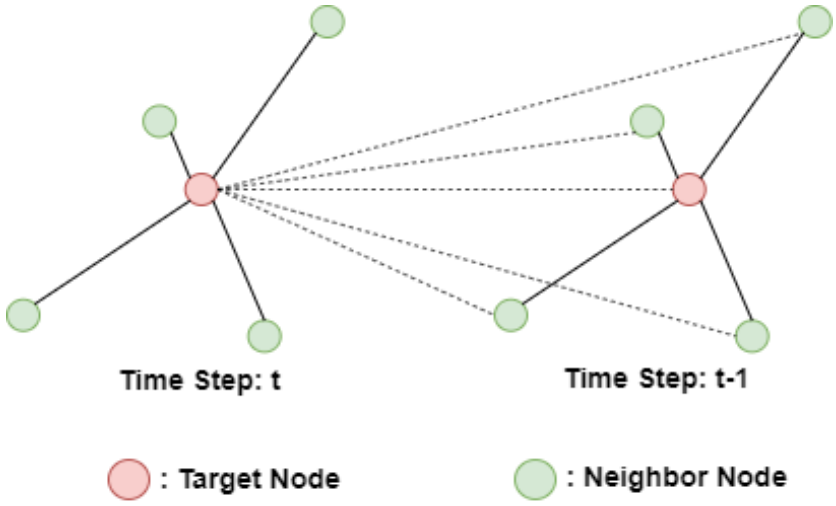

Figure 2 Interaction between target node and its neighbors

A target node is usually influenced by its neighbors spatially. Along the time dimension, the target node's status is mainly influenced by two parts. The first part is its own status at current and previous time steps. The second part is its neighbors' status at current and previous time steps. To capture the spatio-temporal patterns hidden behind it. It firstly utilized a PGNN to encode the status of target node at each time step separately. Then, a convolutional operation was performed along time axis and repeated on each status dimension. If the layer number of PGNN goes deeper, the receptive field of target node will become broader spatially but oversmoothing problem might occur. Specifically, the essence of graph convolution is to aggregate neighbors' information and fuse it with target node's own information. When a GNN's layer number goes deeper, target node could aggregate the information from both its neighbors and the neighbors of its neigbors. If a graph has a high connectivity, the feature embeddings of nodes might become similar because the neighbors of different nodes are highly overlapped, which eventually

degrades GNN performance. Therefore, the number of graph-convolution layer still needs to be controlled.

Considering 1D CNN, it uses 1-dimensional convolution to capture temporal shifting patterns for each node. If the number of 1-d dimensional convolution layer goes deeper, it can cover a wider range along temporal dimension in input data.

Since it is a time-series prediction task, feature extractor is followed by predictor. Predictor is used to predict the traffic volume within a time window given spatio-temporal patterns of the past several time steps. LSTM and RNN were taken as the predictor in this thesis. Although LSTM and RNN might not be competent for long-term prediction due to their structural design, 1D CNN made up their drawbacks. Specifically, a long-time sequence was transformed to a short-sized sequence via convolutional operation and the long-term temporal patterns of original data were preserved in this short-sized sequence.

After pretraining on source cities' datasets, a domain adversarial was performed via pretrained feature extractor and a domain classifier. It utilized limited training data from target city and same-sized training samples from source cities to avoid label imbalance problem. If it trains domain classifier using the data in which source data predominates, domain classifier might focus on classifying the samples from source data correctly but ignore the samples from target data, which can cause that feature extractor mainly learns to capture city-invariant patterns from source cities. Correspondingly, it might have a degraded performance on target city at inference stage. Moreover, the loss function at domain-adversarial stage is different to the counterpart at pretraining stage. The loss function considers regression errors only while pretraining. At domain-adversarial stage, it has two loss items totally:

$$L_{total} = L_{regression} + \lambda L_{domain} \tag{3.1}$$

$$L_{domain} = -\frac{1}{N}\sum_{i=1}^{N}[d_i \log D(f_i) + (1 - d_i)\log\big(1 - D(f_i)\big)] \tag{3.2}$$

Where $L_{regression}$ means the regression errors between predictor outputs and ground-truth volumes; L_{domain} means binary cross-entropy loss between domain classifier outputs and ground-truth labels; λ means weight value of binary cross-entropy loss; D means the output probability of domain classifier; d_i is ground-truth domain label; f_i represents feature extractor's output features; N means total sample number.

Regression errors are used to calculate gradient values and update model parameters like pretraining stage while classification errors are firstly used to update domain classifier's parameters and then pass a gradient reversal layer where the gradient signs are reversed. The gradient values with reversed signs are passed to feature extractor so feature extractor learns how to cheat domain classifier and minimize regression errors simultaneously.

3.3. PGNN

3.3.1. Structural-aware and Position-aware Task

Structural-aware task explores the topological structure of a graph and it mainly considers global and local structures between nodes, edges and other elements to capture complex relationship patterns. Positional-aware task focuses on the relative positional information between nodes and usually each node's positional information can exert significant influences on prediction results.

3.3.2. Computation Graph

To obtain the embedding of a target node, the attributes of its neighbors are aggregated firstly and processed by an activation function. Computation graph reflects how messages are passed from neighbors to the target node and aggregated. It covers the node attributes of both target node itself and its neighbors. Computational graph is composed of four elements: target node, the neighbors of target node, edges and the attributes corresponding to nodes and edges. The depth of computation graph is not limited to one

layer and it means that k-hop neighbors' information is considered when layer goes deeper. In short, the information update of a node is to aggregate the messages from all neighbors and itself. Such a process is expressed by formula below:

$$h_v^{(l)} = Concat(AGG\left(\left\{m_u^{(l)}, u \in N(v)\right\}\right), m_v^{(l)}) \tag{3.3}$$

$$m_u^{(l)} = W^{(l)} h_u^{(l-1)} \tag{3.4}$$

$$m_v^{(l)} = B^{(l)} h_v^{(l-1)} \tag{3.5}$$

Where $h_v^{(l)}$ represents node embedding of target node v at l^{th} layer; $m_u^{(l)}$ is the message of neighbor u at l^{th} layer; $m_v^{(l)}$ means target node v's own message at l^{th} layer; W and B are weight matrices; AGG represents aggregation function; $Concat$ is concatenation operation.

3.3.3. Limitations of Common GNNs

As shown in Figure 3., if two nodes in a graph have same neighborhood structures but at different places, a GNN should have generated two different embeddings. However, common GNN models like GCN and GraphSAGE would output same embeddings for node A and B.

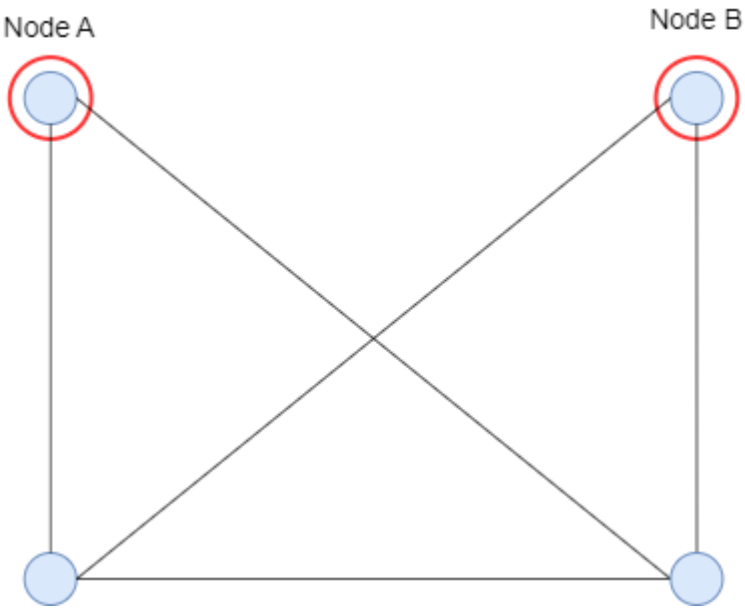

Figure 3 Symmetrical graph

The reason is that common GNN models utilize computation graph to determine the direction of message passing and update node embedding but computation graph might not be suitable for positional-aware tasks such as distinguishing nodes in a symmetrical graph. In Figure 3, the computation graphs of node A and B are completely same because of their same topological structures. Also, symmetrical road networks are common in reality and if it directly utilizes common GNN models to extract the spatial correlations, some subtle spatial differences would be neglected.

3.3.4. Framework of PGNN

The PGNN, proposed by You, Ying, and Leskovec (2019), does not aggregate the messages from target node's local neighbors. Instead, it aggregates the messages from different anchor sets which are randomly selected node subsets. When PGNN performs aggregation, it fuses all nodes' messages all at once unlike common GNN models aggregate each node's message separately. Given graph $G = (\mathcal{V}, \varepsilon)$ and set S covering k anchor sets, $S = \{S_i, \forall i \in 1, \cdots, k\}$, positional-aware message aggregation on one PGNN layer can be expressed by following formulas:

$$h_v = AGG_s(\{M_v[i], \forall i \in 1, \cdots, k\}) \quad (3.6)$$

$$z_v = \sigma(M_v\, \omega) \quad (3.7)$$

$$M_v[i] = AGG_M(\mathcal{M}_i) \quad (3.8)$$

$$\mathcal{M}_i = \{F(u, v, h_u, h_v), u \in S_i\} \quad (3.9)$$

$$S_i \sim \mathcal{V}, i = 1, \cdots, k \quad (3.10)$$

$$z_v \epsilon \mathbb{R}^k, \forall v \epsilon \mathcal{V} \quad (3.11)$$

$$M_v \epsilon \mathbb{R}^{k \times r} \quad (3.12)$$

Where h_v is structure-aware message and z_v is positional-aware embedding of node v; AGG_s and AGG_M are the aggregation functions for aggregating the messages across anchor sets and within an anchor set respectively; M represents anchor-set message matrix whose ith row, $\mathcal{M}_i$, is the ith anchor set's message; F is feature combination function and it combines two nodes' features weighted by their distance; σ is non-linear activation function and ω is a learnable weight matrix.

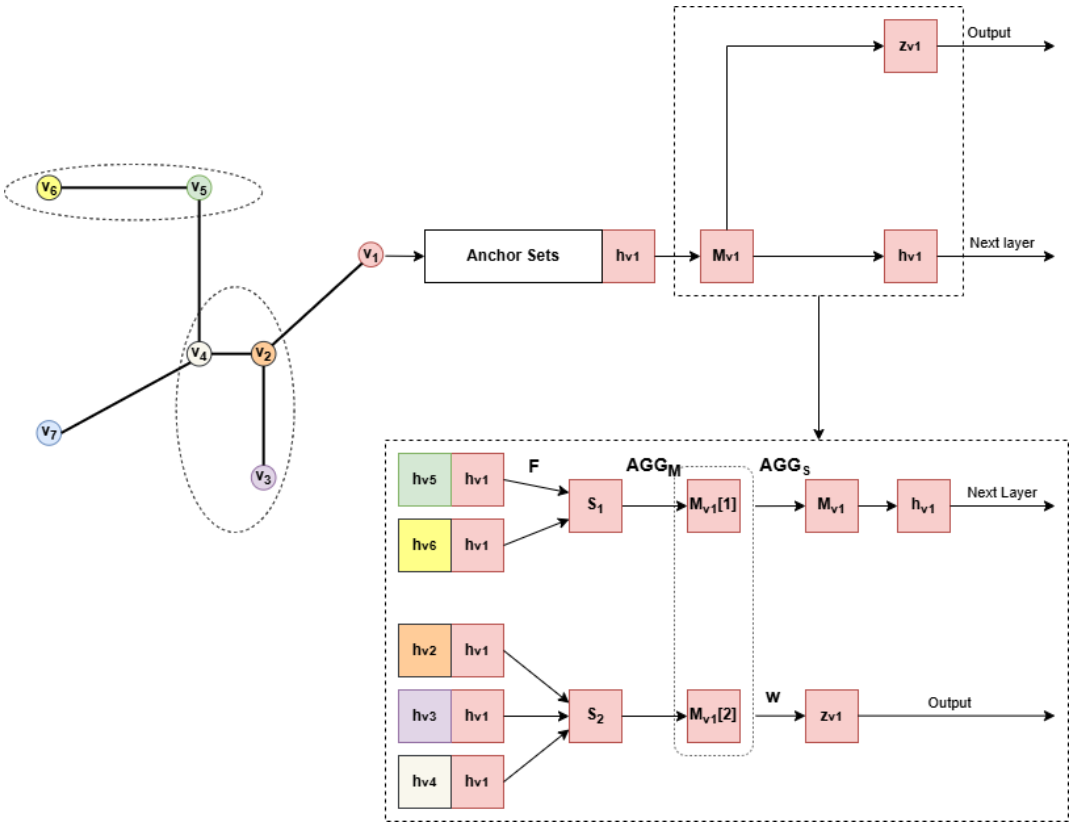

Figure 4 PGNN

It notes that each PGNN layer could not utilize the anchor sets generated by last layer and it needs to randomly generate new sets. Considering that, it could not directly send the positional-aware embeddings of all nodes to next PGNN layer since the relative position of each node to anchor sets is not constant at each PGNN layer. The best solution is to send the structure-aware message, h_v, to next layer because it covers the messages across different anchor sets and is also the output of an order-invariant aggregation function, AGG_s.

3.4. 1D CNN

3.4.1. Convolution Concept and Classification

Convolutional operation can be explained as that a filter moves along input data and performs local operation to extract data patterns. Commonly used convolution operation

can be divided into four types: standard convolution, depth-wise separable convolution, transposed convolution and group convolution.

Standard convolution is that when filter moves along input data, the input area covered by filter is multiplied by filter weights and summed up. After filter slides entire input data, it obtains output features. Standard convolution can effectively capture local features of an image or time sequence. Depth-wise separable convolution is a simplified convolutional operation. It decomposes convolution operation into depth-wise and point-wise convolution. Depth-wise convolution means that each channel of input needs a filter and point-wise convolution uses a 1×1 filter to perform convolution on each channel. Depth-wise separable convolution significantly reduces the filter parameters and promote model training efficiency and it is usually used in network compression. Transposed convolution is the reverse process of normal convolution. It restores small-sized feature maps to larger-sized maps and usually used in generative model or improving image resolution. Group convolution is to divide input channels into several groups and each group is processed by a filter respectively. Convolution results of different filters are concatenated later.

3.4.2. 1D Convolution

1D convolution utilizes a filter to slide along input data and performs element-wise multiplication and summation operation. Its mathematical expression is shown below:

$$y[n] = \sum_{i=0}^{K-1} \omega[i] x[n-i] \tag{3.13}$$

Where $x[i]$ is input sequence with length N; $\omega[i]$ is a filter with length K; $y[n]$ is convolution output with length $N - K + 1$; n and k are the position indices of y and ω.

