BOURNEMOUTH UNIVERSITY

DEPARTMENT OF COMPUTING AND INFORMATICS

Deep Learning Models for Traffic Prediction in Urban Transport Networks

Author:
Ge Zheng

Supervisor:
Dr. Wei Koong Chai

Co-Supervisor:
Professor Vasilis Katos

This thesis is submitted for the degree of

Doctor of Philosophy

June, 2022

Acknowledgements

I would like to thank my supervisor team consisting of Dr. Wei Koong Chai and Professor Vasilis Katos for their expert advice and encouragement throughout this project, especially for Dr. Wei Koong Chai who is always very patient and kind when I have some problems during my PhD study. I also would like to thank my parents for encouraging me always to be positive in my life. Thanks for every staff who supports my study at Bournemouth University.

Declaration

This thesis is submitted for the degree of Doctor of Philosophy at Bournemouth University, United Kingdom. I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 40,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Abstract

With the development of smart technologies built on big data analysis, the vision of smart city, that aims to improve quality of life, reduce energy consumption and pollution and drive economic growth, has been enhanced. One important component of this vision is the urban transport network which is continuously challenged with increasing number of vehicles on roads. This results in issues including long travel time, increasingly persistent traffic congestion, accidents and road safety concerns. The research work in this thesis aims to predict traffic in urban transport networks which are capable of solving those issues.

Understanding and control of traffic patterns are fundamental towards this aim. For this, we develop a short-term traffic flow prediction model, named EM, on linear roadways based on machine learning technology. EM is able to analyse and extract spatial and temporal features from original traffic data for the final prediction. This short-term traffic prediction could give drivers traffic situation in advance and guide them to avoid congested roads to reduce travel time and also relieve traffic congestion.

Moreover, armed with accurate short-term traffic prediction models on linear roadways, we further develop a novel deep learning model, named ALLSCP, for short-term traffic flow prediction on intersections where traffic situations are more complex compared to linear roadways. The difference to EM is that ALLSCP can analyse more detailed features (i.e. global- and local-spatial and short-, medium- and long-term temporal features) for the final prediction. Due to the fact that traffic flows are almost controlled by traffic lights in most cities around the world, this could benefit the optimisation of traffic light strategies so that more vehicles are allowed to pass in a short duration.

After analysing traffic data on linear roadways and intersections, we consider and solve traffic prediction problem on large-scale road networks and develop a deep learning model (named SAGCN-SST) based on Graph Convolution Network (GCN) and Attention mechanism for multiple traffic speed prediction on large-scale road networks. This SAGCN-SST is able to capture dynamic-spatial and temporal features for the final prediction. The predicted traffic speed on large-scale road networks could be used to find vulnerable roads and then optimise routing strategies for reducing traffic congestion and long travel time.

Furthermore, we design another deep learning model based on Virtual Dynamic Graph Convolution Network and Transformer with Gate and Attention mechanisms (VDGCNTGA) for multi-step traffic speed prediction on large-scale road networks, which can explore dynamic and hidden spatial and temporal features in network-wide for the final prediction. VDGCNTGA has ability to generate and update a virtual dynamic road graph each batch to represent dynamic and hidden spatial dependencies that are not described in the real road graph. Compared to our last SAGCN-SST model, VDGCNTGA can exploit more useful hidden spatial dependencies of road segments in network-wide.

Finally, we develop a deep learning model based on Sequence-to-Sequence architecture with an embedded module using Graph Convolution Neural Network and Transformer, named GCNT-Seq2Seq, for long-term traffic speed prediction in large-scale road networks. This model is able to analyse and extract local- and global-spatial and long-term temporal features for the final prediction, which benefits the optimisation of traffic strategies for improving road network efficiency.

For this research, we use real transportation data collected from urban transport networks of different characteristics to evaluate our proposed models and also compare them against well-known existing traffic prediction models.

Keywords: Urban Transport Networks, Traffic Prediction, Deep Learning, Data Analysis, Linear Roadways, Intersections, Large-scale Road Networks

Contents

List of Figures

List of Tables

Glossary of Terms

ACF	AutoCorrelation Function
ANN	Artificial Neural Network
AR	Auto-Regressive
ARIMA	Auto-Regressive Integrated Moving-Average
ARMA	Auto-Regressive Moving-Average
Bi-LSTM	Bidirectional Long Short Term Memory
CAPSNET	Capsule Neural Network
CNN	Convolution Neural Network
CPU	Central Processing Unit
DA	Data Aggregation
DAE	Deep Auto-Encoder
DCNNs	Deep Convolutional Neural Networks
DCRNN	Diffusion Convolutional Recurrent Neural Network
DL	Deep Learning
DLM	Dynamic Linear Model
FARIMA	Fuzzy Autoregressive Integrated Moving-Average

FFNN	Feed Forward Neural Network
GCN	Graph Convolution Network
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HP	Hodrick Prescott
IRNN	Identity RNN
ITSs	Intelligent Transportation Systems
KARIMA	Kohonen map with Auto-Regressive Integrated Moving-Average
KNN	K-Nearest Neighbours
LSTM	Long Short Term Memory
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
Max	Maximum
Min	Minimum
ML	Machine Learning
MSE	Mean Square Error
NN	Neural Network
PACF	Partial AutoCorrelation Function
PeMS	Performance Measurement System

POIs	Points of Interest
RBF	Radial Basis Function
RMSE	Root-Mean Square Error
RNN	Recurrent Neural Network
SAE	Stacked AutoEncoder
SARIMA	Seasonal Auto-Regressive Integrated Moving-Average
SEKNN	Special Event-based K-Nearest Neighbours
Seq2Seq	Sequence-to-Sequence
Std	Standard deviation
STTNs	Spatial-Temporal Transformer Networks
SVM	Support Vector Machine
SVR	Support Vector Regression
Var	Variance
WHO	World Health Organisation

Chapter 1

Introduction

1.1 Background and Motivation

In recent years, with the economic growth, vehicles are more affordable, which results in the increasing number of vehicles on roads. According to the report from Department for Transport in United Kingdom (UK), there are around 32.7 million passenger cars in operation in the United Kingdom in 2021, an increase of 5.5 million from 2000. In the world, there are around 1.4 billion vehicles in 2021 (Hedges n.d.).

Due to an increasing number of vehicles on roads, various traffic problems, including traffic congestion, traffic accidents and long travel time, worsen and become serious challenges in recent decades. Particularly for large urban areas, those traffic problems are worse because more job opportunities bring big challenges for mobility. Two obvious traffic problems are traffic congestion and long travel time. INRIX reported that in 2019, on average, an American citizen lost 99 hours (equivalent to USD $1,377 per driver) (INRIX n.d.*c*) and a British citizen lost 115 hours (equivalent to £893 per driver) due to traffic congestion, especially for road users in London losing 149 hours (INRIX n.d.*d*). It is projected that by the mid-21st century, the world's urban population will almost double from over 3.4 billion in 2009 to 6.4 billion in 2050 and the worst hit areas are usually urban areas, which means the traffic situation will become worse in the future (INRIX n.d.*c*). Figure 1.1 shows the top 25 cities with worst traffic congestion and their hours lost per driver in the world in 2019 (INRIX n.d.*a*), in which per driver lost more than 119 hours due to traffic congestion. From 2020, Covid-19 has spread all over the world. People has been encouraged to work from homes, which results in an decreasing number of vehicles on roads. Even so, traffic congestion still exists, and per driver from the top 25 cities

with worst traffic congestion in the world still lost more than 50 hours (INRIX n.d.*b*). This situation can be observed in Figure 1.2. In addition, traffic accidents also bring a big challenge to the world. According to the report from World Health Organisation (WHO) (Organisation n.d.), approximately 1.3 million people die every year because of traffic accidents.

Apart from the death resulted from traffic accidents, there are also some negative influences, like air pollution and extra fuel costs, from traffic congestion and long travel time. For air pollution, according to Transport and Environment Statistics 2021 Annual Report (Seymour n.d.), transport contributed a substantial portion of air pollutants to UK's domestic total: 34% of NO_X emissions, 13% of $PM_{2.5}$ emissions, and 11% of PM_{10} emissions came from transport in 2019. For fuel costs, (Treiber et al. 2008) found that there are an increase of fuel consumption of the order of 80% due to traffic congestion.

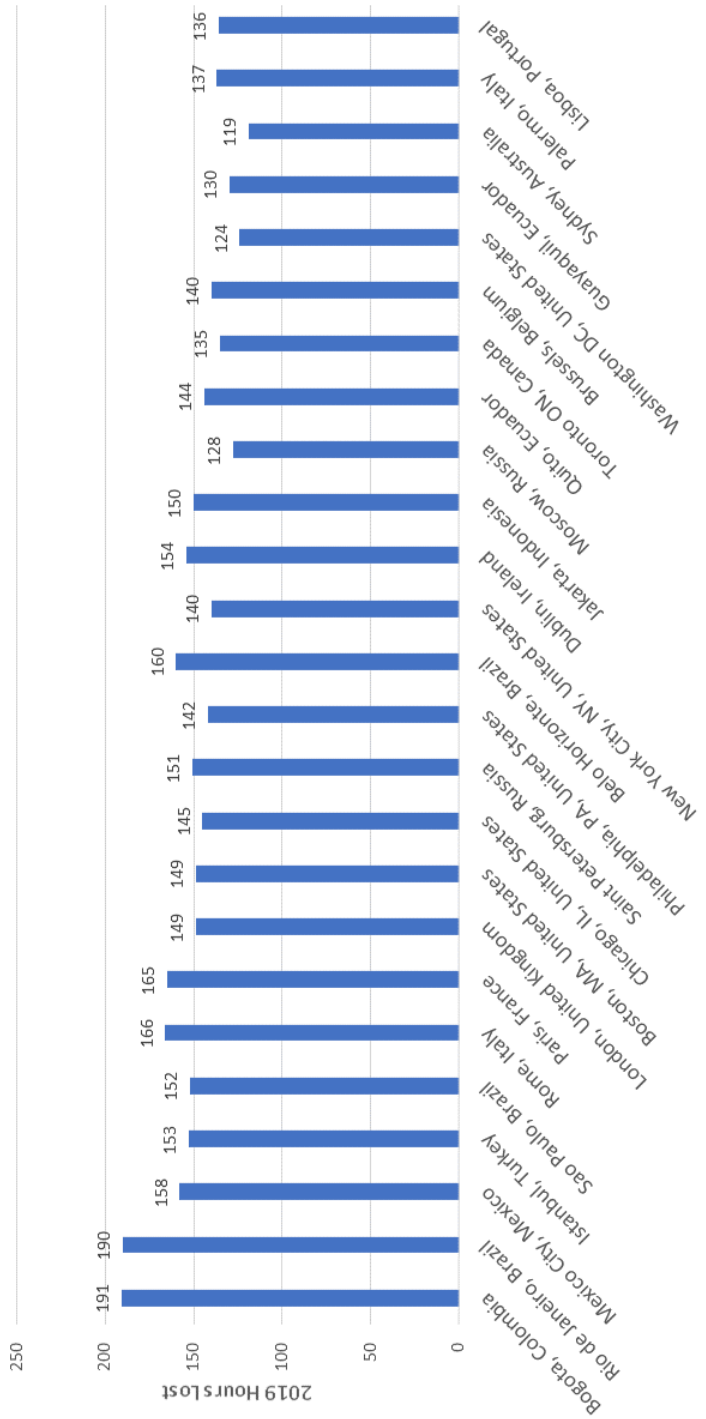


Figure 1.1: Top 25 cities with the worst traffic situations in the world in 2019.