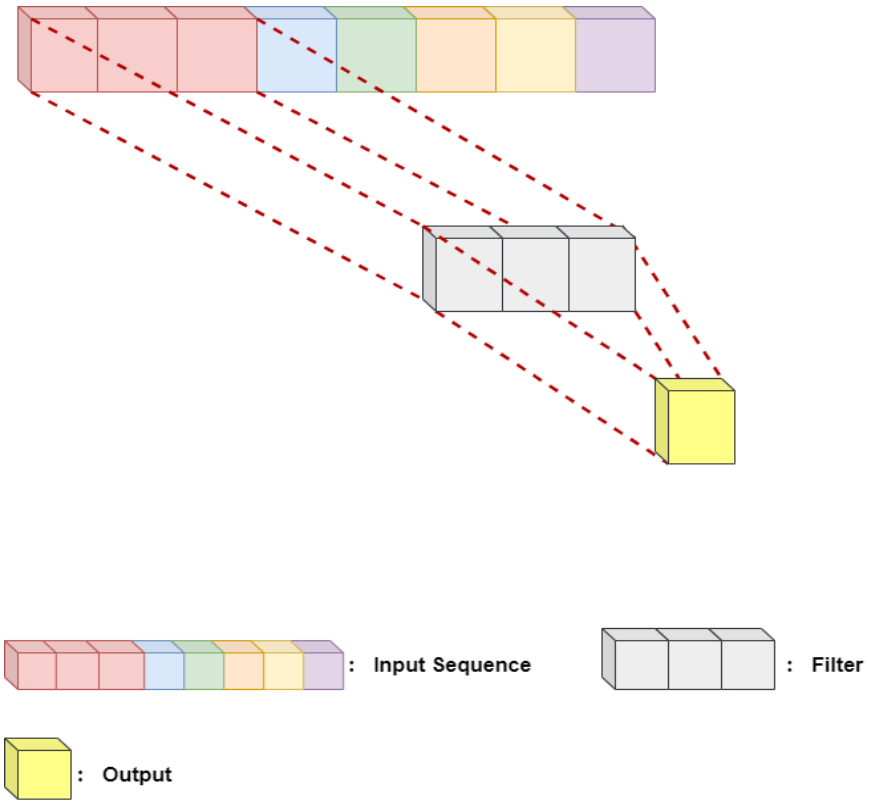

Figure 5 1D Convolution operation

3.5. Predictor

3.5.1. RNN

RNN is designed for processing sequence data and it leverages recurrent connection mechanism to capture the sequential information from input. Compared with traditional Feedforward Neural Network (FNN), it has recurrent structure which enables RNN share the information from previous steps. Figure 6 displays the structure of RNN and RNN includes four parts: input layer, hidden layer, recurrent connection and output layer. Input layer receives sequence data. Hidden layer saves the hidden state of current time step and the hidden state is a function of current input and the hidden state of last time step:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b_h) \tag{3.14}$$

Where h_t represents the hidden state at current time step; x_t is the input at current time step; W_h and W_x are learnable weight matrices; b_h is bias item and σ represents non-linear activation function.

Output layer generates outputs based on hidden states at different time steps:

$$y_t = W_y h_t + b_y \tag{3.15}$$

Where W_y is a learnable weight matrix and y_t and b_y are t^{th} step output and bias item.

The last part, recurrent connection, is used to connect last step's hidden state with the input at current step. Since the hidden state of current step is impacted by current input and last step's hidden state simultaneously.

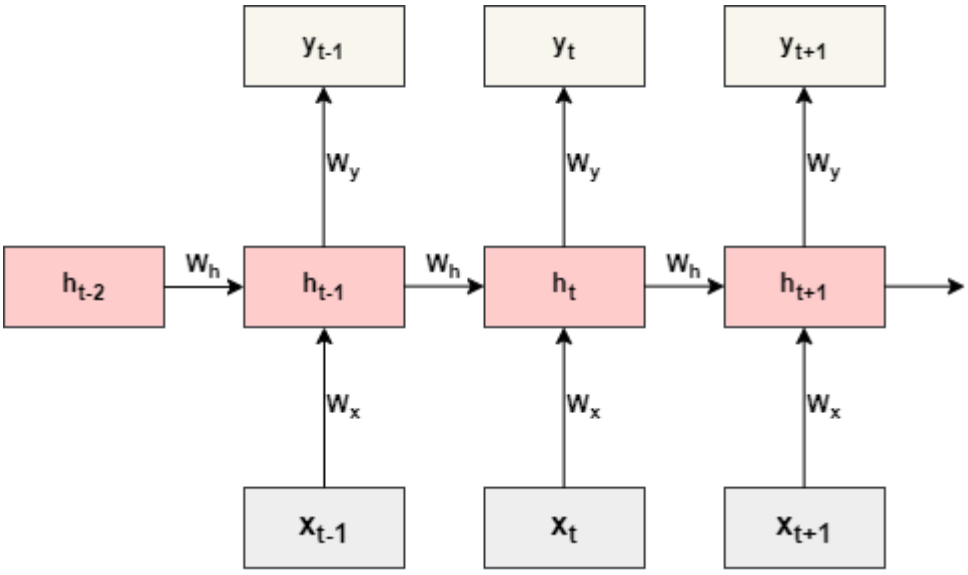

Figure 6 RNN structure

RNN still has some drawbacks. Firstly, gradient explosion and disappearance could occur and that can make RNN hard to converge or performed unstably. RNN updates parameters based on gradient descent and chain rule. If input sequence is too long or the network goes deeper, the gradient value corresponding to previous steps might become too small or too large. Secondly, RNN could be competent for short-term prediction but could not capture long-term patterns effectively. Also, RNN usually converges slower than other models like CNN. Each time RNN only extracts the features corresponding to one time step.

3.5.2. LSTM

LSTM utilizes gated mechanism and cell state to store the information of previous step and Figure 7 shows the structure of it. LSTM includes four parts: input gate, forget gate, output gate and cell state. Input gate controls how much information from input can be saved to cell state. Forget gate is used to decide that how much previous information in the cell state is abandoned and output gate decides the output considering both current cell state and input. Cell state stores the information of previous steps and dynamically updates itself considering current input and last cell state.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{3.16}$$

$$f_t = \sigma\big(W_f[h_{t-1}, x_t] + b_f\big) \tag{3.17}$$

$$\widetilde{C_t} = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{3.18}$$

$$C_t = f_t C_{t-1} + i_t \widetilde{C_t} \tag{3.19}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.20}$$

$$h_t = o_t \tanh(C_t) \tag{3.21}$$

Where h_t and x_t are the hidden state and input at step t; W_i, W_f and W_o are trainable weight metrics; b_i, b_f and b_o are bias items; i_t, f_t and o_t represents input, forget and output gate; $\widetilde{C_t}$ and C_t are candidate cell state and cell state.

LSTM utilizes gated mechanism to save useful information while remove useless information effectively and uses a cell state to store previous information, which mitigates the gradient disappearance and the problem that RNN is not competent for long-term prediction.

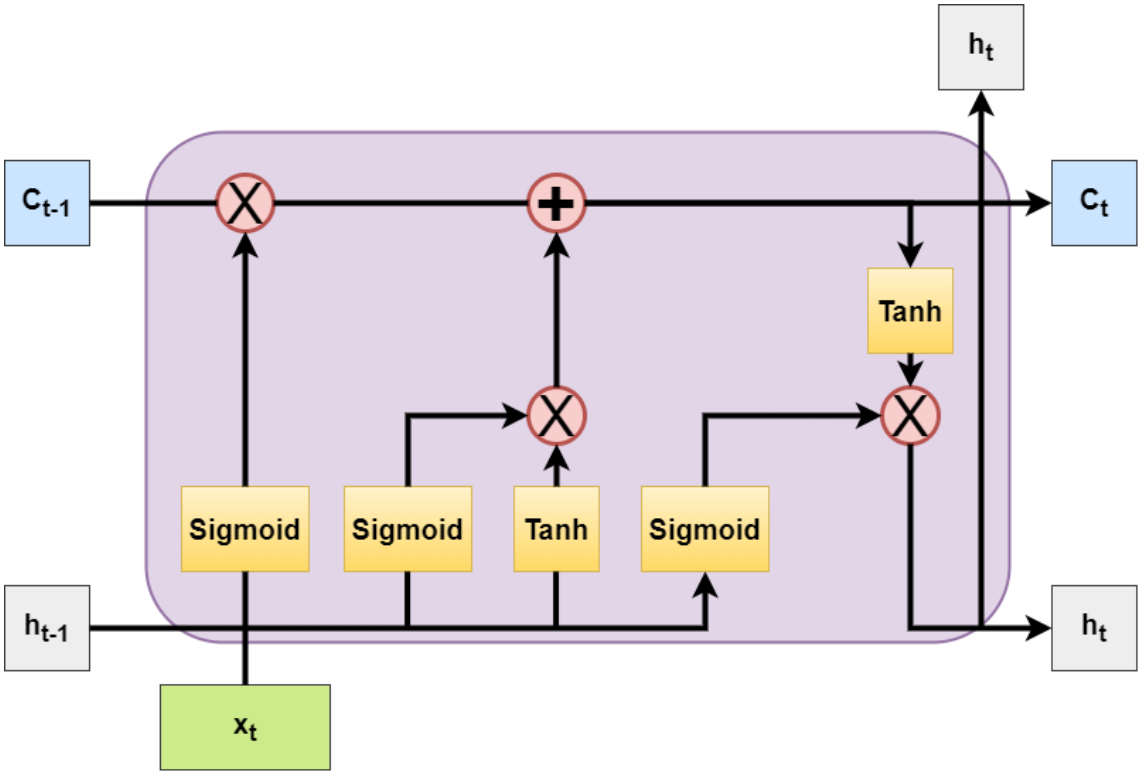

Figure 7 One neuron structure of LSTM

3.6. Fine-tuning

Fine-tuning is a transfer-learning technique that it fine-tunes a pretrained model's parameters based on specific down-stream missions. Its procedure covers three steps. Firstly, it needs to selected a fitted pretrained model that already learned useful knowledge from large-scale datasets. Secondly, the parameters of shallow layers are frozen while deep layers' counterparts are used for fine-tuning. Since the shallow layers of a pretrained model usually extract common knowledge across different datasets. Thirdly, pretrained

model is trained using target data and some hyperparameters like learning rate are adjusted further.

In this thesis, it pretrained feature extractor and predictor only with source cities' data and transferred them to target data as initialized state. Instead of freezing shallow layers, it fine-tuned entire feature extractor and predictor so it might fit data distribution of target city more effectively.

3.7. Domain Adversarial

Prediction mission in this thesis is a transductive transfer-learning mission. Namely, it predicts the traffic volume on both source cities and target city and the proposed architecture needs to solve the problem that the data distribution between source cities and target city is different. The core idea of domain adversarial is to learn domain-invariant spatial-temporal patterns by using the training data from two domains. Figure 8 shows the domain-adversarial strategy used in this thesis.

In this thesis, it mainly includes four parts: feature extractor, domain classifier, predictor and gradient reversal layer. Feature extractor aims to distil spatio-temporal patterns from cities; domain classifier tries to judge the features extracted by feature extractor belong to source cities or target city and predictor predicts traffic volume for each city given feature extractor's output. Gradient reversal layer plays as a vital role in domain adversarial. During forward propagation, it transmits feature extractor's output to domain classifier normally. When domain classifier calculates binary cross-entropy loss and performs backpropagation, gradient values are multiplied by a negative number when they pass the gradient reversal layer. This negative number is depicted as $-\lambda$ and λ is a hyper parameter reflecting the power of gradient reverse. The reason why it sets a gradient reversal layer between feature extractor and domain classifier is to make feature extractor updates its parameters along the direction of gradient and it could cause domain classifier's binary cross-entropy loss become larger during next iteration. Through such an

adversarial process, feature extractor could learn domain-invariant but useful information using the data from source cities and target city.

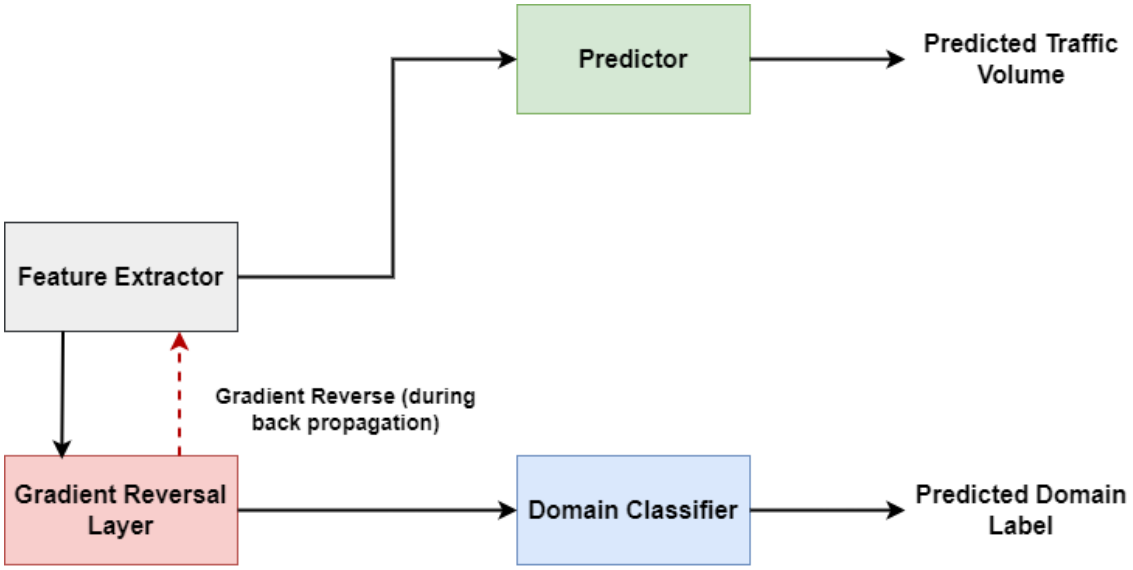

Figure 8 Domain adversarial architecture

4. Case Study

4.1. Background

Large-scale public emergency could disrupt daily life and policy making for local authorities. In this thesis, the public emergency means COVID-19. COVID-19, an unprecedented epidemic, initially was found in late 2019 in China and started dispersing to entire world rapidly since early 2020. It caused catastrophic damages to economy, manufacturing, transport, logistics, etc. By June 2022, it caused 2,141 deaths per 1 million population in Europe (Worldometer 2023). It reports that by the first quarter of 2021, COVID-19 caused the real GDP of European-Union countries dropped continuously since 2020 (Muggenthaler, Schroth, and Sun 2021). When the epidemic spread rapidly among population, each country started taking epidemic prevention measures. Italy, the firstly European country affected by the epidemic severely, announced national-wise lockdown from March 9th 2020 and took strict isolation measures. Other EU countries like Spain, France and Germany kept pace with Italy and they closed schools and unnecessary stories and limited large-scale assembling.

Correspondingly, original normal travel patterns were interfered. For example, school-based and amusement-based trips dramatically dropped due to the lock-down policy and some studies explored potential shifting patterns during the epidemic. Das et al. (2021) studied significant changes in mobility patterns in India. They found that travellers increasingly relied on private transport and a significant drop in public-transport passenger volumes was found. Harris and Branion-Calles (2021) studied the commute mode changes in Canada. They also found a significant decline in public-transport demands and more original commuters started remote working. Moreover, they found that demographic features influenced commute mode choice. Drift, Wismans, and Olde Kalter (2021) explored the pattern changes in Netherland. Except obtaining the results similar to Harris and Branion-Calles (2021) and Das et al. (2021), it is interesting that travel distance reduced significantly and more travellers shifted to cycling and walking instead of driving private vehicle. Beck and Hensher (2020) studied the household travel behaviour changes after easing restrictions in Australia. The epidemic made travel frequency

dropped and the popularity of remote work might alter pre-pandemic travel behaviours in the future.