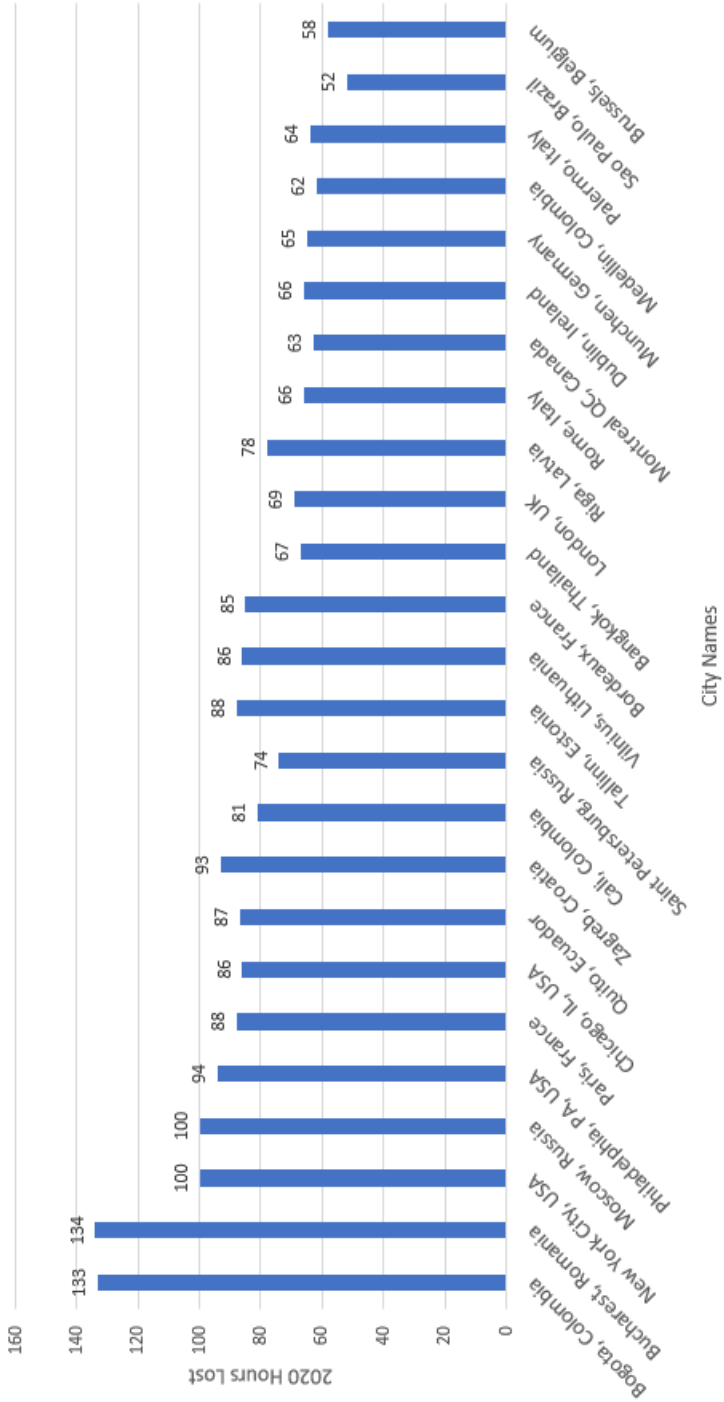


Figure 1.2: Top 25 cities with the worst traffic situations in the world in 2020.

To solve traffic problems mentioned above and reduce their negative influences to the world, the advanced Intelligent Transportation Systems (ITSs) (Vaa et al. 2007) have been developed for aiming to provide synchronous traffic information to achieve traffic efficiency by minimising traffic problems including traffic congestion, accidents and long travel time. Besides, it is also used for road safety and efficient infrastructure usage. Application areas of ITSs includes Advanced Traffic Management System (ATMS), Advanced Traveller Information System (ATIS), Advanced Vehicle Control System (AVCS), Advanced Public Transportation System (APTS), Advanced Rural Transportation System (ARTS), and Advanced Commercial Vehicles Operations System (ACVOS).

All applications or systems of ITSs mentioned above are supported by traffic information collection, processing, analysis and sharing. Traffic prediction as important traffic information and fundamental of ITSs can efficiently release worse traffic situations like traffic congestion, accidents and long travel time. For example, divers can be informed by the predicted traffic states in advance so that they can select roads with smooth traffic. Besides, the predicted traffic states on the road networks can also be used to find vulnerable road segments and then help the navigator reroute so as to help divers avoid busy roads and reduce travel time. For the traffic managers, the traffic prediction, especially for long-term traffic prediction, can be used to help them make traffic strategies and decisions to optimise the road efficiency. An efficient application is that traffic prediction can be used to control traffic lights on intersections to allow more vehicles to pass in a short duration.

In recent years, many researchers have paid attention on various traffic predictions on roadways, areas and road networks. The methodologies used in those traffic prediction have evolved into using Deep Learning (DL) (LeCun et al. 2015) technique from the very earliest stage using statistical technique (Hogg et al. 2005) and the second stage using traditional Machine Learning (ML) (Jordan & Mitchell 2015) technique. This evolution is based on the development of technologies on two areas: 1) the advanced sensor technology and 2) the new generation of powerful computers. The advanced sensor technology enables traffic data to be more affordable for analysing traffic patterns and being used to optimise traffic prediction models while the new generation of powerful computer enhances the ability of computation so as to enable models to be built with million parameters for obtaining more features from traffic data. Currently, the traffic prediction problem has been solved in its third stage using deep learning technique.

The traffic problems mentioned above and the development of new technologies in sensor and computation motivate the research work of this thesis to predict traffic in urban transport networks based on DL technique. Furthermore, to summarise and clarify the logic of this research, we use Figure 1.3 to show the overall logic of background that motivates

this research work. From this figure, it is observed clearly that obvious traffic problems include traffic congestion and incidents and the extra traffic problems include environment pollution, long travel time and extra fuel costs caused by traffic congestion and incidents. Traffic prediction can provide traffic states in advance used to reroute and make traffic strategies for supporting the applications under ITSs so as to solve those traffic problems.

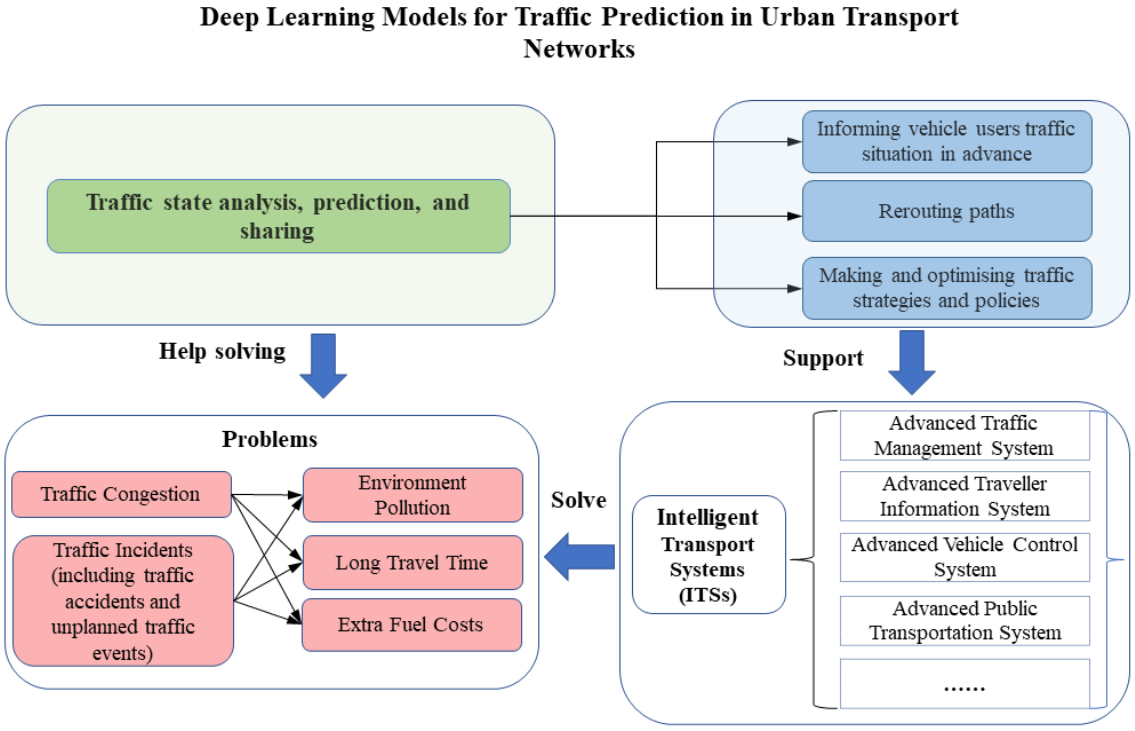


Figure 1.3: Thesis logic map.

1.2 Aims and Objectives

Considering that traffic prediction as an important role on ITSs benefits reducing travel time, traffic congestion and accidents, this PhD research project aims to develop novel deep learning models for traffic prediction in urban transport networks and offer high prediction accuracy. To achieve this aim, the objectives are as follows:

1. **Short-term traffic flow prediction on linear roadways** – To build a novel hybrid DL model for short-term traffic flow prediction on linear roadways to identify traffic situations such as traffic congestion. The ability to predict emerging traffic patterns is important for enabling many tasks in traffic management. By identifying traffic situations in advance, ITSs may make management decisions in a timely and more-informed manner. Compared to long-term traffic prediction or multi-step traffic

prediction on network-wide where traffic patterns are more complex and not easily captured, we start from analysing short-term traffic states on linear roadways. Based on short-term traffic prediction, travellers can change their travel routes on-the-fly to avoid travelling on the roads with traffic congestion to reduce travel time and not exacerbating current traffic congestion.