In short, various pandemic prevention measures might change traffic demand dramatically and make the trend of traffic volume unstable especially during early-epidemic period. In this thesis, it uses proposed model architecture to extract the spatio-temporal shifting patterns of three cities: Barcelona, Bangkok and Antwerp. Specifically, Barcelona and Bangkok are source cities but Antwerp is the target city.

4.2. Road Network Demonstration

Figure 9 shows the road network structure of Barcelona. Barcelona is the capital city of the autonomous region of Catalonia, Spain, and the country's second largest city. Since March 21st 2020, Barcelona authority performed a series of pandemic prevention measures. Specifically, travel restrictions were only ineffective for medical emergency and work. Social assembling was limited to six persons and mobility was restricted from 10 pm to 6 am. Restaurants were allowed to operate at thirty-percent capacity indoors and fully outdoors. Public venues like cinemas and theatres could operate at fifty-percent capacity, with strict ventilation requirements to ensure safety. As situation improved, many of these restrictions gradually eased, culminating in the removal of most capacity limits by October 2021 as the city returned to near-normal operations.

Figure 9 Road network of Barcelona

The road network of Bangkok is displayed in Figure 10. On March 26, 2020, a state of emergency was declared, enforcing strict travel restrictions, a nightly curfew, and banning gatherings. Inter-provincial movement was discouraged, and international flights were suspended. Bangkok authorities further restricted public venues on April 29, while social distancing measures, mask mandates, and hygiene practices became widespread across the city. As cases dropped, the Thai government initiated a phased relaxation of restrictions starting in May 2020, reopening low-risk businesses. Schools were closed from April, with plans for reopening shifted to July 2020 under strict protocols. Testing efforts were ramped up, particularly focusing on vulnerable groups. By July 2020, the country had contained much of the virus, with 50 provinces reporting no new cases in the past 28 days.

Figure 10 Road network of Bangkok

Antwerp is an important city in the Flemish region in northern Belgium. Antwerp is the second largest city in Belgium and one of the busiest ports in Europe and its road network is shown in Figure 11. In Antwerp, Belgium, the response to the COVID-19 pandemic involved several key measures throughout 2020. In March, Belgium implemented strict lockdown measures, including school closures, banning gatherings, and limiting non-essential movements. On March 18, a nationwide lockdown came into effect, restricting travel and mandating social distancing. By May, as the situation improved, the country began gradually easing restrictions, allowing certain businesses to reopen and outdoor activities to resume under specific guidelines.

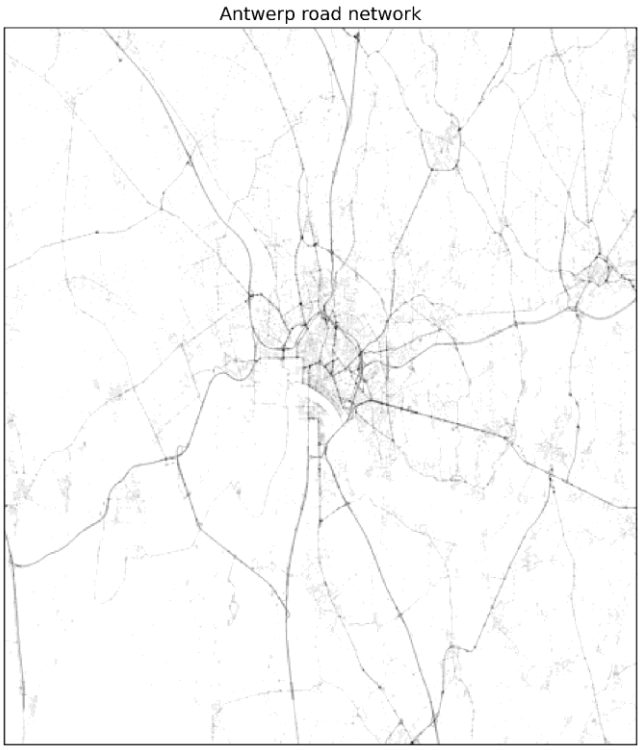

Figure 11 Road network of Antwerp

4.3. Data Description

The traffic data used in this thesis was provided by HERE (2021). Originally it included two equally-sized parts which were corresponded to the records from January to June in 2019 and 2020 separately. This thesis aims to extract the shifting patterns during pandemic so it used the traffic data in 2020 only. Specifically, the image-shaped data for one day is a (288,495,436,4) tensor where the first dimension means that traffic volumes are aggregated per five minutes; the second and third dimension represent a 495×436 resolution for each city and each pixel's size is 100m ×100m. The last dimension represents traffic volumes aggregated on four directions: north-east, north-west, south-west and south-east.

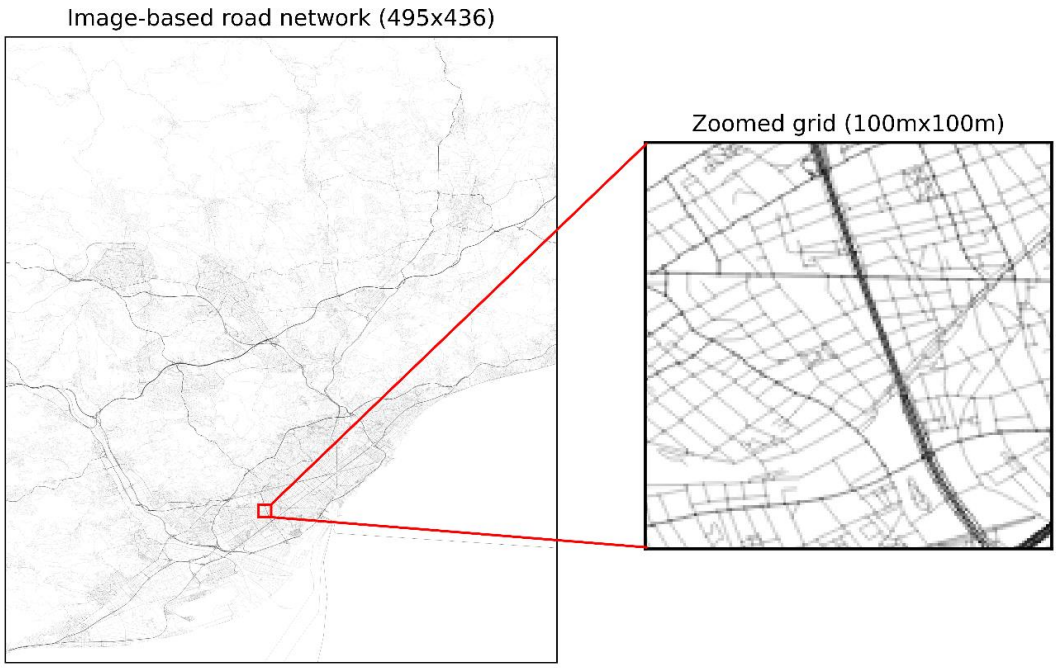

Figure 12 Image-based volume data

Also, it has a static map whose dimension is (9,495,436). The first channel is a gray-scale representation of the city map in the same resolution as the dynamic data. The rest eight channels represent the connectivity of each grid on eight directions separately: north, north-east, east, south-east, south, south-west, west and north-west and Figure 13 gives an example of it.

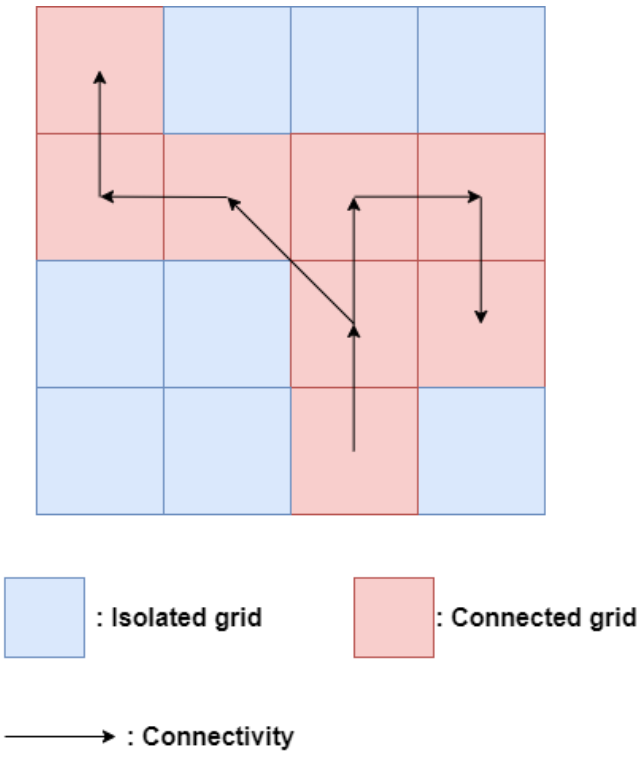

Figure 13 Graph connectivity of static map

4.4. Study Area and Critical Nodes Selection

If it considers entire map as model input, it would be computationally expensive and this thesis only focused on each city's central area and Figure 14 shows the study areas of three cities respectively.

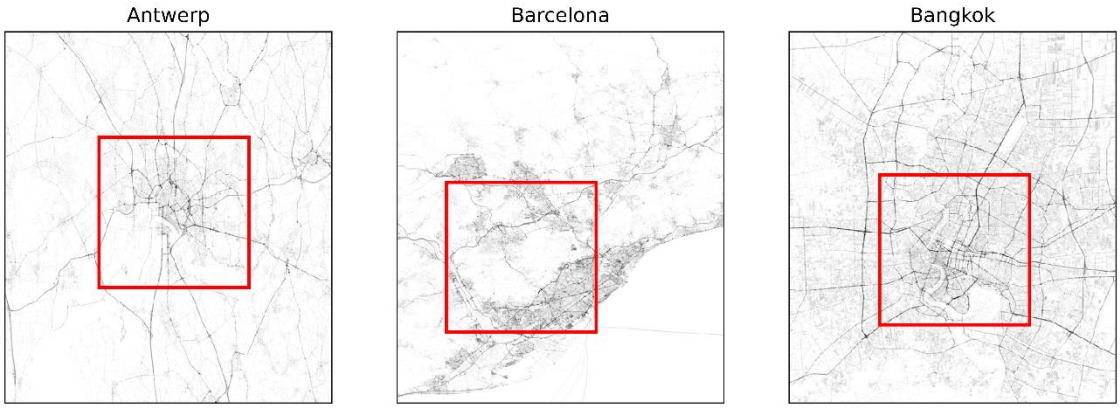

Figure 14 Study areas of Antwerp, Barcelona and Bangkok (the study area of each city is bounded by a 200×200 red box)

It calculated grid-level daily volume from January 2020 to June 2020 for each city separately. Figure 15 reflects the daily-volume varying patterns of Antwerp. It finds that relatively high traffic volumes, which were over 100vehs/day approximately, were mainly distributed along central ring and the roads stretching outside. From January to March, it seemed no significant changes in daily volume. When authority announced a nationwide lockdown on March 18, a significant reduction in traffic volume, especially at central area of Antwerp, can be noticed in April and May. When local government eased restrictions since May, traffic volume started increasing gradually and it can be found in June.

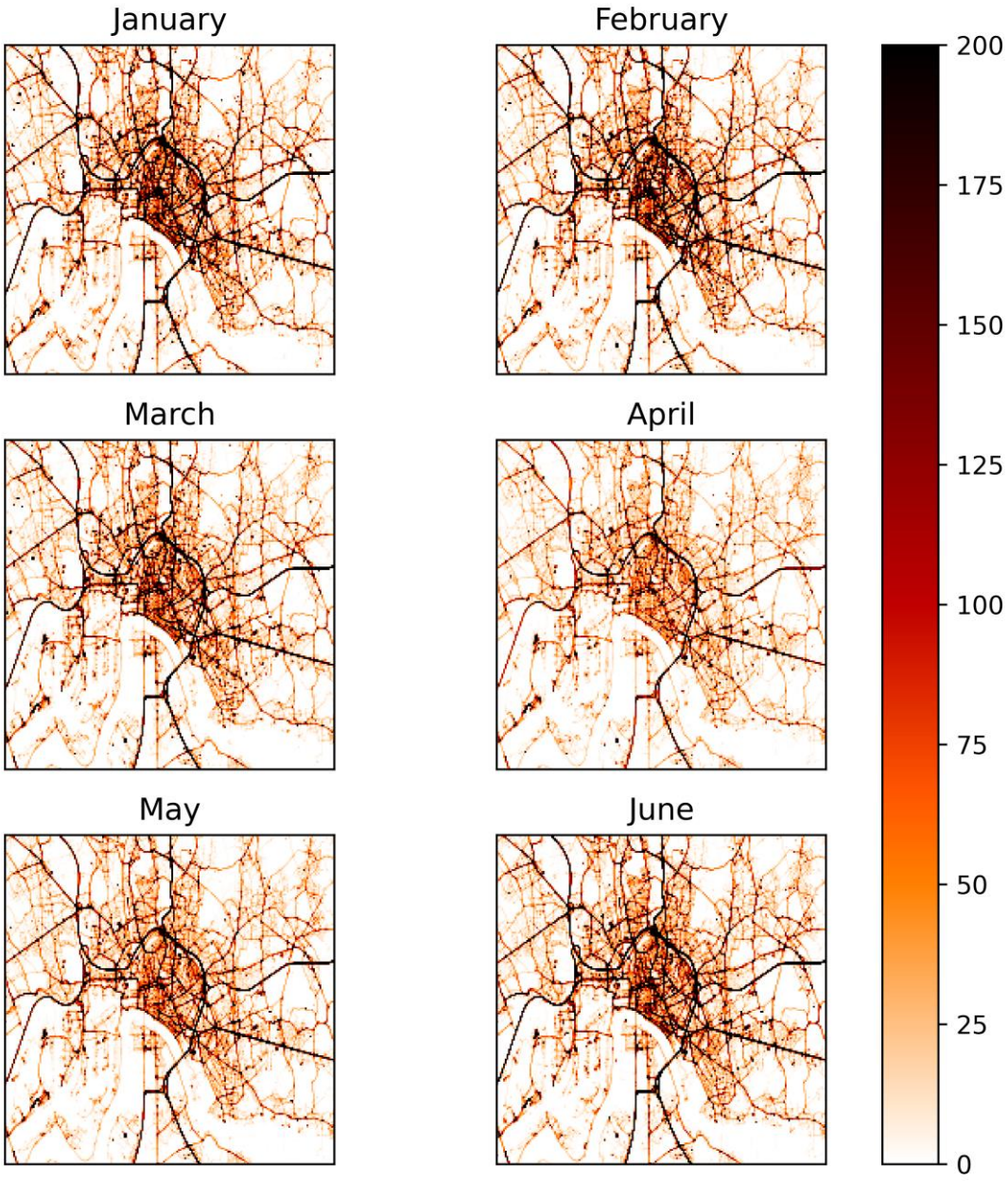

Figure 15 Daily traffic volume of Antwerp (unit: vehs/day)

The situation of Barcelona is shown in Figure 16. Similarly, there seems no obvious variations from January to March. With a series of restrictions published by the authority, traffic volume also dropped dramatically over entire study area. Traffic volume rebounded in June because Barcelona authority also eased the restrictions gradually.

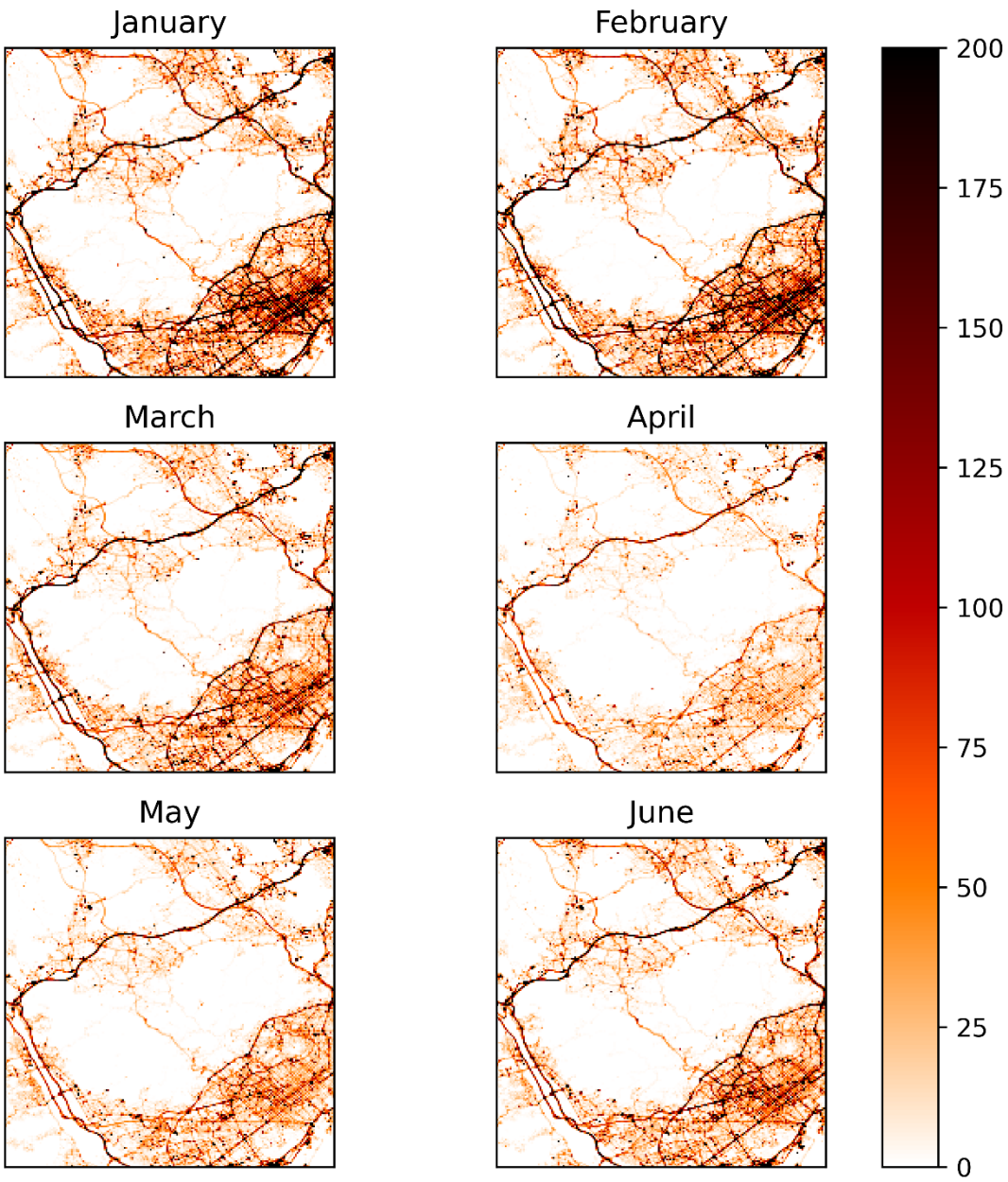

Figure 16 Daily traffic volume of Barcelona (unit: vehs/day)

As for the situation of Bangkok, traffic volume mainly distributed along the main roads. It observes that a volume reduction began from April. Since a state of emergency was declared on March 26, 2020. The traffic volume significantly decreased especially at central area of Bangkok while peripheral areas seemed to be influenced more slightly and this situation lasted until May. In June, traffic volume started increasing and that could be observed on main roads with previous restrictions being relaxed gradually.

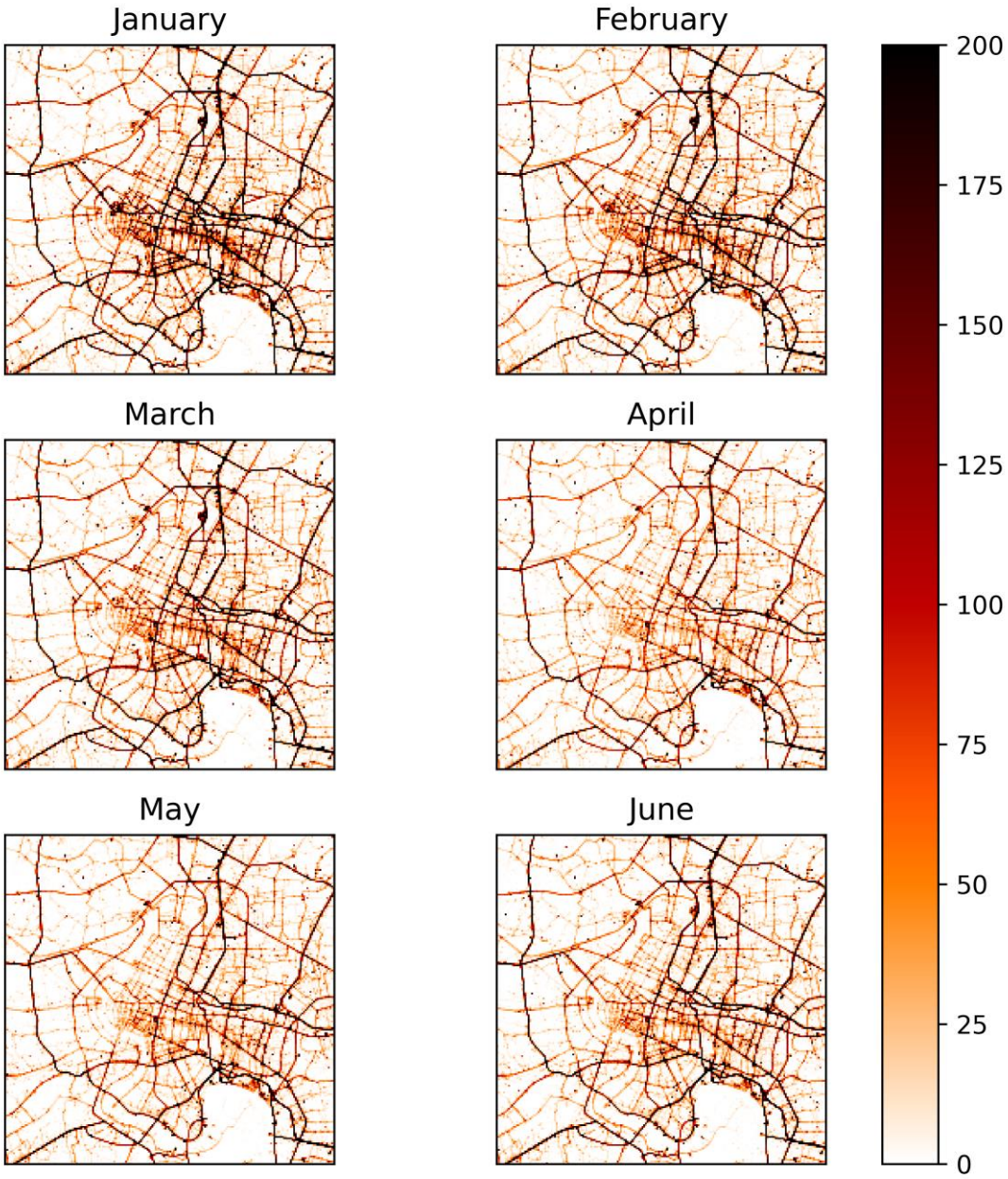

Figure 17 Daily traffic volume of Bangkok (unit: vehs/day)

Before building graph structures, it calculated each grid's daily volume in 2020 and Table 3 Statistical outcomes of daily volume at grid level (unit: vehs/day) summarizes the statistical results for three cities. It finds that at grid level, daily volume is sparse since nearly 75-percent grids' volumes do not exceed 30 vehicles per day. It means that only few grids play as vital roles in a road network such as the grids representing important intersections and city's main roads. If it extracts entire road network structure and inputs the

traffic volume directly on it to proposed model architecture, model might extract useful spatio-temporal patterns difficultly and hard to converge. Because critical grids can be separated by lots of zero-volume grids and these zero-volume grids hinders GNN to learn the spatial correlation between critical nodes.

Table 3 Statistical outcomes of daily volume at grid level (unit: vehs/day)

	Antwerp	**Barcelona**	**Bangkok**
Mean	16	9	35
Variance	23,662	16,489	10,247
25th Percentile	0	0	0
Median	4	2	3
75th Percentile	29	24	22
Max Value	4,508	11,257	4,798

To solve this problem, as shown in Figure 18, it firstly detected critical nodes by a certain rule within study area and connected them together so a simplified road network structure was established but critical information of original network was saved effectively. Specifically, critical nodes in road network were determined by the percentile of daily traffic volume and the percentile was adjusted in different cities. For example, if a grid's daily volume value exceeds lower bound, the daily volume value of the 90th percentile, and does not exceed upper bound, the daily volume value of 99th percentile, this grid would be selected as a critical node. Table 4 shows the lower and upper bound volume values corresponding to three cities. After determining all critical nodes of a road network, it connected these critical nodes together based on the paths in original road network. It calculated the shortest distance between critical nodes and set a distance threshold to judge whether two critical nodes should have an edge to link them. If the shortest distance between two critical nodes does not exceed the threshold, an edge between them would be created. Edge weight was calculated by a Gaussian kernel function expressed as following:

$$K(x, x') = exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right) \quad (4.1)$$

Where x is the shortest distance between a pair of critical nodes; x' is the median value of shortest distances for all critical-node pairs and σ represents the band width of Gaussian kernel function.

Table 4 Lower and upper bound of daily traffic volume for critical node selection

	Antwerp	**Barcelona**	**Bangkok**
Lower-bound percentile number	84	90	90
Upper-bound percentile number	97	99	99
Distance threshold value	10	10	10
Critical node amount	1466	1273	1137

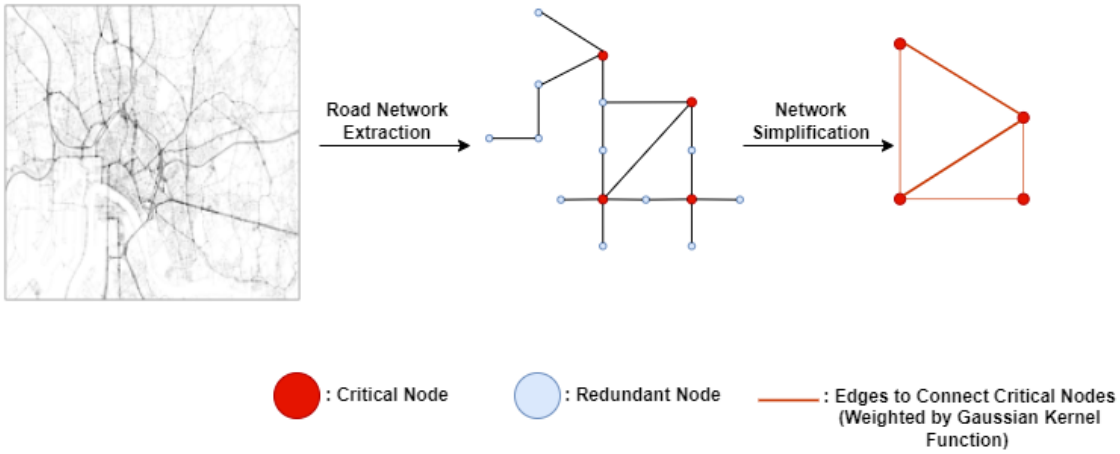

Figure 18 Road network simplification

Figure 19 displays the spatial distribution of critical nodes in Antwerp. Partial critical nodes distribute along the central circular road network while others distribute along the

peripheral road network and follow the roads stretching outside. It can be explained by that most vehicles reach the central ring through radial road network and they are either parked at parking lots along the ring or diverted to the roads stretching outside the city.

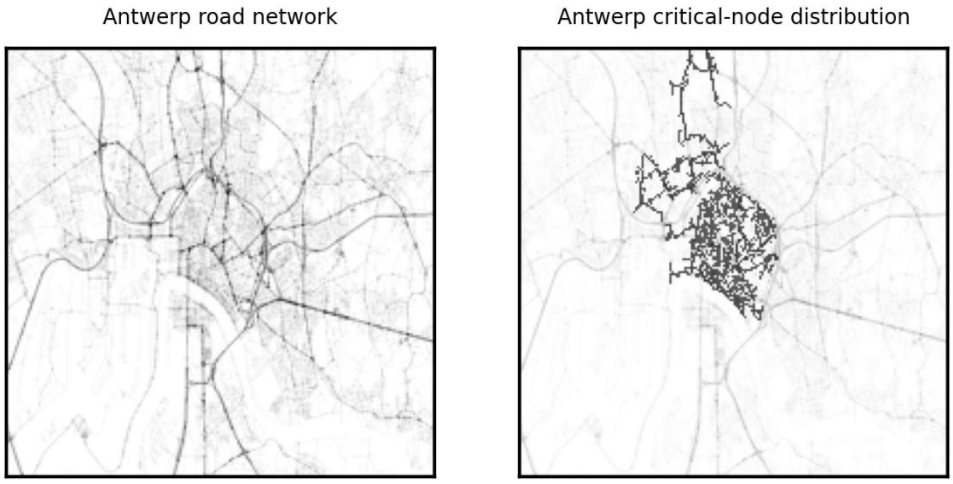

Figure 19 Critical nodes of Antwerp

Figure 20 shows critical-node positions in original road network of Barcelona. It finds that most critical nodes are distributed in southern area of the network and one possible reason is that workplaces, amusement facilities, shopping malls and historical sites are located at that region.

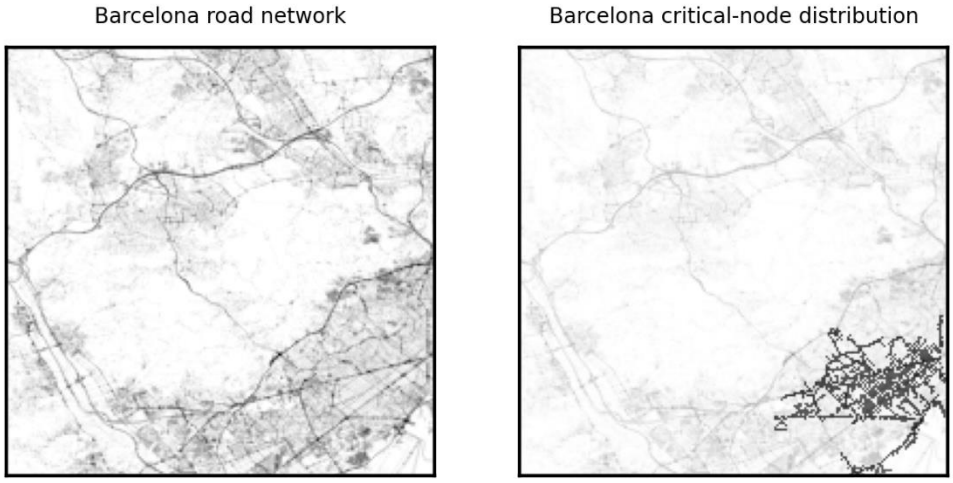

Figure 20 Critical nodes of Barcelona

Critical nodes of Bangkok can be seen in Figure 21. Partial nodes distribute along main roads and other nodes are mainly situated around tourism attractions, shopping malls and residential areas.