2. **Short-term traffic flow prediction on intersections** – To develop a novel model or an algorithm based on DL for short-term traffic flow prediction in intersections. Statistics (Yu et al. 2013) show that, in the world, traffic delays caused by traffic congestion in intersections accounted for more than 33% of total delay of urban traffic while accidents in intersections took more than 50% of all traffic accidents. Therefore, accurate traffic prediction on intersections could efficiently release those situations. Besides, an increasing number of intersections are controlled by traffic lights. A better-optimised traffic signal control method that splits traffic phases differently based on the number of waiting vehicles can reduce traffic congestion and improve traffic flow (Jafari et al. 2021). This results in smooth traffic flow that brings less challenges for both short-term and long-term traffic predictions. Meanwhile, the accurate traffic prediction can benefit the optimisation of traffic signal as well. The predicted traffic flow can be shared with traffic signal control systems and used to adjust traffic phases to maximise traffic flow and reduce traffic congestion. Therefore, accurate traffic prediction and a better-optimised traffic signal control method benefit each other.

3. **Multi-step traffic speed prediction on large-scale networks** – To develop a novel model or an algorithm based on the latest deep learning (DL) technologies for multi-step traffic speed prediction on large-scale road networks. This could benefit identifying vulnerable roads and optimising traffic routing strategies so as to improve the efficiency of the whole road network. "vulnerable roads" means to refer to road segments or junctions that are especially prone to traffic congestion during both normal operating conditions as well as on occasions when interruption events occur (e.g., road accidents or maintenance work). Predicted traffic speed on large-scale road networks can help find vulnerable roads so that the traffic department could adapt some strategies like installing traffic lights and resetting off-peak times and parking price. Besides, it also helps navigating software reroute so as to reduce travel time for drivers.

4. **Long-term traffic speed prediction on large-scale networks** – To build a novel DL model to analyse and extract specific traffic patterns for long-term traffic speed prediction on large-scale road networks. Compared to short-term traffic predic-

tion, long-term traffic prediction is more challenging due to sensitivity of error propagation. In addition, long-term traffic prediction could be better for meeting the requirement of decision making for traffic management while short-term traffic prediction mainly works for real-time traffic control. Therefore, accurate long-term traffic prediction could offer helps for traffic managers to make efficient strategies and measures to improve the traffic efficiency in network-wide.

1.3 Contributions

Based on achieved results from the PhD research work that aims to develop novel DL models for traffic prediction on urban transport networks and offer high prediction accuracy, our major contributions are as follows:

- We reviewed, categorised and summarised existing works that solve traffic prediction problems from our perspectives. Specifically, we detailed the latest well-known existing hybrid DL models by analysing their architectures and functions of inside modules. Furthermore, we also conducted comparison experiments by selecting ten representative and classical models and evaluating them on two large real-world datasets (cf. Chapter 2).

- We designed an Ensemble Model, named EM, for short-term traffic flow prediction on linear roadways and obtained high prediction accuracy. This EM model is able to analyse and extract spatial-temporal features for traffic data (including traffic flow and speed) collected from linear roadways by considering two important elements including 1) high quality data (in volume and granularity) and 2) combinations of prediction models (cf. Chapter 3).

- We developed a novel DL model based on Autoregressive integrated moving-average, two Long-short term memories, Stacked autoencoder and Capsnet , named ALLSCP, for short-term traffic flow prediction on intersections. The ALLSCP model are designed to be capable of analysing and extracting short-, medium-, and long-term temporal as well as global and local spatial features for contributing to final prediction on intersections. We evaluated our ALLSCP model on real-world traffic data collected from intersections and also tested it on collected traffic data from linear roadways to show the robustness of ALLSCP (cf. Chapter 4).

- We built a novel DL model, named SAGCN-SST (Self-Attention Graph Convolution Network with Spatial, Sub-spatial, and Temporal blocks), based on the advanced DL technologies: Graph Convolution Network (GCN) and Attention Mechanism, for multi-step traffic speed prediction on large-scale road networks. The SAGCN-SST model is developed to be able to analyse dynamic temporal-spatial features in network-wide for multi-step traffic speed prediction. We then evaluated this model on two real-world traffic datasets from different places, named `LOOP-SEATTLE` (Cui et al. 2018, 2019) from Greater Seattle and `METR-LA` (Li et al. 2018) from Los Angles in United States (cf. Chapter 5).

- We further proposed a novel Virtual Dynamic Graph Convolution Network and Transformer-based model with Gated and Attention mechanisms (VDGCNTGA) for multi-step traffic speed prediction on large-scale road networks by considering hidden spatial dependencies under real road networks and dynamic spatial-temporal features. Compared to our SAGCN-SST model, the differences are that VDGCNTGA can exploit hidden spatial dependencies of road segments in network-wide by generating the virtual graph using the attention mechanism and offer higher prediction accuracy. We then evaluated this model on two real-world and large traffic datasets from different places: `PEMS-BAY` from Bay Area and `METR-LA` from Los Angles in United States (cf. Chapter 6).

- We finally designed a DL model, named GCNT-Seq2Seq, based on the Sequence-to-Sequence architecture with an embedded module for long-term traffic speed prediction. The embedded module uses Graph Convolution Neural Network for the local spatial dependency analysis by conducting convolution operation on the $k-hop$ neighbourhood matrix while utilises Transformer for the global spatial dependency analysis by implementing the attention mechanism that assigns individual weights to neighbour road segments for contributing to the targeted road segment. We evaluated the GCNT-Seq2Seq model on a public large dataset named `METR-LA` from Los Angles in United Sates (cf. Chapter 7).

1.4 Organisation

This thesis is organised into eight chapters. The first chapter introduces the background, motivation and scope, core research works, main contributions and related publications during the whole PhD research period. The structure of this thesis is summarised in Figure 1.4.

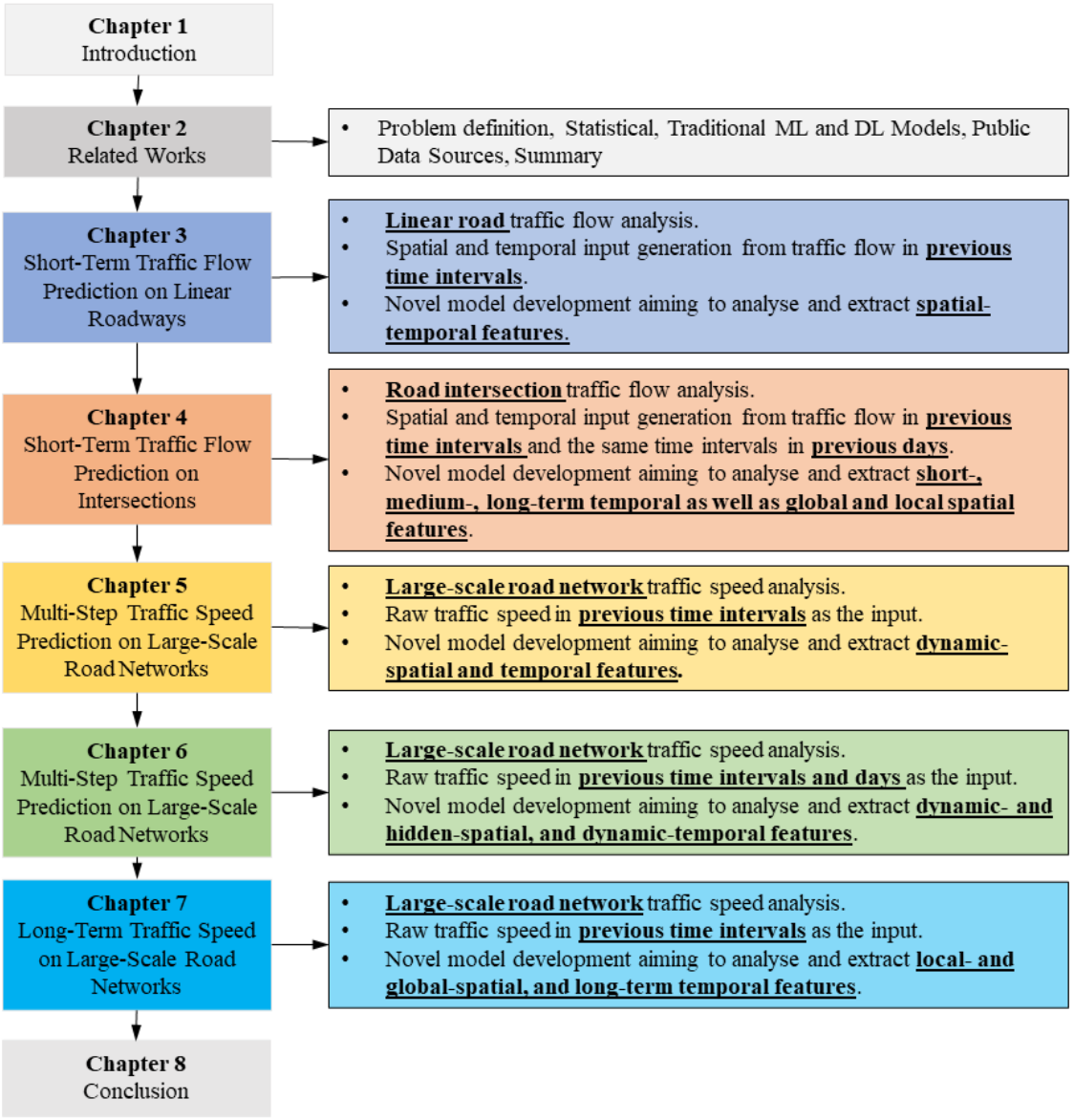


Figure 1.4: Thesis structure.

Chapter 2 provides an overview of the whole state-of-the-art that could be divided into three stages: the very early stage based on statistical models; the second stage based on traditional ML models; and the third (also current) stage based on DL models; for traffic predictions including traffic flow, speed and travel time. Besides, it also defines the general traffic prediction problem and the performance evaluation methods, and summarises public data sources used in literature.

Chapter 3 describes our designed Ensemble Model, EM, for solving short-term traffic flow prediction problem on linear roadways. We collected and analysed traffic data from linear roadways and then evaluated our designed EM model on the collected traffic datasets. By the end, we analysed and discussed the result of our EM model by comparing it to the

results of several well-known existing models.

Chapter 4 presents our developed DL model, ALLSCP, for short-term traffic flow prediction on intersections. It details the specific features (i.e. short-, medium-, and long-term temporal as well as global and local spatial features) that our developed model is able to extract from original traffic data for final prediction. Two real-world datasets from different places are then collected and used for evaluating the developed model. In addition, comparison experiments via comparing our developed model to several well-known existing models and ablation experiments by removing one module once to build the variants of ALLSCP are conducted and analysed to show the performance of our developed model.