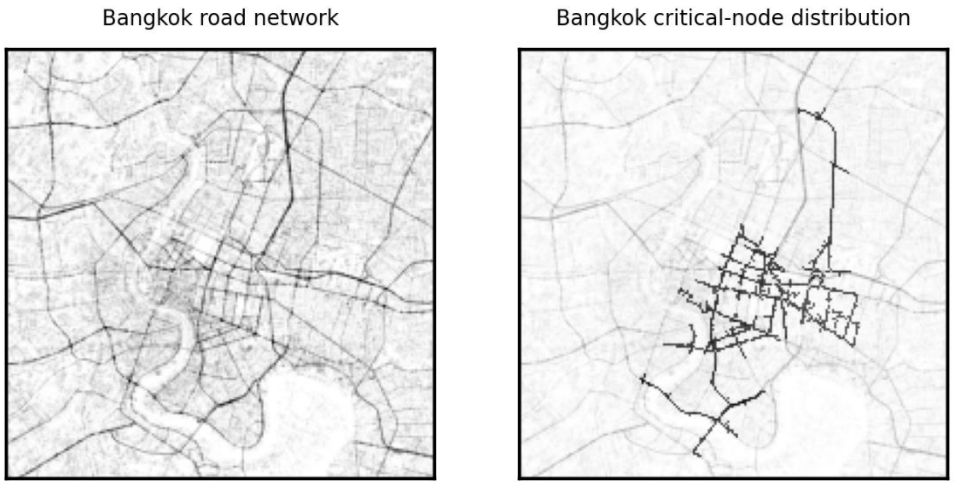

Figure 21 Critical nodes of Bangkok

5. Results Analysis and Discussion

5.1. Data Preparation

5.1.1. Volume Aggregation of Critical Nodes

It knows each volume sample shaped as (288,495,436,4) so it aggregated volume data along the first dimension and the traffic volume for one day was obtained. The sample was converted to a (495,436,4) tensor. Then, it aggregated volume data along this tensor's last dimension and the volumes on four directions were summed up and its shape was changed as (495,436). Since it can obtain the coordinates of all critical nodes based on Chapter 4 results, the daily traffic volume corresponding to each critical node can be extracted from that (495,436) shaped tensor easily.

5.1.2. Outlier Processing and Normalization

Since some daily traffic-volume values are excessively high or excessively low, it sets two thresholds and modifies volume value once it is beyond the boundaries. Specifically, if one critical node's daily volume value exceeds the upper bound or is below the lower bound, the bound value will replace the volume value. Table 5 includes the bounding values of each city. After removing outliers, a min-max normalization was performed:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{5.1}$$

Where x' and x are the value after and before normalization respectively; x_{min} represents the minimum value of dataset and x_{max} is the maximum value of dataset.

Table 5 Lower and upper bound value for outlier processing

	Antwerp	**Barcelona**	**Bangkok**
Lower bound (unit: vehs/day)	29	28	26

	Antwerp	Barcelona	Bangkok
Upper bound (unit: vehs/day)	362	399	328
Percentile (lower bound)	75	77	77
Percentile (upper bound)	97	99	98

5.1.3. Data Split

It took the volume data from Jan 2nd to May 24th in 2020 and split this 144-day data into training set, validation set and test set for each source city but the data amount of target city was only 72-day because target city usually has data-scarcity problems and it mainly utilizes the knowledge learned from data-rich source cities. The ratio between training set, validation set and test set would be 0.6: 0.2: 0.2. In each city, it would treat 28-day volume data as one sample and the sample would be sent to proposed architecture and the model architecture would predict the traffic volume for next 7 days. Table 6 shows the data shape corresponding to training set, validation set and test set. It notes that the third dimension represents maximum critical-node number among three cities. If a city's critical-node number is less than the maximum value, a padding operation would be done so that different cities could have same-sized data.

Table 6 Data shape for three cities

	Antwerp (source)	Bangkok (source)	Barcelona (target)
Input volume (training set)	(70, 28, 1466, 1)	(70, 28, 1466, 1)	(35, 28, 1466, 1)
Output volume (training set)	(70, 7, 1466, 1)	(70, 7, 1466, 1)	(35, 7, 1466, 1)
Input volume (validation set)	(23, 28, 1466, 1)	(23, 28, 1466, 1)	(12, 28, 1466, 1)
Output volume (validation set)	(23, 7, 1466, 1)	(23, 7, 1466, 1)	(12, 7, 1466, 1)

	Antwerp (source)	Bangkok (source)	Barcelona (target)
Input volume (test set)	(24, 28, 1466, 1)	(24, 28, 1466, 1)	(12, 28, 1466, 1)
Output volume (test set)	(24, 7, 1466, 1)	(24, 7, 1466, 1)	(12, 7, 1466, 1)
Domain label (training set only)	(70, 1)	(70, 1)	(35, 1)

5.2. Model Preparation

In proposed model architecture, it firstly used a PGNN to extract the spatial dependency of daily volume. A 1D CNN followed the PGNN and learned the temporal dependency of each node; the outputs of 1D CNN were fed to a LSTM predictor. Base-line models included LSTM, Chebyshev Convolutional LSTM (ChebConv-LSTM), RNN and Cheb-Conv-RNN. Epoch number and batch size were 40 and 16 respectively. Table 7 covers the parameter information of proposed architecture.

Table 7 Parameter setting of proposed architecture

Module	Layer/operation	Layer number	Hidden size/output channels/kernel size
PGNN	PGNN layer	2	Layer 1 = 128 Layer 2 = 64
	Anchor set number		Layer 1 = 50 Layer 2 = 50
	Dropout		Layer 1 = 0.2 Layer 2 = 0.2
	Activation function		Layer 1 = ReLU Layer 2 = ReLU
1D CNN	Convolution layer	3	Layer 1 = 128 Layer 2 = 64 Layer 3 = 64

Module	Layer/operation	Layer number	Hidden size/output channels/kernel size
	Convolution kernel		Layer 1 = 7 Layer 2 = 7 Layer 3 = 6
	Stride		Layer 1 = 1 Layer 2 = 1 Layer 3 = 2
	Activation function		Layer 1 = LeakyReLU Layer 2 = LeakyReLU Layer 3 = LeakyReLU
LSTM	LSTM layer	1	Layer 1 = 128
	Dropout		Layer 1 = 0.2
	Activation function		Layer 1 = ReLU
Domain Classifier	Linear layer	3	Layer 1 = 128 Layer 2 = 64 Layer 3 = 1
	Dropout		Layer 1 = 0.2 Layer 2 = 0.2 Layer 3 = 0.2
	Activation function		Layer 1 = LeakyReLU Layer 2 = LeakyReLU Layer 3 = LeakyReLU

5.2.1. Optimizer Selection

Adaptive Moment Estimation (Adam) is an optimizer algorithm considering both momentum and self-adaptive learning rate. It adjusts learning rate dynamically via calculating the first and second moment of gradient and it is expressed by following formulas:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t \quad (5.2)$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1{}^t} \quad (5.3)$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2{}^t} \quad (5.4)$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad (5.5)$$

$$v_t = \beta_1 v_{t-1} + (1-\beta_2) g_t{}^2 \quad (5.6)$$

Where θ_t is the parameter at t^{th} iteration; η is learning rate; $\hat{m}_t$ and $\hat{v}_t$ are the corrected first and second moment of gradient g_t at t^{th} iteration separately; m_t and v_t are the first and second moment of gradient g_t at t^{th} iteration. β_1 and β_2 are the decay rate.

Nesterov-accelerated Adaptive Moment Estimation (Nadam) is a derivative of Adam and adopts Nesterov momentum. Compared to Adam, a pre-update of gradient is performed when updating model parameters and such an advanced-deployment strategy could make model converge more quickly and reduce oscillation during training. The formula below expressed how parameters update using Nadam:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \left(\hat{m}_t - \beta_1 (\hat{m}_t - \hat{m}_{t-1}) \right) \quad (5.7)$$

It utilized four Nadam optimizers to update the parameters of PGNN, 1D-CNN, LSTM and domain classifier separately and each optimizer has a learning rate scheduler for adjusting learning rate as epoch number goes larger. Their information is listed in Table 8.

Table 8 Optimizer parameters

	Nadam (PGNN)	**Nadam (1D-CNN)**	**Nadam (LSTM)**	**Nadam (domain classifier)**
Learning rate	5e-4	5e-4	4e-4	4e-4
Weight decay	1e-5	1e-5	1e-5	1e-5
Momentum decay	4e-3	4e-3	4e-3	4e-3
Gamma (learning rate scheduler)	0.55	0.55	0.6	0.55
Step size (learning rate scheduler)	11	11	11	11

5.2.2. Loss Function Selection

Considering the loss function for performance evaluation, it used Mean Square Error (MSE) and Mean Absolute Error (MAE) for volume prediction and Binary Cross-Entropy (BCE) loss for domain classification and they are expressed by the formulas below:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{5.8}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{5.9}$$

$$BCE = \frac{-1}{N}\sum_{i=1}^{N}[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)] \tag{5.10}$$

Where N is sample number; y_i and $\hat{y}_i$ are ground-truth value and predicted value respectively.

5.3. Source Domain Experiment

Table 9 covers the performances of models using RNN as predictors on Antwerp's dataset. It finds that PGNN-CNN-RNN outperformed others. Its MSE was significantly lower than that of RNN, meaning that the module consisting of PGNN and CNN could extract spatio-temporal correlations from data more effectively than RNN. Also, the MAE value of PGNN-CNN-RNN was 28% lower than RNN's counterpart. Interestingly, the performance difference between PGNN-CNN-RNN and ChebConv-RNN seems much smaller. The MSE value of ChebConv-RNN was only 5% larger than that of PGNN-CNN-RNN and their gap in MAE value was also slight. One possible explanation is that ChebConv model could effectively extract the spatial dependencies between critical nodes at each time step but it did not consider the potential connections along temporal dimension.

In short, it knows that PGNN-CNN module can extract spatio-temporal patterns from the volume data disrupted by COVID-19 more effectively and it seems to have more robustness against the impacts from external factors. Compared to Chebyshev convolutional operation, PGNN-CNN can not only catch topological patterns of road network and information flowing characteristics in the graph but also establish the connectivity between nodes at different time steps.

Table 9 Loss values of Antwerp with RNN predictor

	MSE	**RMSE**	**MAE**
PGNN-CNN-RNN	2208.4	47.0	29.5
ChebConv-RNN	2324.4	48.2	31.0
RNN	3062	55.3	38.0

Figure 22 shows the predicting results of seven-day accumulated volumes in Antwerp. Figure 22 (a) reflects the spatial distribution of true labels and traffic volumes were mainly distributed evenly at the city center. It can notice that relatively high volumes, which exceeded 800 vehicles, distributed along rings outside the city center. In Figure 22 (b),

PGNN-CNN-RNN caught such spatial correlations in some way and the volumes over 800 vehicles can be found at the outside rings. ChebConv-RNN's predicting results were shown in Figure 22 (c). It also finds that relatively high volumes appeared at outside rings while relatively low volumes were mainly distributed at city center and this feature seems consistent with the ground truth.

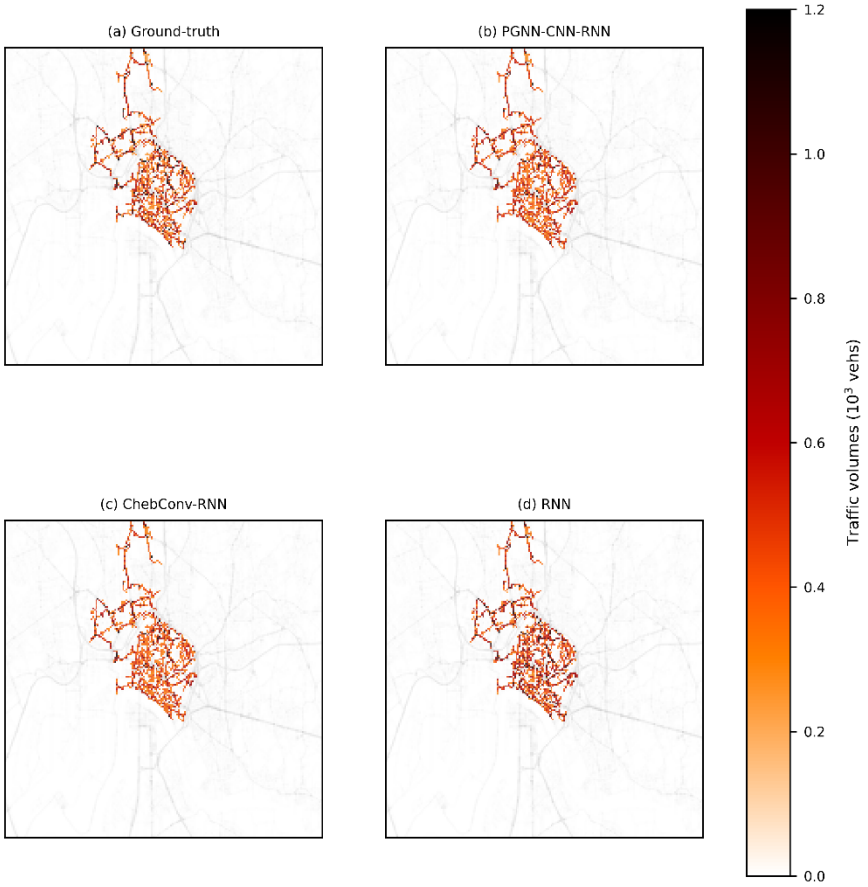

Figure 22 Seven-day accumulated volumes in Antwerp with RNN predictor

Figure 23 visualizes the Mean Absolute Percentage Error (MAPE) values on spatial aspect. It finds that in Figure 23 (c), predicting errors were relatively high at city center for RNN since RNN architecture focuses on the status of nodes themselves along time axis and does not consider the spatial correlations between nodes. Figure 23 (b) depicts the error distribution of ChebConv-RNN, Cheb-Conv also seems difficult to extract the spatio-temporal patterns at city center and possible reason is that an oversmoothing problem exists. Specifically, most critical nodes at the city center had similar volume values and

similar topological structures because they were densely distributed and common graphic convolution operation is difficult to distinguish them.