Chapter 5 describes our novel DL model, SAGCNT-SST, for multi-step traffic speed prediction on large-scale road networks. It explains how SAGCNT-SST function extract complex dynamic-spatial and temporal features for prediction on network-wide. Furthermore, two large datasets from different places are described and used for comparison experiments and ablation experiments to show results.

Chapter 6 presents our novel DL model, VDGCNTGA, by building an inside algorithm to generate a virtual road graph to exploit hidden spatial dependencies under the real road graph, for multi-step traffic speed prediction on large-scale road networks. It details how VDGCNTGA generate a virtual road graph and extract dynamic- and hidden-spatial and dynamic-temporal features for final prediction. Similar to Chapter 5, two large datasets from different places are then described and used for comparison experiments and ablation experiments to show results.

Chapter 7 presents a novel DL model, GCNT-Seq2Seq, based on the Sequence-to-Sequence architecture with an embedded module, for long-term traffic prediction on large-scale road networks. The used dataset for evaluating the GCNT-Seq2Seq model and comparing it to several well-known existing DL models is also described. By the end, the results of our proposed model and its competed existing models are analysed and discussed.

Finally, Chapter 8 concludes this thesis and lists several future research directions related to this research work.

1.5 List of Publications and Award

1.5.1 Journal Papers

1. Ge Zheng, Wei Koong Chai, Vasilis Katos, and Michael Walton; "A Joint Temporal-Spatial Ensemble Model for Short-Term Traffic Prediction", *Neurocomputing*, volume 457, pages 26–39, Elsevier, 2021.

2. Ge Zheng, Wei Koong Chai, and Vasilis Katos; "A Dynamic Spatial-temporal Deep Learning Framework for Traffic Speed Prediction on Large-scale Road Networks", In *Expert Systems with Applications*, pages 116585, Elsevier, 2022.

3. Ge Zheng, Wei Koong Chai, Jiankang Zhang, and Vasilis Katos; "VDGCNTGA: A Novel Network-wide Virtual Dynamic Graph Convolution Neural Network and Transformer-based Traffic Prediction Model", Submitted to *IEEE Transactions on Knowledge and Data Engineering*.

4. Ge Zheng, Wei Koong Chai, Jing-Lin Duanmu and Vasilis Katos; "A Survey and Comparative Study of Hybrid Deep Learning Models for Multi-step Traffic Prediction in Large-scale Road Networks", Submitted to *Information Systems Research*.

1.5.2 Conference Papers

1. Ge Zheng, Wei Koong Chai, and Vasilis Katos; "An Ensemble Model for Short-Term Traffic Prediction in Smart City Transportation System", In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1-6.

2. Ge Zheng, Wei Koong Chai, and Vasilis Katos; "The Sequence-to-Sequence architecture with An Embedded Module for Long-Term Traffic Speed Forecasting with Missing Data", In *IEEE 26th International Conference on Automation and Computing (ICAC)* Accepted in 6th of June, 2021.

1.5.3 Best Conference Paper Award

1. Ge Zheng, Wei Koong Chai, and Vasilis Katos; "The Sequence-to-Sequence architecture with An Embedded Module for Long-Term Traffic Speed Forecasting with Missing Data", In *IEEE 26th International Conference on Automation and Computing (ICAC)* Accepted in 6th of June, 2021.

Chapter 2

Literature Review

2.1 Introduction

Traffic prediction research has evolved rapidly over the years. For our purpose here, it can be divided into three distinct stages: 1) the first stage mainly employs statistical methods, e.g., (Van Der Voort et al. 1996, Ahmed & Cook 1979, Ho & Xie 1998, Williams et al. 1998); 2) the second stage uses traditional ML methods, e.g., (Ghosh et al. 2007, Castro-Neto et al. 2009, Zhang et al. 2013, Lippi et al. 2013); 3) the third stage utilises more advanced DL methods, e.g., (Mikolov et al. 2010, Huang et al. 2014, Jia et al. 2016, Toncharoen & Piantanakulchai 2018, Diehl et al. 2019). The transition from the first to the second stage is due to the breakthrough in ML models with non-linear kernels or activation functions that can efficiently analyse non-linear relations of traffic data in time domain. This compensates for the disadvantage of previous statistical models which largely failed to take into account non-linear relations of traffic data (Zhang 2003). With the emergence of new sensor technologies, traffic data with more features can be collected for improving the prediction accuracy. Following this, DL models that are able to learn features from large high-dimensional datasets have been used for traffic prediction. This promotes the methods of solving traffic prediction problem into the third stage and overcomes the disadvantage of ML models that are shallow and often, insufficient for analysing large high-dimensional traffic data.

This chapter reviews existing works for traffic prediction from all three stages in detail after giving general traffic prediction problem formulation in Section 2.2. The first stage based on statistical methods is described in Section 2.3, the second stage based on ML methods is presented in Section 2.4 and the third stage which is also the current stage

based on DL methods is detailed in the Section 2.5. In addition, public data sources and performance evaluation methods are described in Section 2.6 and Section 2.7, respectively. Finally, the summary will be presented in Section 2.8.

2.2 General Traffic Prediction Problem Formulation

The existing works for solving traffic prediction problems in literature generally considers traffic data in Euclidean space or Non-Euclidean space (Monti et al. 2017). In Euclidean space, traffic data from roadways or road networks is considered to be grid data. On the other hand, in Non-Euclidean space, traffic data is considered as structured graph data and road distances between detectors on the roadways or road networks are usually used to calculate the weights of edges (Li et al. 2018). The section will describe traffic data used in existing works including traffic flow or speed, road network data and external information such as weather and road properties, and then give two common problem formulations. One only uses traffic data and/or external information to solve traffic prediction problem in Euclidean space while the other one uses both traffic data (and/or external information) and road network data to solve the problem in Non-Euclidean space.

2.2.1 Traffic Data

In this thesis, traffic data refers to traffic flow and/or traffic speed. Traffic data from a roadway and/or a road network with N sensors is represented as $\mathrm{x_t} = \{x_t^1, x_t^2, \ldots, x_t^i, \ldots, x_t^{N-1}, x_t^N\}; \mathrm{x_t} \in \mathbb{R}^N, (i = 1, 2, 3, \ldots, N)$, in which x_t^i denotes the traffic data measured at node i at t^{th} time interval. Typically, a time interval can represent 5, 15, 30, 45 and 60 minutes (Bickel et al. 2007). Then $\mathbf{X} = \{\mathrm{x_{t-T+1}}, \mathrm{x_{t-T+2}}, \ldots, \mathrm{x_{t-1}}, \mathrm{x_t}\}; \mathbf{X} \in \mathbb{R}^{T \times N}$, $(T = 1, 2, 3, \ldots)$ gives the traffic data collected from N sensors in the network for the past T time intervals. Conversely, the traffic data for the future is written as $\mathbf{X}' = \{\mathrm{x_{t+1}}, \mathrm{x_{t+2}}, \ldots, \mathrm{x_{t+t'}}, \ldots, \mathrm{x_{t+T'}}\} \in \mathbb{R}^{T' \times N}$ where T' is the prediction horizon. Generally, traffic prediction problems can be categorised into short- ($T' < 30$ minutes) and long-term ($T' \geq 30$ minutes).

2.2.2 Road Network Data

Considering a road network represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes representing sensor locations or road segments with $|\mathcal{V}| = N$. $\mathcal{E}$ is the set of edges representing

physical connectivity between sensor locations or road segments. For general networks like social networks, $\mathcal{G}$ can be represented by $\mathbf{A} \in \mathbb{R}^{N \times N}$, the $N \times N$ symmetric adjacency matrix, with its element $A_{i,j} = 1$ if there exists a link between node i and j and 0 otherwise. However, in road networks used for traffic prediction, $\mathcal{G}$ is represented by $\tilde{\mathbf{A}} = (\mathbf{A} + \mathbf{I_N}) \in \mathbb{R}^{N \times N}$ where $\mathbf{I_N}$ is the $N \times N$ identity matrix because the future traffic states of a node is influenced by its own current state. The degree matrix of graph $\mathcal{G}$, $\mathbf{D} \in \mathbb{R}^{N \times N}$ is then defined as $D_{i,i} = \sum_j A_{i,j}$, which sums the number of edges connected to each node. Some existing works like (Li et al. 2018, Zhang, Li, Lin, Wang & He 2019) use $\tilde{\mathbf{A}}$, that only represents the physical connections between the node and its adjacent nodes, to analyse spatial dependencies. Other existing works like (Cui et al. 2019, Zhao, Song, Zhang, Liu, Wang, Lin, Deng & Li 2019) use $\tilde{\mathbf{A}}^k = (\mathbf{A} + \mathbf{I_N})^k$, that represents the physical connections between the node and its $k - hop$ neighbours, in place of $\tilde{\mathbf{A}}$.

2.2.3 External Information

The traffic system is a complex system that can be influenced by various external factors, such as Points Of Interest (POIs)(such as high street, shopping malls and restructures), incidents, weather, holidays, weekdays and weekends. For example, the distribution of POIs around a road segment can affect traffic states by determining the visiting patterns and the attractiveness of the road segment (Zhu et al. 2021). Therefore, for enhancing the performances of models, those external factors are considered in some existing works such as (Jia & Yan 2020, Sun, Yang, Han, Ma & Zhang 2020, Wang et al. 2021, Zhu et al. 2021). Here, we use $\mathbf{E}$ to contain those external information for formulating the traffic prediction problem in Section 2.2.4.

2.2.4 Problem Formulation

Based on the traffic data, the graph information of the road network and the external information explained above, the traffic prediction problem defined in Euclidean space and Non-Euclidean space in literature can be formulated as Eq. (2.1) and Eq. (2.2), respectively.

$$\mathbf{X}' = F\Big(\mathbf{X}\Big) \; or \; \mathbf{X}' = F\Big(\mathbf{X}; \mathbf{E}\Big) \tag{2.1}$$

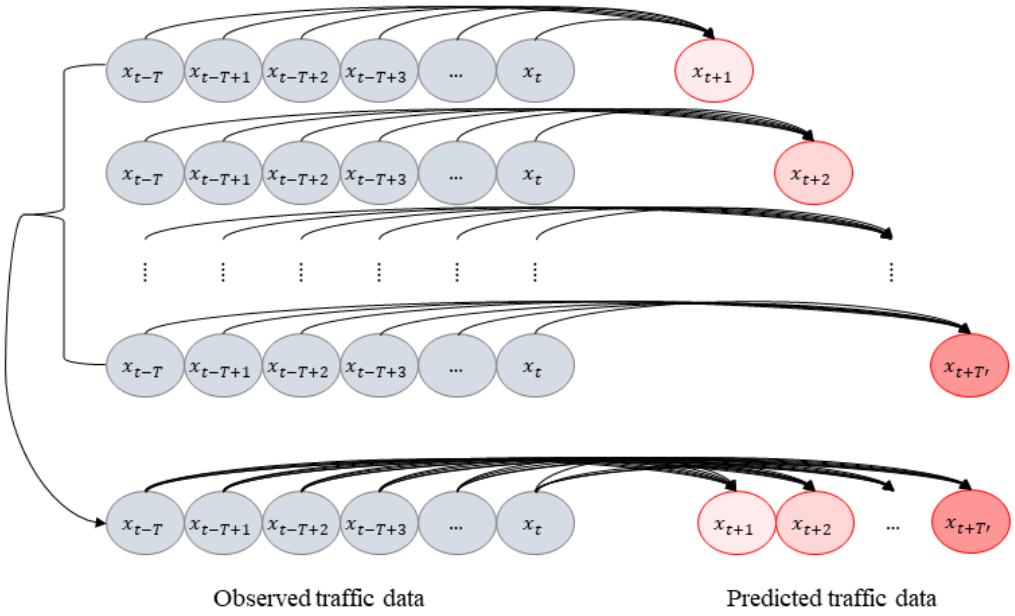
$$\mathbf{X}' = F\Big(\mathbf{X}; \mathcal{G}\Big(\mathcal{V}, \mathcal{E}, \tilde{\mathbf{A}}(or \tilde{\mathbf{A}}^k)\Big)\Big) \; or \; \mathbf{X}' = F\Big(\mathbf{X}; \mathcal{G}\Big(\mathcal{V}, \mathcal{E}, \tilde{\mathbf{A}}(or \tilde{\mathbf{A}}^k)\Big); \mathbf{E}\Big) \tag{2.2}$$

where the objective is to learn the mapping function $F(.)$ and compute the traffic data in the next T' time intervals given the traffic data in the past T time intervals and/or external information $\mathbf{E}$ and/or road network information $\mathcal{G}$.

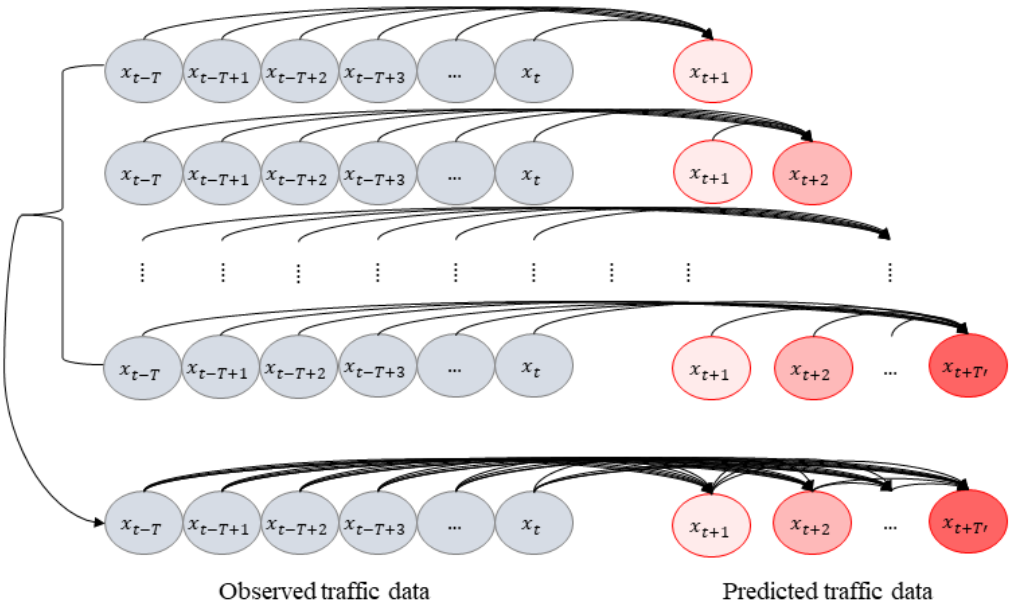
In addition, we also see two main approaches for addressing the traffic prediction problem above: the direct prediction approach (cf. Figure 2.1 (a)) and the auto-regressive approach (cf. Figure 2.1 (b)). The direct prediction approach, for instance adopted by (Zheng et al. 2020), uses observed traffic data to directly predict traffic states in the multiple future time steps at the same time. The farther away from the observations the predicted traffic data is, the larger the error tend to be. The auto-regressive approach, on the other hand, adopted in (Zhang, Li, Lin, Wang & He 2019, Cui et al. 2019, Li et al. 2018), predicts traffic data based on the short-temporal dependency. The error-prone predictions are included as the inputs along with last prediction to make further predictions, which results in error accumulations.

2.3 Statistical Models

Statistical models are based on statistical analysis of data and are very efficient tools to find and analyse patterns from data. Based on those advantages, it has been introduced for solving traffic prediction problems at the early stage. One of the earliest statistic model used for short-term traffic prediction is the Auto-Regressive Moving Average (ARMA) (Box et al. 1970). It considers the problem as a pure temporal process. Based on ARMA model, (Ahmed & Cook 1979) further built the Auto-Regressive Integrated Moving-Average (ARIMA) model for analysing freeway traffic time series data. 166 datasets from three places including Los Angeles, Minneapolis and Detroit are collected to optimise parameters of ARIMA model for short-term traffic prediction. Both ARMA and ARIMA include a general Auto-Regressive module (AR) making predictions using previous traffic data and a general Moving Average (MA) module making predictions using the mean and previous errors. The difference is that ARMA is commonly used for stationary data while ARIMA is not only utilised for stationary data but also for non-stationary data by differentiating to remove the trend in a time series and then making it stationary.



(a)



(b)

Figure 2.1: (a) Traffic prediction with long-temporal dependencies. (b) Traffic prediction with the error accumulation. (The colour indicates the prediction error, the darker the larger.)

ARIMA has three important parameters: (1) p – the number of auto-regressive terms, (2) d – the number of non-seasonal differences for converting data to be stationary and (3) q – the number of lagged forecast errors in the prediction equation. For the short-term traffic prediction, the input data is first processed to fulfil the stationary property by differentiating the input d times. Then short-term temporal features are extracted from

previous p time intervals with the number of lagged prediction errors q by using Eq. (2.3).

$$x^i_{t+t'} = c + \phi_1 x^i_t + \phi_2 x^i_{t-1} + \cdots + \phi_{p-1} x^i_{t-(T-2)} + \phi_p x^i_{t-(T-1)} \tag{2.3}$$

where ϕ_p is the parameter of the auto-regressive part of ARIMA, and c is a constant.

Motivated by the accuracy of traffic prediction achieved by ARMA and ARIMA, some variants were further developed. For example, Van Der Voort *et. al.* (Van Der Voort et al. 1996) combined a Kohonen map with ARIMA model (named KARIMA) for predicting short-term traffic flow, in which the Kohonen map is used as an initial classifier and each class has an individually tuned ARIMA model associated with it. William *et. al.* (Williams & Hoel 2003) argued that seasonal patterns could be exploited to improve prediction accuracy, and proposed the Seasonal ARIMA (SARIMA) model for traffic flow prediction.

In addition, ARIMA has also been combined with traditional ML algorithms and recent popular DL algorithms for particular pattern analysis of traffic data. Li *et. al.* (Li et al. 2016) integrated ARIMA and Support Vector Regression (SVR) to build hybrid models (named AS and SA) for short-term highway traffic flow prediction, in which ARIMA is used to analyse short-term periodical traffic patterns. The results are compared to the results from single ARIMA and SVR, and indicate that ARIMA is capable of improving traffic prediction accuracy when combined with SVR (Li et al. 2016). Mehdi *et. al.* (Mehdi et al. 2019) combined ARIMA and the fuzzy regression modules to build a Fuzzy Autoregressive Integrated Moving Average (FARIMA) model for traffic prediction on cloud computing, in which ARIMA is also used to analyse short-term traffic patterns. Another work that combines ARIMA and Neural Network (NN) modules to build a NN-ARIMA model was developed in (Ma, Antoniou & Toledo 2020), in which ARIMA is utilised to extract local-periodical patterns of traffic data. Furthermore, our work (Zheng et al. 2021) proposed a joint temporal-spatial ensemble model, named ALLSCP, by integrating ARIMA, Long-Short Term Memory (LSTM), Stacked AutoEncoder (SAE) and Capsule Neural Network (CAPSNET) for short-term traffic flow prediction. The function of ARIMA in ALLSCP is similar to its in (Li et al. 2016, Mehdi et al. 2019, Ma, Antoniou & Toledo 2020) for analysing short-term traffic patterns.

2.4 Machine Learning Models

As discussed in Section 2.1, statistical models, including ARIMA and its various variants, are capable of analysing short-term traffic patterns but could not capture long-term and non-linear relationships in traffic data. Even in hybrid models that are built by combining several modules based on their capabilities, statistical models are only considered as a module to analyse one of traffic patterns like the short-term traffic pattern. To account for the non-linearity in traffic data, ML models, that have been widely used for solving non-linear prediction problems in various different fields such as in the chemical industry (Liu et al. 2015, 2018), biology (Costello & Martin 2018), and modern manufacturing systems (Chen et al. 2017), were then advocated. In this section, we will present ML models for solving traffic prediction problem.

According to (Shalev-Shwartz & Ben-David 2014, Sun, Aljeri & Boukerche 2020), ML models used for solving traffic prediction problems can be categorised into two groups: 1) **regression models** that learn the relationship of the dependent variables and the independent variables and then use a curve or a line to fit them and 2) **classification models** that calculate the similarities of the observed data and its targeted data and then use the learned similarities to make multiple classifications as predictions. The following content will explain those two groups, respectively.

2.4.1 Regression Models

The type of well-known regression models applied for traffic prediction is linear regression where predictions are achieved by considering it as a linear mapping between the observed traffic data and its targeted traffic data. For example, (Hobeika & Kim 1994) developed a linear regression model to predict traffic flow on a given road segment by considering the strong effect of the upstream traffic states on the current location traffic state. Authors in (Hobeika & Kim 1994) combined three prediction modules and each of them was based on different traffic variables including upstream traffic flow, current traffic flow and the average of historical traffic flow. These three modules were evaluated and selected by a linear regression model for traffic prediction on three different prediction horizons. Another work that uses a linear regression model to predict travel time based on the linear relationship between future travel time and previous travel time was developed in (Rice & Van Zwet 2004).