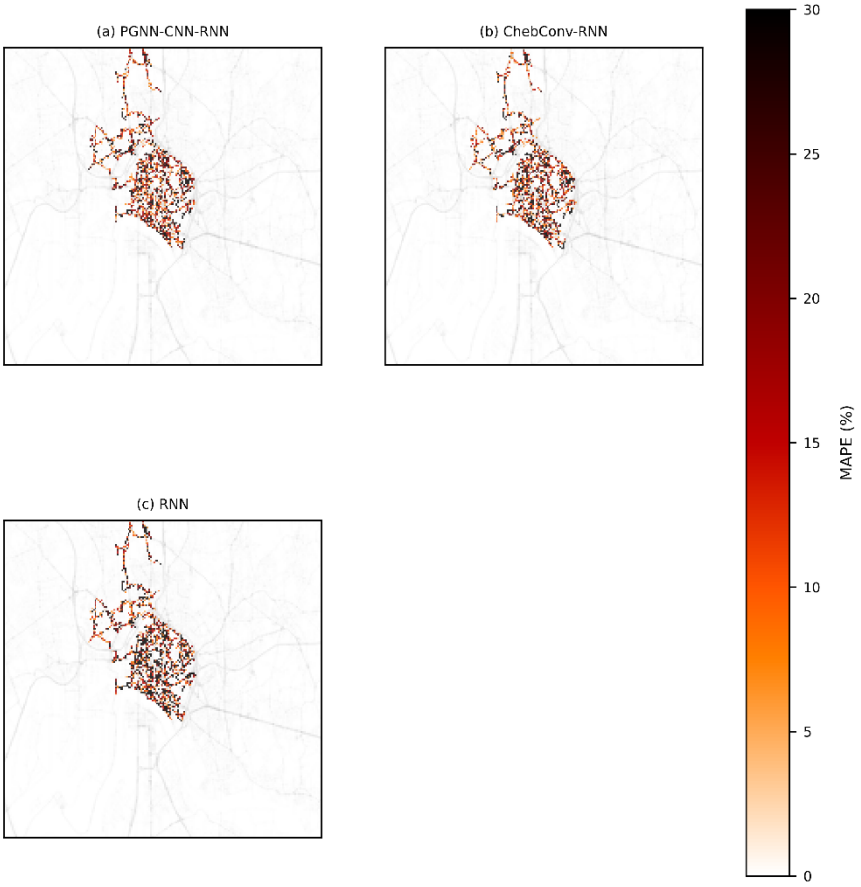

Figure 23 MAPE values of Antwerp for RNN predictor

Table 10 shows the model performances in Antwerp using LSTM as predictor. It knows that PGNN-CNN-LSTM owned smaller loss function values than the rest models. Considering MSE, ChebConv-LSTM had the worst value, which reaches 2504. For MAE, the value of ChebConv-LSTM was slightly lower than LSTM's value but 11% larger than PGNN-CNN-LSTM's counterpart. Compared with PGNN-CNN-RNN architecture, PGNN-CNN-LSTM had a smaller MSE value while its MAE value was almost same as PGNN-CNN-RNN's value. Therefore, it knows that the performance difference between PGNN-CNN-LSTM and PGNN-CNN-RNN was subtle but ChebConv-LSTM was worse than ChebConv-RNN because both MSE and MAE values of ChebConv-LSTM were larger. The predicting results of three models are visualized in Figure 24.

Table 10 Loss values of Antwerp with LSTM predictor

	MSE	RMSE	MAE
PGNN-CNN-LSTM	2143.3	46.3	29.7
ChebConv-LSTM	2504.8	50.0	33.5
LSTM	2392.0	48.9	34.0

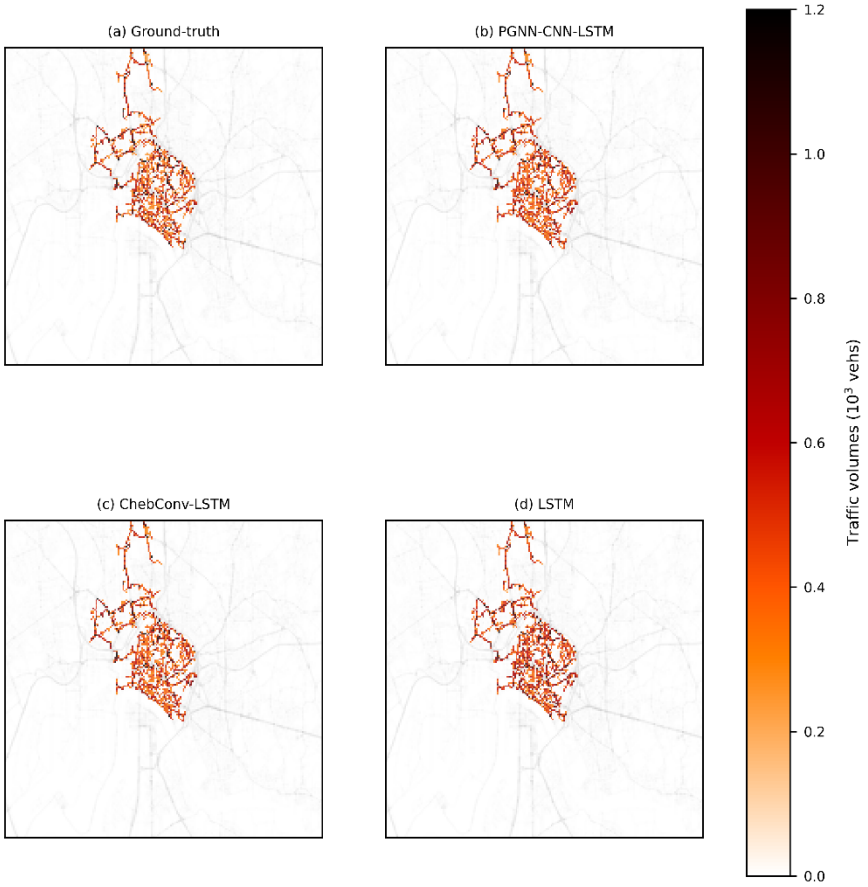

Figure 24 Seven-day accumulated volumes in Antwerp with LSTM predictor

Figure 25 shows the MAPE distribution while using LSTM as predictor. Similar to the situation in Figure 25, MAPE values were still relatively large at city center and it reflects that GNN-based networks might be difficult to learn spatio-temporal patterns from the region where nodes are with similar features and highly-overlapped neighbors.

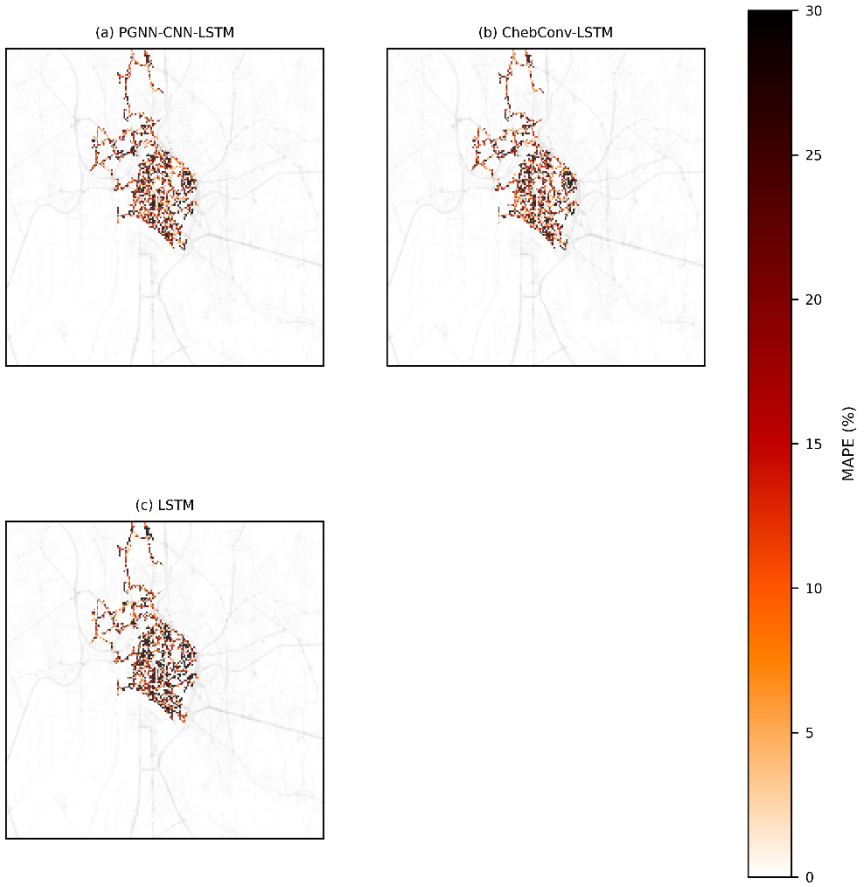

Figure 25 MAPE values of Antwerp with LSTM predictor

Table 11 encapsulates the predicting results of models taken RNN as predictor in Bangkok. It shows PGNN-CNN-RNN had the smallest loss values among three models and its MSE was 7% and 36% lower than that of ChebConv-RNN and RNN respectively. The differences in MAE seemed smaller and PGNN-CNN-RNN's value was slightly lower than the rest two models' counterparts. Considering ChebConv-RNN, the gap between it and RNN was significantly large in MSE; its MSE was nearly 17% smaller than RNN's value. One explanation can be that the graph structure of Bangkok might have less nodes whose neighbors are highly overlapped and it reduces the negative impacts of oversmoothing on GNN-based models.

Table 11 Loss values of Bangkok with RNN predictor

	MSE	RMSE	MAE
PGNN-CNN-RNN	1868.2	43.2	29.6
ChebConv-RNN	2006.4	44.7	31.0
RNN	2542.0	50.4	37.0

Figure 26 displays the predicting results of these three models. Figure 26 (a) demonstrates the spatial distribution of true labels. The traffic volumes exceeding one thousand vehicles can be found in the southern area of the road network and partial intersections at city center while the rest of road network had relatively lower volumes. PGNN-CNN-RNN's results are shown in Figure 26 (b) and model caught the spatial patterns mentioned above in some way. Figure 26 (c) and Figure 26 (d) are ChebConv-RNN and RNN predicting results separately.

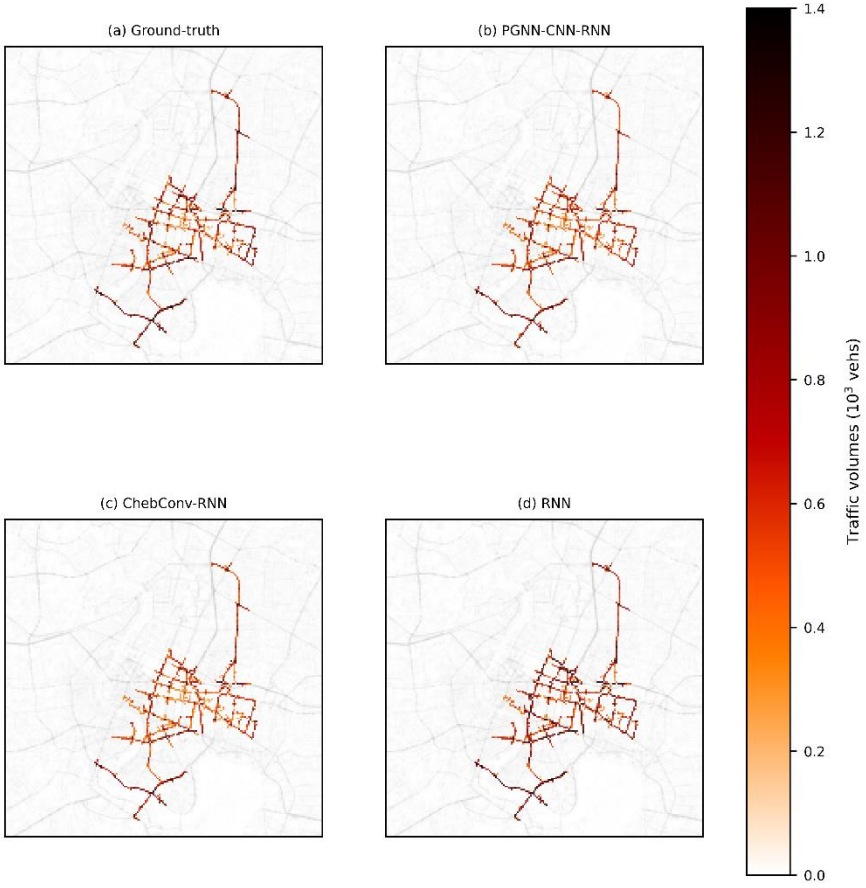

Figure 26 Seven-day accumulated volumes in Bangkok with RNN predictor

Figure 27 reflects the MAPE distribution spatially on the road network. For PGNN-CNN-RNN, it finds that relatively high MAPE values mainly distributed at the central area of road network where it has more intersections. The reason might be that the spatio-temporal patterns of that region are more complicated than other areas and would be caught difficulty by model. In Figure 27 (b), relatively high MAPE values distributed more densely at the central area, reflecting that ChebConv-RNN could not effectively extract spatio-temporal patterns and proved the superiority of PGNN-CNN. Figure 27 (c) shows the worst the performance among them. RNN nearly could not capture spatial-temporal patterns from road network's central area.

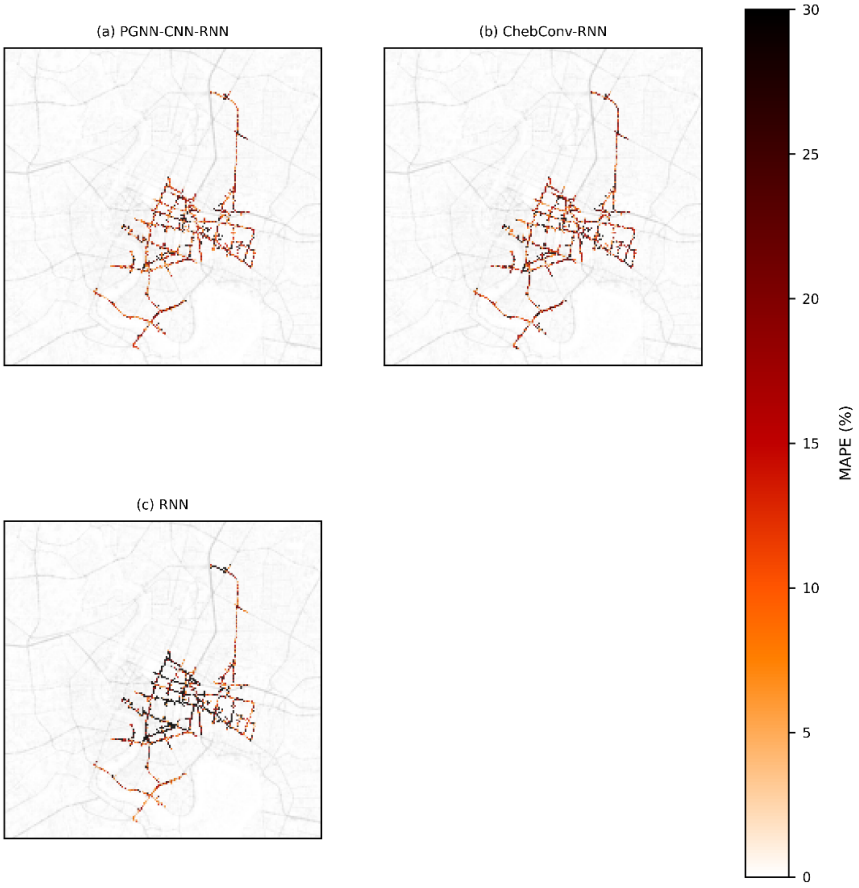

Figure 27 MAPE values of Bangkok with RNN predictor

Table 12 covers model performances on Bangkok dataset with LSTM predictor. For MSE, ChebConv-LSTM was with the worst value, followed by LSTM. The MAE differences among three models were not large but the lowest MAE value still belonged to PGNN-CNN-LSTM. It finds that the predictor type does not influence model performance greatly

compared to Table 12. The reason is that in 1D CNN, 1D convolutional operation extracts long-term temporal patterns in advance and sends the processed data to predictor. This process could make up one shortcoming of RNN and LSTM that they can not remember long-term temporal features existed in raw data. Figure 28 shows the visualized outcomes.