Linear regression models mentioned above (Hobeika & Kim 1994, Rice & Van Zwet

2004) could not capture non-linear patterns in traffic flow. Therefore, researchers combined linear regression models with other methods for traffic prediction in the later literature. For example, (Fei et al. 2011) joined a Bayesian inference framework into a linear regression model to build a Bayesian inference-based Dynamic Linear Model (DLM) for online short-term travel time prediction on a freeway stretch. DLM is one part of the proposed method in (Fei et al. 2011) that considers the predicted freeway travel time as the sum of the median of historical travel times, time-varying random variations in travel time, and a model evolution error. The median is used to represent travel time patterns while the variation represents unexpected fluctuations. Bayesian inference is used to revise the state of a priori knowledge of travel time based on new available information. Furthermore, (Okawa et al. 2017) built a traffic prediction model based on Convolved Bi-Linear Poisson Regression on multiple inter-city roads, which aims to reduce the number of model parameters. In addition, the stochastic variational Bayes method (Paisley et al. 2014) is utilised to solve the optimisation problem, which allows model parameters to be updated online.

In summary, regression models have several disadvantages: 1) difficult to deal with long-term traffic prediction problems; 2) hard to find a set of model parameters to represent traffic patterns hidden in real traffic data; 3) learned parameters only suitable for a specific road, not general for other roads; 4) only suitable for solving simple road traffic prediction problem, not for roads with complex structures and richer traffic patterns. On the contrary, for traffic prediction on simple roads or just road stations, regression models would be efficient and applicable.

2.4.2 Classification Models

The two common classification models used for traffic prediction are K-Nearest Neighbours (KNN) and Support Vector Machine (SVM). KNN can capture the spatial relationship between road segments by calculating the similarities of historical traffic states in a sensor with its neighbours. On the other hand, SVM avoids the disadvantages of solutions easily fall into local optimum compared to other nonlinear prediction models. A variant of SVM used for solving regression problem is named as SVR. The idea behind SVR is that the traffic prediction problem is transformed into a linear regression problem in high dimensional space by its kernel function.

(Zhang et al. 2013) proposed a KNN model for short-term traffic flow prediction on urban expressways, in which the original data is pre-processed and standardised to avoid the

magnitude difference of the sample data and improve the prediction accuracy. Habtemichael *et. al.* (Habtemichael & Cetin 2016) proposed an enhanced KNN model for short-term traffic prediction by identifying similar traffic patterns using the weighted Euclidean distance. Another improved KNN model was developed in (Cai et al. 2016) to enhance prediction accuracy based on spatial-temporal correlation. The authors used equivalent distances to replace the physical distances among road segments and then utilised a spatial-temporal state matrix to describe the traffic state of a road segment. Finally, Gaussian weighted euclidean distance is used to find the similarities of historical traffic states and the current traffic state for prediction. To solve the negative effects of the uncontrollable and unpredictable elements of special events on prediction, (Yu et al. 2019) built a Special Event-based KNN (SEKNN) model, in which both euclidean distance and cosine distance function are used to calculate the similarities of the defined benchmark state and trend vectors.

An early work that introduced SVR into the traffic prediction topic was conducted in (Ding et al. 2002) for traffic flow prediction on a given crossroad. SVR is commonly considered as a kernel-based ML model. Therefore, different SVRs have been developed by defining different kernels for traffic prediction. For example, both (Castro-Neto et al. 2009) and (Hong et al. 2011) use Radial Basis Function (RBF) as the kernel function of SVR for traffic flow prediction. The difference is that SVR in (Castro-Neto et al. 2009) is working online and could predict short-term traffic flow under both typical and atypical situations and SVR in (Hong et al. 2011) uses the ant colony optimisation algorithm to determine the values of its parameters. Furthermore, (Lippi et al. 2013) introduced two seasonal kernel functions including seasonal RBF function and seasonal linear function to SVR to facilitate SVR to make use of the seasonal information in the traffic records.

In summary, classification models could analyse nonlinear relationships of traffic data compared to statistical models. However, they are shallow to analyse complex traffic patterns hidden under road networks, especially for large-scale road networks with hundreds of sensor stations or road segments.

2.5 Deep Learning Models

DL models used for traffic prediction can be categorised into two groups: individual and hybrid DL models. The individual DL model only uses one type of DL model while the hybrid DL model combines several types of DL models to build the hybrid model (Li et al. 2021). We will review DL models based on those two groups in the next section.

2.5.1 Individual Deep Learning Models

Lv *et. al.* (Lv et al. 2015) built a SAE model to learn generic traffic flow features and then trained it in a greedy layer-wise fashion. SAE is a deep learning network built based on multiple autoencoders as building blocks where each block has one input layer, one hidden layer and one output layer. Another work, that is also built based on deep autoencoders with symmetrical layers for the encoder and the decoder to learn temporal correlations of a transportation network, was proposed in (Zhang, Yao, Hu, Zhao, Li & Hu 2019).

Following the success of Convolution Neural Network (CNN) in the area of image recognition, it has been introduced to the traffic prediction area where traffic data is first transformed into images and then learned by CNN. For example, (Jiang & Zhang 2018) converted GPS data into images and built a DL model based on CNN to learn features from the converted images for prediction. Another work that learns traffic as images was (Zhang et al. 2017), in which the proposed model, ST-ResNet, employs convolution-based residual networks to model nearby and distant spatial dependencies between any two regions in a city. Considering three patterns of crowd traffic flow including temporal closeness, period and trend, ST-ResNet is capable of analysing those patterns and then fuses them for final prediction. Furthermore, (Kim et al. 2018) proposed a neural network with capsules that could replace max pooling in CNN model to avoid losing important information by locally taking the highest activation values, for traffic flow prediction.

Considering traffic data having definite temporal patterns, (Vinayakumar et al. 2017, Khan et al. 2019, Ramakrishnan & Soni 2018) evaluated various recurrent neural networks including simple Recurrent Neural Network (RNN), LSTM, Gated Recurrent Unit (GRU) and Identity RNN (IRNN) for traffic prediction. In those works, authors consider modelling traffic data as pure time series data and predict the future time series based on previous time series with long-term dependencies.

In summary, individual DL models mentioned above could predict traffic states in the future but could not fully exploit features that are helpful for improving prediction accuracy. For example, features learned from traffic images by CNN only represent local relations of traffic data due to the limitation of the kernel size in CNN layers. Besides, features learned from various RNN only carry temporal relations of traffic data. However, traffic data has not only spatial dependencies but also temporal dependencies, especially on large-scale road networks.

2.5.2 Hybrid Deep Learning Models

Single DL models presented in Section 2.5.1, such as RNN and its variants (LSTM and GRU) (Fu et al. 2016), and CNN (Toncharoen & Piantanakulchai 2018), are unable to fully analyse spatial-temporal dependencies of traffic data hidden in large-scale road networks. Following this, in recent literature, hybrid DL models are constructed by combining several individual DL models to improve prediction accuracy. Furthermore, (Yin et al. 2021*a*, Shi et al. 2019, Seo et al. 2017, Zhu et al. 2018) summarised that existing hybrid DL models commonly consist of two types of modules for analysing spatial and temporal dependencies, respectively. Through reviewing the recent literature, hybrid DL models that satisfy this condition commonly contain CNN or Graph Convolution Network (GCN) for spatial dependency analysis and LSTM or GRU for temporal dependency analysis.

We present a summary of hybrid DL models in Table 2.1 according to their main constituent models and further segregated chronologically by year of publication. From this, we can classify them into three based on their choice of model for analysing spatial dependencies: 1) CNN-based models, 2) GCN-based models, and 3) transformer-based models. We also show the prediction task (either predicting traffic flow, speed and/or occupancy), prediction horizon considered and the dataset(s) used in those works.

Table 2.1: Hybrid DL models for traffic prediction based on the analysis of spatial-temporal dependencies in the recent literature.

Reference	Methodology	Task	Prediction Horizon	Dataset	Year
(Wu & Tan 2016)	CNN+LSTM	flow	short-term	PeMS data	2016
(Liu et al. 2017)	CNN+LSTM+Bi-LSTM	flow	short-term	PeMS data	2017
(Yu et al. 2017)	CNN+LSTM	speed	short- & long-term	GPS data	2017
(Yang et al. 2018)	CNN+LSTM	speed	short-term	GPS data	2018
(Wu et al. 2018)	Attention+CNN+GRU	flow	long-term	PeMS data	2018
(Duan et al. 2018)	CNN+LSTM	flow	long-term	Taxis GPS data	2018
(Jin et al. 2018)	CNN+LSTM+ANN	flow	short- & long-term	Mobile & Taxi data	2018
(Zang et al. 2018)	CNN+LSTM	speed	long-term	Highways' data	2018
(Zhao, Lin & Xu 2019)	CNN+LSTM	occupancy & flow	short- & long-term	Bike sharing & parking datasets	2019
(Zhao, Qu, Zhao & Jiang 2019)	CNN+Self-attention-LSTM	flow	short-term	Abilene & GEANT datasets	2019
(Zheng et al. 2019)	CNN+LSTM	flow	long-term	Tian Chi platform data	2019
(Essien et al. 2019)	CNN+LSTM	flow	long-term	Greater Manchester data	2019
(He et al. 2019)	CNN+LSTM	flow	long-term	TaxiBj & BikeNYU	2019
(Yao et al. 2019)	LocalCNN+LSTM+Attention	flow	long-term	Taxi-NYU & Bike-NYU	2019
(Le Nguyen et al. 2019)	CNN+LSTM	flow	long-term	Abilene dataset	2019
(Niu et al. 2019)	CNN+LSTM	speed	long-term	Taxi GPS data	2019

(Ma, Zhong, Li, Ma, Cui & Wang 2020)	CapsNet+LSTM	speed	short-term	GPS data	2020
(Li et al. 2018)	GCN+Seq2Seq(GRU)	speed	short- & long-term	Pems-bay & Metr-la	2017
(Liao et al. 2018)	GCN+Seq2Seq	speed	short- & long-term	Baidu map traffic data	2018
(Cui et al. 2019)	GCN+LSTM	speed	short-term	Greater Seattle traffic data and INRIX GPS data	2019
(Xie et al. 2019)	GCN+Seq2Seq(RNN)	speed	short- & long-term	GPS data	2019
(Kim et al. 2019)	GCN+RNN	speed	short-term	Santander traffic data	2019
(Do et al. 2019)	Attention-GCN +GRU	flow	short- & long-term	Melbourne data	2019
(Zhang, Li, Lin, Wang & He 2019)	GCN+Seq2Seq(GRU) with Attention mechanism in the decoder	speed	short- & long-term	Beijing traffic data	2019
(Li, Xiong, Chen, Lv, Hu, Zhu & Wang 2019)	GCN+LSTM+Soft-attention	flow	short-term	PeMS data	2019
(Pan et al. 2019)	Attention-GCN+GRU (the Seq2Seq framework)	flow & speed	short- & long-term	TDrive & Metr-la	2019
(Chen et al. 2019)	GCN+GRU	speed	short- & long-term	Metr-la & Pems-Bay	2019
(Zhang, Guan, Cao, Wang & Wu 2019)	ARMA+GCN+LSTM	speed	short-term	GPS data	2019
(Zhao, Song, Zhang, Liu, Wang, Lin, Deng & Li 2019)	GCN+GRU	speed	short- & long-term	SZ-taxi & Los-Loop	2019

(Fang et al. 2022)	Attention-GCN+LSTM	average queue length & speed	short- & long-term	Taxi GPS data	2021
(Yin et al. 2021*b*)	Attention-GCN+Attention-LSTM	flow	short- & long-term	PeMSD4 & PeMSD8	2021
(Zheng et al. 2022)	Self-attention+GCN+GRU (the Seq2Seq framework)	speed	short- & long-term	Loop-Seattle & Metr-la	2022
(Zheng et al. 2020)	spatial- & temporal-transformer	speed & flow	short- & long-term	Pems-bay & Xiamen datasets	2020
(Xu et al. 2020)	spatial- & temporal-transformer	flow	short- & long-term	PeMSD7(M) & Pems-Bay	2020

We also create a taxonomy of these models which is shown in Figure 2.2. In the following, we will detail our taxonomy.

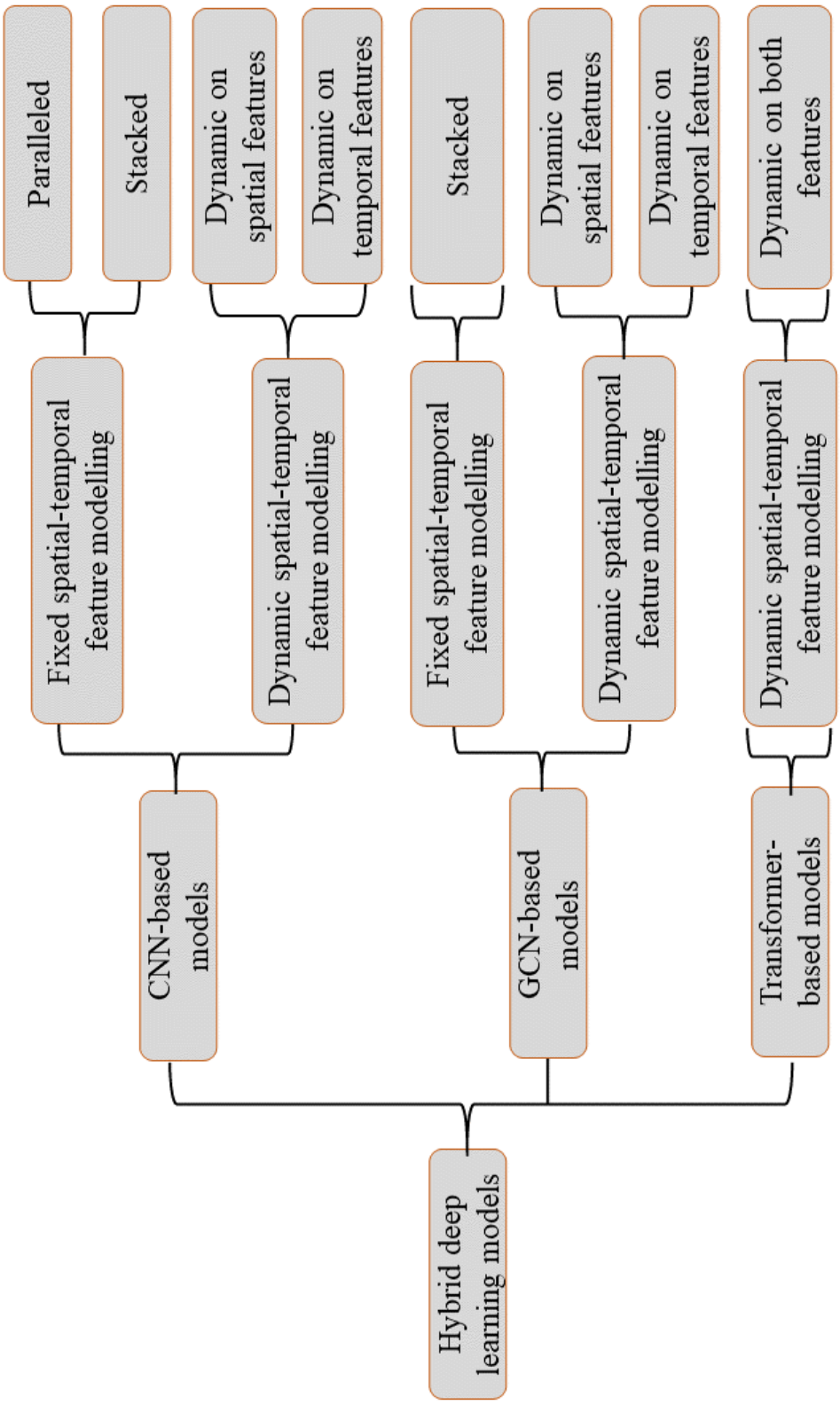


Figure 2.2: The classification of hybrid DL models for traffic prediction reviewed in this thesis.

Convolution Neural Network-based Models

CNN has been applied in different fields including sentence classification (Chen 2015), image classification (Krizhevsky et al. 2012), video classification (Karpathy et al. 2014) and human action recognition (Ji et al. 2012). It extracts local spatial features via its convolutional kernels. CNN was introduced into solving traffic prediction problems due to its ability of processing spatial correlations (Lawrence et al. 1997). For example, (Ma et al. 2017) exploited CNN for traffic speed prediction on large-scale road networks by considering traffic data as images. However, due to CNN only focusing on spatial feature extraction, (Ma et al. 2017) neglected the important contributions of temporal features towards the final prediction. As such, LSTM and GRU are often combined with CNN to form hybrid DL models to supplement the disadvantages of CNN for temporal feature extraction (cf. Table 2.1).

From Figure 2.2, CNN-based models could be grouped into fixed CNN-based models and dynamic CNN-based models based on the types of spatial-temporal feature analysis.

Fixed CNN-based models: Fixed CNN-based models consider the spatial-temporal dependencies of traffic data being fixed via sharing parameters, which means the different neighbouring sensors or road segments affect traffic state on the targeted sensor or road segment in the same level. From the perspective of framework design, fixed CNN-based models can also be categorised into two: paralleled and stacked.

1. Paralleled – CNN and LSTM (or GRU) are respectively used to process spatial and temporal features and then their outputs are concentrated and sent to a fully-connected layer for final prediction

2. Stacked – CNN is used to extract spatial features from original spatially-fused traffic time series first and then its output is sent to LSTM (or GRU) as the input for the temporal feature extraction and the final prediction

Adopting the paralleled framework, Wu *et al.* (Wu & Tan 2016) used a 1D CNN and two LSTMs to build a hybrid DL model, named CLTFP. 1D CNN is used to exploit inner spatial dependencies of the road network. One of LSTMs is used to capture short-term temporal features from previous hours and the other is utilised to extract periodic features from past days and weeks. The spatial, short-term temporal and periodic features are fused into a feature vector and then sent to a regression layer to perform predicting. The other two similar models (named U-Net and TreNet) were developed by Niu *et al.* (Niu

et al. 2019) and Zhao *et al.* (Zhao, Lin & Xu 2019) respectively. Compared to (Wu & Tan 2016, Niu et al. 2019, Zhao, Lin & Xu 2019) that directly combine CNN and LSTM in the parallel way, Liu *et al.* (Liu et al. 2017) firstly combined CNN and LSTM to generate a Conv-LSTM module in the stacked way and then combined the Conv-LSTM and Bi-LSTM in the parallel way for traffic flow prediction. Conv-LSTM is used for spatial-temporal feature extraction and Bi-LSTM is utilised for periodic feature extraction. A fully-connected layer as a predictor is used to generate final prediction by feeding the concentration of the extracted spatial-temporal and periodic features. Considering the influence of exterior factors on traffic prediction, such as weather condition and the physical characteristics of roads, Zheng *et al.* (Zheng et al. 2019) developed a model named DELA for traffic flow prediction. This model not only includes an integrated model composed of CNN and LSTM but also contains an embedding component for learning exterior factors such as route structure information, weather conditions and date information.

For the stacked, two models (named SRCNs and MSTFLN) were developed by Yu *et al.* (Yu et al. 2017) and Zhang *et al.* (Zang et al. 2018), respectively. Both models learn traffic data as a series of static images by combining CNN and LSTM. SRCNs uses 2D CNN instead of 1D CNN due to considering the drawback of 1D CNN failing to analyse interactions of roads in the network. MSTFLN utilises an extra CNN for final prediction apart from the combined modules used for the spatial-temporal feature extraction. To analyse traffic state in details, Jin *et al.* (Jin et al. 2018) proposed a DL-based approach named STRCNs to predict both inflow and outflow of crowds in each region of a city. STRCNs consists of four modules: a basic neural network and three recurrent convolutional network modules composed of CNN and LSTM. The basic neural network captures external features and the three recurrent convolutional network modules focus on learning both spatial and temporal dependencies in crowd flows corresponding to closeness, daily influence and weekly influence, respectively. The four outputs from these modules are merged together with different learnable weights before passing the $tanh$ layer for final prediction. Compared to (Yu et al. 2017, Zang et al. 2018, Jin et al. 2018) that predict traffic data without considering the types of roads, Yang *et al.* firstly used a Spatial-Temporal Correlation Algorithm (STCA) to identify and extract the critical road sections and then developed a hybrid DL model (named CRS-ConvLSTM NN) based on CNN and LSTM for traffic speed prediction on these critical road sections. To accelerate the speed of training process, Duan *et al.* (Duan et al. 2018) used a greedy reinforcement policy to train a deep hybrid model (named CNN-LSTM) based on CNN and LSTM. Essien *et al.* (Essien et al. 2019) developed a stacked autoencoder, including an encoder based on CNNs and a decoder based on Bi-LSTMs, for traffic flow and speed prediction. Another model (named STCNN) based on the encoder-decoder architecture was proposed in

(He et al. 2019) for predicting long-term traffic flow. The encoder consists of a ConvLSTM to learn the spatial-temporal traffic dependencies and a Skip-ConvLSTM to learn the periodic traffic patterns. The decoder consists of another ConvLSTM for decoding the spatial-temporal dependencies from the output of the encoder.