Table 12 Loss values of Bangkok with LSTM predictor

	MSE	**RMSE**	**MAE**
PGNN-CNN-LSTM	1880.5	43.4	30.1
ChebConv-LSTM	2147.6	46.3	33.7
LSTM	1999.0	44.7	32.0

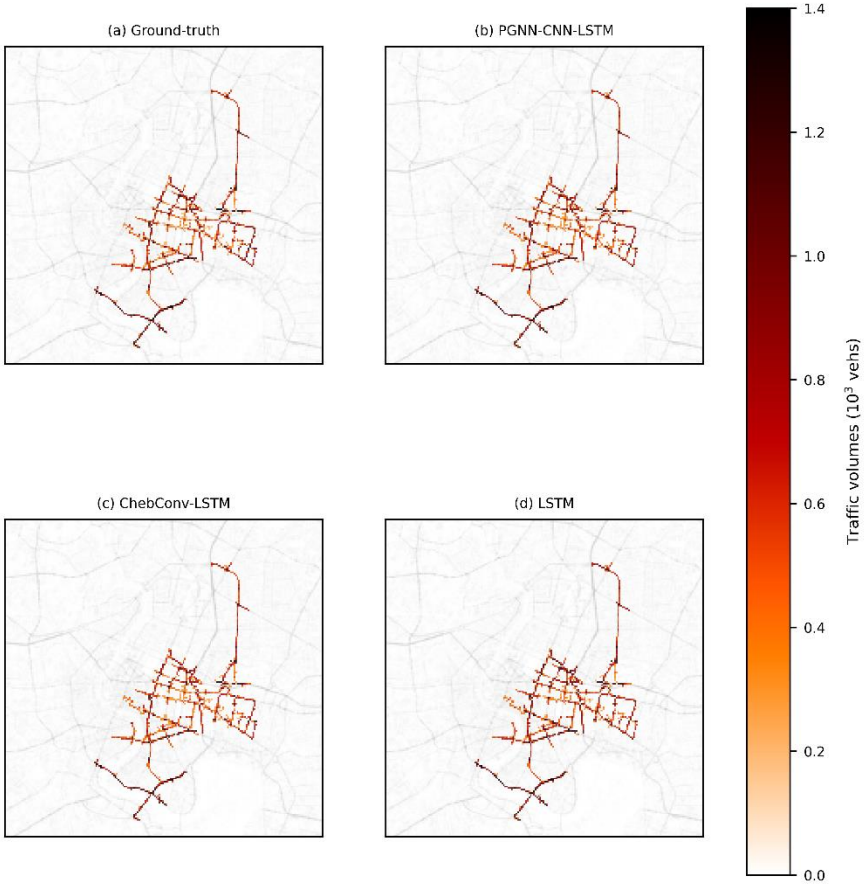

Figure 28 Seven-day accumulated volumes in Bangkok with LSTM predictor

MAPE value distribution is shown in Figure 29. Figure 29 (a) corresponds to PGNN-CNN-LSTM model and relatively high MAPE values mainly assembled at some intersections of the central area due to sophisticated spatio-temporal characteristics. In Figure 29 (b), the spatial distribution of error was similar to the situation in Figure 29 (a). When it comes to Figure 29 (c), it has no GNN and 1D CNN to provide processed spatio-temporal features to LSTM predictor, LSTM's performance unsurprisingly became the worst among three models.

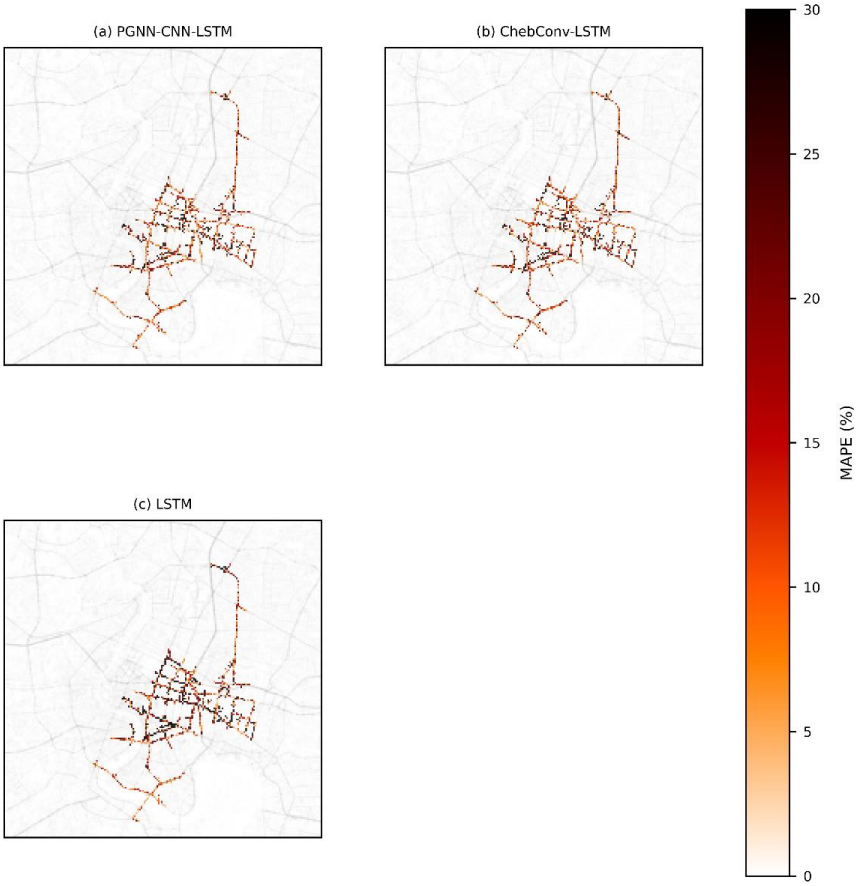

Figure 29 MAPE values of Bangkok with LSTM predictor

5.4. Target Domain Experiment

Since proposed architecture utilized a domain-adversarial strategy, the PGNN-CNN-LSTM, previously trained on source data, were combined with a domain classifier and a gradient reversal layer for domain adversarial. Besides, the PGNN-CNN-LSTM, PGNN-CNN-RNN, ChebConv-LSTM, ChebConv-RNN, LSTM and RNN, which were already

trained on source data, were fine-tuned using target data as base-line models. To avoid label imbalance, it extracted same-sized samples from source cities for domain adversarial. Table 13 summarizes the performances of proposed architecture and based-line models with RNN predictor. It sees that Domain-Adversarial Neural Network (DANN)-based PGNN-CNN-LSTM had the lowest MSE and MAE values, nearly 14% lower than that of fine-tuned PGNN-CNN-RNN, indicating that domain-adversarial process might make model to learn domain-invariant patterns more effectively than just fine-tuning model. Also, the difference between fine-tuned PGNN-CNN-RNN and fine-tuned Cheb-Conv RNN was subtle. It can be explained that data scarcity limited the performance of PGNN-CNN module and the module was difficult to extract spatio-temporal patterns through fine-tuning. Interestingly, it finds that fine-tuned RNN significantly underperformed other models. Its MSE value was nearly four times of fine-tuned PGNN-CNN-RNN's value. The RNN model might be greatly influenced by data-distribution difference and data scarcity.

Table 13 Loss values of Barcelona with RNN predictor

	MSE	**RMSE**	**MAE**
DANN-based PGNN-CNN-LSTM	1293.0	36.0	22.2
Fine-tuned PGNN-CNN-RNN	1467.2	38.3	27.3
Fine-tuned ChebConv-RNN	1431.2	37.8	26.3
Fine-tuned RNN	5864.0	76.6	60.0

It can see the volume prediction results of target city in Figure 30. True labels in Figure 30 (a) tell us that the traffic volumes of Barcelona seem obviously lower than that of Bangkok and Antwerp. Most volume values were approximately below four hundred vehicles and evenly distributed in the road network. Relatively high volumes, exceeding four hundred vehicles, can be found on the south and north side of road network. In Figure 30 (b), this pattern was caught by DANN-based PGNN-CNN-RNN. However, it seems that the architecture overestimated traffic volumes in the middle area since

adversarial-based architecture may still not overcome the differences in data distribution effectively. It can be seen in Figure 30 (c), (d) and (e) that the volume differences between central area and two sides of city were not caught by rest models effectively. Especially, it sees that in Figure 30 (e), fine-tuned RNN predicted relatively high volumes occupied nearly entire network, which was contradicted to the spatial patterns of true labels.

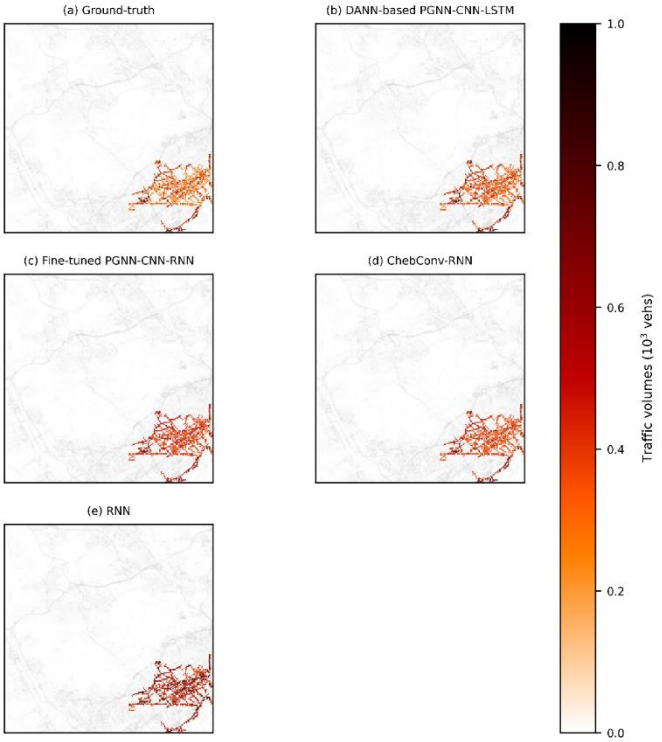

Figure 30 Seven-day accumulated volumes in Barcelona with RNN predictor

The spatial distribution of MAPE can be seen in Figure 31. It notices that MAPE values were relatively large for all models. It finds that the MAPE values in Figure 31 (a) were less than the rest parts, indicating that domain adversarial strategy was helpful for learning domain invariant features in some way.

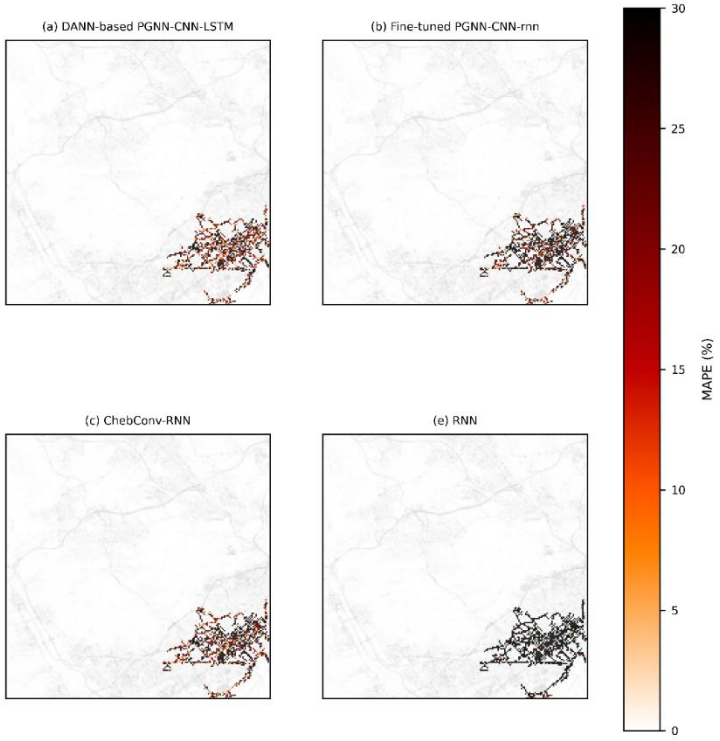

Figure 31 MAPE values of Barcelona with RNN predictor

The performance comparison between proposed architecture and the baselines with LSTM predictor is in Table 14. Although the difference between DANN-based architecture and fine-tuned PGNN-CNN-LSTM was small, DANN-based architecture performed better. By contrast, it finds that fine-tuned LSTM significantly outperformed fine-tuned RNN and LSTM's MSE value was only half of RNN's loss value. Besides, it has a significant gap between fine-tuned PGNN-CNN-LSTM and fine-tuned ChebConv-LSTM. The MSE of latter was 23% larger than that of the former. Figure 32 and Figure 33 present the predicting results with LSTM predictor and corresponding MAPE distribution separately.

Table 14 Loss values of Barcelona with LSTM predictor

	MSE	**RMSE**	**MAE**
DANN-based PGNN-CNN-LSTM	1293.0	36.0	22.2

	MSE	RMSE	MAE
Fine-tuned PGNN-CNN-LSTM	1332.5	36.5	24.0
Fine-tuned ChebConv-LSTM	1640.7	40.5	30.3
Fine-tuned LSTM	2468.0	49.6	36.0

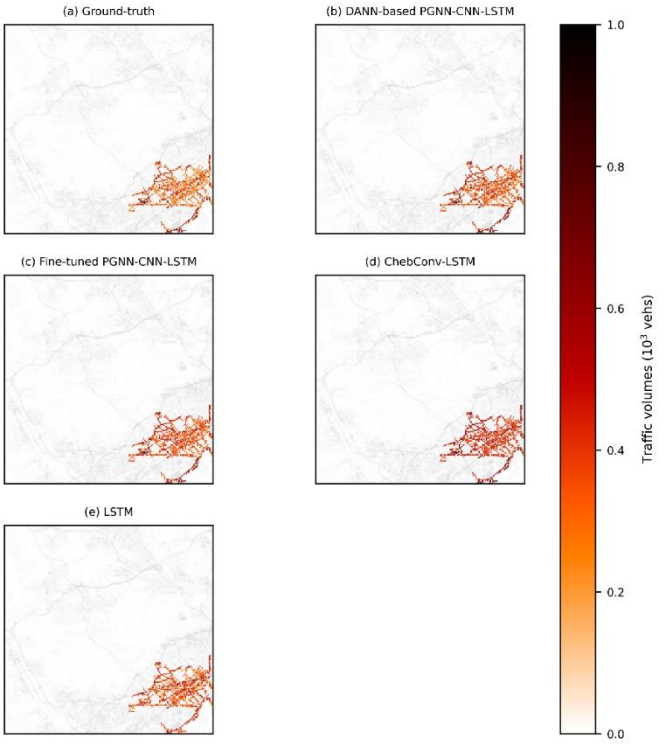

Figure 32 Seven-day accumulated volumes in Barcelona with LSTM predictor

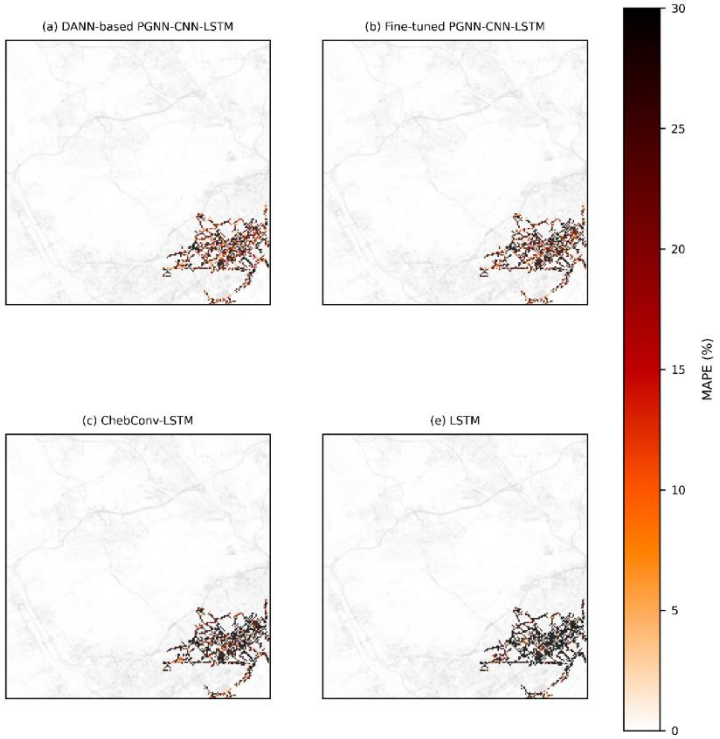

Figure 33 MAPE values of Barcelona with LSTM predictor

5.5. Sensitivity Analysis

Table 15 shows the model performance on the source cities with a 14-day prediction time window. For models using an LSTM predictor, the MSE and MAE values of the PGNN-CNN-LSTM model were significantly lower than those of other models, indicating that the PGNN-CNN-LSTM maintained strong robustness against the impact of the pandemic, even with a longer prediction time window. A similar trend was observed in models using an RNN predictor.

Table 15 Loss values of source cities (prediction time window length: 14 days)

	MSE (Antwerp)	**MSE (Bangkok)**	**MAE (Antwerp)**	**MAE (Bangkok)**
PGNN-CNN-LSTM	2463.4	1768.5	31.3	29.6
ChebConv-LSTM	2788.3	2155.4	36.5	34.4

	MSE (Antwerp)	MSE (Bangkok)	MAE (Antwerp)	MAE (Bangkok)
LSTM	2906.0	2386.0	37.0	35.0
PGNN-CNN-RNN	2580.9	1816.6	31.2	29.2
ChebConv-RNN	2711.1	2179.8	33.2	32.5
RNN	3022.0	2313.0	35.0	35.0

Table 16 summarizes the model performance on the target city. The MSE of the DANN-based PGNN-CNN-LSTM model was below one thousand, outperforming even the results achieved with a 7-day prediction time window. Among the models using an LSTM predictor, performance differences were minor, except for the standalone LSTM model, whose MSE was nearly double that of the others.

Table 16 Loss values of target city (prediction time window length: 14 days)

	MSE (Barcelona)	MAE (Barcelona)
DANN-based PGNN-CNN-LSTM	982.1	19.4
Fine-tuned PGNN-CNN-LSTM	1064.8	20.5
Fine-tuned Cheb-Conv-LSTM	1177.0	25.7
Fine-tuned LSTM	2364.0	37.0
Fine-tuned PGNN-CNN-RNN	1035.4	20.4
Fine-tuned Cheb-Conv-RNN	1027.4	21.4
Fine-tuned RNN	3098.0	44.0

6. Conclusion

To capture temporal pattern shifts in traffic volume during emergency, this thesis proposes a method that leverages distinct deep-learning models for spatial correlation extraction and spatio-temporal pattern fusion. Experimental results demonstrate the effectiveness of the proposed model structure.

6.1. Contributions and Findings

This study used traffic volume data from a simplified road network as input, extracting spatio-temporal patterns through a feature extractor, with the output then utilized for volume prediction. The proposed model employed a domain-adversarial strategy and consisted of three main components: a feature extractor, a domain classifier, and a predictor. The feature extractor included a PGNN and a 1D-CNN, connected sequentially. The domain classifier was implemented as a MLP and the predictor was a LSTM model. Additionally, the study compared the performance of the proposed structure with baseline models and analyzed potential reasons for the differences.

From a structural perspective, most existing literature employs Chebyshev convolution; however, this study utilized position-aware message aggregation to capture spatial correlations in traffic volume and incorporated a 1D CNN stacked on top of PGNN. Experimental results demonstrated that the PGNN-CNN module outperformed Chebyshev convolution and confirmed that the 1D CNN effectively fused spatio-temporal patterns. Additionally, the study examined the impact of different predictors on prediction performance and found that the LSTM predictor was not consistently superior to the RNN predictor, especially in models incorporating a 1D CNN. This finding suggests that the feature extractor effectively condensed the temporal information of long time series, compensating for the limitations of the RNN structure.

Considering transfer learning, this study compared the performance of domain adversarial and fine-tuning, finding that domain adversarial performed better. This suggests that domain adversarial training can adapt more quickly to new data distributions.

6.2. Limitations

1. During data preprocessing, it simplified entire road network by filtering nodes whose daily volume was less or over the threshold. The threshold value determination was intuitive and it had no unified standard in different cities. Correspondingly, the simplification process might loss critical topological information of network and degrade GNN performance.

2. This study only took the traffic volumes during pandemic as model input and did not consider the pattern differences between pre-pandemic volumes. This method could not measure the disturbance of pandemic on traffic volume effectively. Specifically, feature extractor might be difficult to learn irregular pattern shift without any reference data and it would become more challenging if the amount of source city goes larger. Therefore, pre-pandemic data would be necessary for measuring impact of the pandemic on volume data.

3. The domain adversarial method adopted in this study was that domain classifier tries to judge spatio-temporal patterns, generated by feature extractor, belonging to source or target domain correctly and feature extractor tries to generate domain-invariant patterns and cheats the classifier. However, this methodology is mainly used in classification mission and may not fit regression mission very well. It might be more feasible to measure difference between different data distributions by distance-based criteria.

6.3. Future Research Directions

1. It could propose a feasible network simplification methodology that keeps critical information of road network and removes redundant nodes so it does not need remove redundant nodes by intuition but promotes performance and training efficiency of GNN.

2. Statistical-based or machine learning method such as Analysis of Variance (ANOVA) and Bayesian inference method can be used to measure the disruption of public

emergency on traffic volume and the results act as auxiliary information of predicting model.

3. Domain adversarial for regression mission can adopt distance-based criteria like Wasserstein distance so domain classifier is not needed and the loss function of feature extractor includes regression loss and the Wasserstein distance between different domains. Feature extractor is able to minimize regression error and the divergence between different data distributions by itself.

References

Bai, Lei, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. “Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting.”

Beck, Matthew J., and David A. Hensher. 2020. “Insights into the Impact of COVID-19 on Household Travel and Activities in Australia – The Early Days of Easing Restrictions.” *Transport Policy* 99:95–119. doi: https://doi.org/10.1016/j.tranpol.2020.08.004.

Das, Sanhita, Alice Boruah, Arunabha Banerjee, Rahul Raoniar, Suresh Nama, and Akhilesh Kumar Maurya. 2021. “Impact of COVID-19: A Radical Modal Shift from Public to Private Transport Mode.” *Transport Policy* 109:1–11. doi: https://doi.org/10.1016/j.tranpol.2021.05.005.

Drift, Sander, Luc Wismans, and Marie-José Olde Kalter. 2021. “Changing Mobility Patterns in the Netherlands during COVID-19 Outbreak.” *Journal of Location Based Services* 16:1–24. doi: 10.1080/17489725.2021.1876259.

Geng, Xu, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu. 2019. “Spatiotemporal Multi-Graph Convolution Network for Ride-Hailing Demand Forecasting.” Pp. 3656–63 in *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33.

Guo, Shengnan, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. 2022. “Learning Dynamics and Heterogeneity of Spatial-Temporal Graph Data for Traffic Forecasting.” *IEEE Transactions on Knowledge and Data Engineering* 34(11):5415–28. doi: 10.1109/TKDE.2021.3056502.

Harris, M., and Michael Branion-Calles. 2021. “Changes in Commute Mode Attributed to COVID-19 Risk in Canadian National Survey Data.” *Transport Findings*. doi: 10.32866/001c.19088.

HERE. 2021. “Sample Map Data for Students | HERE.” Retrieved October 28, 2024 (https://www.here.com/developer/sample-map-data).

Huang, Songtao, Hongjin Song, Tianqi Jiang, Akbar Telikani, Jun Shen, Qingguo Zhou, Binbin Yong, and Qiang Wu. 2024. “DST-GTN: Dynamic Spatio-Temporal Graph Transformer Network for Traffic Forecasting.”

Huang, Yunjie, Xiaozhuang Song, Shiyao Zhang, and James J. Q. Yu. 2021. “Transfer Learning in Traffic Prediction with Graph Neural Networks.” Pp. 3732–37 in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*.

Li, Junyi, Fangce Guo, Yibing Wang, Lihui Zhang, Xiaoxiang Na, and Simon Hu. 2020. “Short-Term Traffic Prediction with Deep Neural Networks and Adaptive Transfer Learning.” Pp. 1–6 in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece: IEEE.

Li, Junyi, Kaihang Zhang, Loutao Shen, Zhebing Wang, Fangce Guo, and Panagiotis Angeloudis. 2021. “A Domain Adaptation Framework for Short-Term Traffic Prediction.”

Liu, Haiyang, Chunjiang Zhu, Detian Zhang, and Qing Li. 2023. “Attention-Based Spatial-Temporal Graph Convolutional Recurrent Networks for Traffic Forecasting.”

Liu, Hao. 2022. “Decoupled Traffic Spatial-Temporal Graph Neural Network for Traffic Flow Prediction.” Pp. 987–90 in *2022 4th International Academic Exchange Conference on Science and Technology Innovation (IAECST)*. Guangzhou, China: IEEE.

Liu, Lu, Yibo Cao, and Yuhan Dong. 2023. “Spatial-Temporal Interactive Graph Neural Network for Traffic Forecasting.” Pp. 1174–79 in *2023 8th International Conference on Computer and Communication Systems (ICCCS)*. Guangzhou, China: IEEE.

Mo, Jiqian, and Zhiguo Gong. 2023. “Cross-City Multi-Granular Adaptive Transfer Learning for Traffic Flow Prediction.” *IEEE Transactions on Knowledge and Data Engineering* 35(11):11246–58. doi: 10.1109/TKDE.2022.3232185.

Muggenthaler, Philip, Joachim Schroth, and Yiqiao Sun. 2021. “The Heterogeneous Economic Impact of the Pandemic across Euro Area Countries.”

Pan, Sinno Jialin, and Qiang Yang. 2010. “A Survey on Transfer Learning.” *IEEE Transactions on Knowledge and Data Engineering* 22(10):1345–59. doi: 10.1109/TKDE.2009.191.

Shi, Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting.” *Advances in Neural Information Processing Systems* 28.

Song, Chao, Youfang Lin, Shengnan Guo, and Huaiyu Wan. 2020. “Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting.” *Proceedings of the AAAI Conference on Artificial Intelligence* 34(01):914–21. doi: 10.1609/aaai.v34i01.5438.

Tang, Yihong, Ao Qu, Andy H. F. Chow, William H. K. Lam, S. C. Wong, and Wei Ma. 2022. “Domain Adversarial Spatial-Temporal Network: A Transferable Framework for Short-Term Traffic Forecasting across Cities.”

Tsai, Meng-Ju, Hsiu-Yuan Chen, Zhiyong Cui, and Yinhai Wang. 2021. “Multivariate Long And Short Term LSTM-Based Network for Traffic Forecasting Under Interference: Experiments During COVID-19.” Pp. 2169–74 in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. Indianapolis, IN, USA: IEEE.

Wang, Leye, Xu Geng, Xiaojuan Ma, Feng Liu, and Qiang Yang. 2018. “Cross-City Transfer Learning for Deep Spatio-Temporal Prediction.”

Wei, P., Y. Cao, and D. Sun. 2013. “Total Unimodularity and Decomposition Method for Large-Scale Air Traffic Cell Transmission Model.” *Transportation Research Part B: Methodological* 53:1–16. doi: https://doi.org/10.1016/j.trb.2013.03.004.

Worldometer. 2023. “COVID - Coronavirus Statistics - Worldometer.” Retrieved November 1, 2024 (https://www.worldometers.info/coronavirus/?utm_campaign=homeAdvegas1?#google_vignette).

Wu, Qiong, Kaiwen He, Xu Chen, Shuai Yu, and Junshan Zhang. 2022. “Deep Transfer Learning Across Cities for Mobile Traffic Prediction.” *IEEE/ACM Transactions on Networking* 30(3):1255–67. doi: 10.1109/TNET.2021.3136707.

Wu, Zonghan, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. “Graph WaveNet for Deep Spatial-Temporal Graph Modeling.”

Xu, F. F., Z. C. He, and Z. R. Sha. 2013. “Impacts of Traffic Management Measures on Urban Network Microscopic Fundamental Diagram.” *Jiaotong Yunshu Xitong Gongcheng Yu Xinxi/Journal of Transportation Systems Engineering and Information Technology* 13:185–90.

Xu, Xin-yue, Jun Liu, Hai-ying Li, and Jian-Qiang Hu. 2014. “Analysis of Subway Station Capacity with the Use of Queueing Theory.” *Transportation Research Part C: Emerging Technologies* 38:28–43. doi: https://doi.org/10.1016/j.trc.2013.10.010.

You, Jiaxuan, Rex Ying, and Jure Leskovec. 2019. “Position-Aware Graph Neural Networks.”

Yu, Bing, Haoteng Yin, and Zhanxing Zhu. 2018. “Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting.” Pp. 3634–40 in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*.

Zhang, Chuanting, Haixia Zhang, Jingping Qiao, Dongfeng Yuan, and Minggao Zhang. 2019. “Deep Transfer Learning for Intelligent Cellular Traffic Prediction Based on Cross-Domain Big Data.” *IEEE Journal on Selected Areas in Communications* 37(6):1389–1401. doi: 10.1109/JSAC.2019.2904363.

Zhang, Jinlei, Feng Chen, Yinan Guo, and Xiaohong Li. 2020. “Multi-Graph Convolutional Network for Short-Term Passenger Flow Forecasting in Urban Rail Transit.” *IET Intelligent Transport Systems* 14(10):1210–17. doi: https://doi.org/10.1049/iet-its.2019.0873.

Zhang, Junbo, Yu Zheng, and Dekang Qi. 2017. “Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction.”

Zhang, Yingxue, Yanhua Li, Xun Zhou, Xiangnan Kong, and Jun Luo. 2022. “STrans-GAN: Spatially-Transferable Generative Adversarial Networks for Urban Traffic Estimation.” Pp. 743–52 in *2022 IEEE International Conference on Data Mining (ICDM)*.

Zhao, Ling, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. 2020. “T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction.” *IEEE Transactions on Intelligent Transportation Systems* 21(9):3848–58. doi: 10.1109/TITS.2019.2935152.

Zheng, Chuanpan, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2019. “GMAN: A Graph Multi-Attention Network for Traffic Prediction.”

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.

Munich, 2024.10.29, Zhaofan Chen

Place, Date, Signature