Dynamic CNN-based models:

Fixed CNN-based models consider that different neighbouring sensors or road segments affect traffic state on the targeted sensor or road segment in the same level. The fact is that different neighbouring sensors or road segments bring different effects for traffic state on the targeted sensor or road segment. Similarly, traffic state in different previous time intervals also bring different effects for traffic state in the future. Due to these reasons, the attention mechanism, that represents a major breakthrough in the field of natural language processing, has been introduced into defining different weights in the space and/or time dimensions for analysing dynamic spatial-temporal dependencies. By dynamic CNN-based models, we refer to models that has the ability to analyse dynamic spatial-temporal dependencies.

For obtaining dynamic dependency in the space dimension, Wu *et al.* (Wu et al. 2018) proposed a Deep Neural Network-Based Traffic Flow prediction model named DNN-BTF, which uses the attention mechanism to learn from the traffic speed and flow and generate dynamic weights first. Then, these weights are joined into traffic flow for generating a near-term traffic flow matrix that is sent to the CNN and GRU modules for spatial and temporal feature extractions, respectively.

For achieving dynamic dependency in the time dimension, Zhao *et al.* (Zhao, Qu, Zhao & Jiang 2019) proposed an end-to-end DL approach (named WSTNet) based on wavelet multi-scale analysis, which adopts the attention mechanism into LSTM for obtaining different weights relating to positions of the input sequences across the entire time intervals. Compared to (Wu et al. 2018, Zhao, Qu, Zhao & Jiang 2019), WSTNet uses discrete wavelet decomposition to decompose original traffic data into multilevel time-frequency traffic sub-series at different time scales before sending to an integrated model based on CNN and LSTM with attention mechanism.

For obtaining dynamic dependencies in both space and time dimensions, Yao *et al.* (Yao et al. 2019) proposed a DL model, Spatial-Temporal Dynamic Networks (STDN), which defines and addresses dynamic-spatial dependencies and dynamic-temporal dependencies by a flow-gated local CNN and LSTM with a self-attention mechanism, respectively.

Graph Convolution Neural Network-based Models

The CNN-based models in Section 2.5.2 consider road networks as regular grids and traffic data with regular Euclidean structure. However, road networks are inherently irregular and traffic data should be treated as non-Euclidean data (Ahmed et al. 2018). Therefore, GCN with the capability of dealing with non-Euclidean structured data has been introduced into the task of traffic prediction. Similar to CNN-based models, GCN-based models can be also classified into two: fixed and dynamic, based on considering spatial-temporal dependencies as fixed or dynamic.

Fixed GCN-based models: Commonly, stacked architecture is adopted for GCN-based models that aim to achieve fixed spatial-temporal features towards final prediction. For example, Zhao *et al.* (Zhao, Song, Zhang, Liu, Wang, Lin, Deng & Li 2019) proposed a Temporal Graph Convolutional Network (named T-GCN) model. It combines GCN with GRU for traffic speed prediction. GCN is used to learn complex topological structures from the $k - hop$ neighbourhood matrix for capturing spatial dependencies and GRU is utilised to learn changes of traffic data along the time dimension for capturing temporal dependencies. To enrich traffic information, Cui *et al.* (Cui et al. 2019) proposed a Traffic Graph Convolutional Long Short-Term Memory Neural Network (named TGC-LSTM), based on GCN and LSTM, to learn the interactions of roads in a large-scale road network from all $k - hop$ neighbourhood matrices. An L1-norm on graph convolution weights and an L2-norm on graph convolution features are added to the loss function for enhancing the interpretability of the model.

In addition, based on the breakthrough of the sequence-to-sequence (Seq2Seq) architecture that is capable of dealing with the long-term sequence problems, Seq2Seq has been integrated into hybrid DL models for analysing the long-term dependency. For example, Li *et al.* (Li et al. 2018) developed a DL model based on Diffusion Convolutional Recurrent Neural Network (DCRNN) under the Seq2Seq architecture for traffic speed prediction, and achieved better performance on two large real-world datasets, when compared to recurrent neural network with Fully Connected LSTM hidden units (FC-LSTM) (Sutskever et al. 2014). Both the encoder and the decoder inside the Seq2Seq architecture consist of GRUs embedded by diffusion convolutional process. The other two hybrid models under the Seq2Seq architecture for traffic speed prediction were developed in (Liao et al. 2018) and (Xie et al. 2019). The hybrid model in (Liao et al. 2018) incorporates the offline geographical and social attributes, spatial dependencies and online crowd queries with a deep fusion. The model in (Xie et al. 2019) adds more temporal attributes including date information on public holidays, working days, peak hours and off-peak

hours for contributing to final prediction in the decoder of the Seq2Seq architecture. Pan *et al.* (Pan et al. 2019) developed a deep-meta-learning based model named ST-MetaNet under the Seq2Seq architecture to predict traffic in a road network. The encoder and the decoder in ST-MetaNet have the same network structure that consists of four components: 1) basic RNN for learning long range temporal dependencies, 2) Meta-knowledge learner for learning the meta-knowledge of nodes and edges from node attributes and edge attributes, respectively, 3) Meta-GAT for capturing diverse spatial correlations by individually broadcasting locations' hidden states along edges, and 4) Meta-RNN for capturing diverse temporal correlations associated with the geographical information of locations.

Dynamic GCN-based models:

Fixed GCN-based models address the traffic prediction problem as a fixed spatial-temporal process. Similar to dynamic CNN-based models, dynamic GCN-based models treat the traffic prediction problem as a dynamic spatial-temporal process via introduction of the attention mechanism for considering the fact that different neighbouring sensors or road segments and different previous time intervals individually affect the targeted sensor or road segment and the future time intervals differently.

For applying the attention mechanism on the space dimension to extract dynamic spatial dependency, our work in (Zheng et al. 2022) developed a model named SAGCN-SST, which joins the attention mechanism into GCN layers for analysing dynamic spatial dependencies and uses the Seq2Seq architecture for dealing with the long-temporal dependency, for traffic speed prediction in large-scale road networks.

Using attention mechanism on the time dimension, Li *et al.* (Li, Xiong, Chen, Lv, Hu, Zhu & Wang 2019) proposed a graph and attention-based long short-term memory network (named GLA), to capture the spatial-temporal features of traffic flow data. GLA uses GCN to mine the spatial relationships of traffic data, and then the output of GCN is fed to LSTM with the soft attention mechanism for the dynamic temporal feature extraction. Another model (named AGC-Seq2Seq-Att) which also joins the attention mechanism into LSTM for dynamic temporal feature analysis, was developed in (Zhang, Li, Lin, Wang & He 2019).

For both spatial and temporal dimensions, Do *et al.* (Do et al. 2019) proposed traffic flow predictor with spatial and temporal attentions (named STANN) under the Seq2Seq architecture. The original traffic data and the adjacency matrix in previous time intervals are processed by the attention mechanism to generate the spatial attention matrix before sending to the Seq2Seq architecture consisting of the convolutional GRU encoder and decoder for the spatial-temporal feature extraction.

Transformer-based Models

Inspired by the newly proposed transformer in (Vaswani et al. 2017) for efficiently modelling long-range dependencies in natural language processing, researchers have introduced the transformer architecture to replace CNN (or GCN) for spatial correlation analysis and LSTM (or GRU) for temporal dependency analysis. Self-attention-based transformer can directly learn dynamic spatial and temporal dependencies by distributing different weights on neighbours in space dimension and on previous time intervals in time dimension for contributing to the targeted sensor or road segment. Besides, transformer can exploit more hidden information in traffic data by defining multi-heads and accelerate training phase by paralleling process. Xu *et al.* (Xu et al. 2020) developed a Spatial-Temporal Transformer Networks (STTNs) to improve the accuracy of long-term traffic prediction, which consists of spatial transformer for modelling dynamic spatial dependencies with self-attention mechanism and temporal transformer for modelling dynamic long-range temporal dependencies across previous time intervals. Another transformer-based model (named GMAN) was developed under the Seq2Seq architecture in (Zheng et al. 2020). Both encoder and decoder under the Seq2Seq architecture consist of multiple spatial-temporal attention blocks to model the impact of the spatial-temporal factors on traffic state. In addition, it also includes a spatial-temporal embedding to encode vertices into vectors using the *node2vec* approach in (Grover & Leskovec 2016) for the vertex representation learning.

2.5.3 Module Summary

Hybrid DL models reviewed in Section 2.5.2 have strong ability to exploit more features in both space and time domains compared to single models. In this section, we will summarise common modules used in these hybrid DL models and categorise them into three groups based on their ability to extract features in space and/or time domain.

Modelling Features in Space Domain

Convolution Neural Network (CNN): Convolutional Neural Network (CNN) is a type of neural networks inspired by biological processes where the neuron connectivity pattern resembles the organisation of animal visual cortex (LeCun et al. 1995). It was initially used for image recognition where each neuron extracts features only in a restricted region of the image by the filters that are able to find relationships between neighbouring inputs.

In terms of addressing traffic prediction problem, most existing works firstly integrate historical traffic data into the shape of grid data based on the locations of sensors or road segments, and then learn it like image data. The spatial features can be extracted by implementing convolutional operation on the restricted region of the traffic image data by Eq. (2.4).

$$\mathbf{CN} = pool(ReLU(\mathbf{W^j_{cn}X} + b^j_{cn})); j = 1, ..., C \tag{2.4}$$

where $\mathbf{W^j_{cn}} \in \mathbb{R}^{c1 \times c2}$ is the parameter matrix of the j^{th} filter with the kernel size $c1 \times c2$ and C is the number of filters. b^j_{cn} is the bias of the j^{th} filter. $ReLU$ is the activation function, and $pool$ is pooling layer. CN is the output of this CNN layer.

Graph Convolution Neural Network (GCN): Graph convolutional neural networks (GCNs) were developed to analyse and learn non-Euclidean data in space domain, and have been shown to achieved good performance for classification in citation networks (Kipf & Welling 2017), syntax-aware neural machine translation (Bastings et al. 2017), 3D human pose regression (Zhao, Peng, Tian, Kapadia & Metaxas 2019) and traffic prediction (Yu et al. 2018). GCNs used for solving traffic prediction problem include spectral GCN, diffusion GCN, and traffic GCN. Briefly,

- **Spectral GCN** was developed in (Estrach et al. 2014), which implements convolution operation on graph data from the spectral domain by computing the eigen-decomposition of the graph Laplacian matrix $\mathbf{L} = (\mathbf{D} - \mathbf{A}) \in \mathbb{R}^{N \times N}$ (Note that some works use the normalised graph Laplacian matrix $\mathbf{L} = (\mathbf{I_N} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}) \in \mathbb{R}^{N \times N}$). Based on this, the spectral GCN can be defined as Eq. (2.5):

$$\mathbf{SGC} = (\mathbf{U}g_\theta\mathbf{U^T})\mathbf{X} = \mathbf{U}diag(\theta)\mathbf{U^T}\mathbf{X} \tag{2.5}$$

  where $\mathbf{U} \in \mathbb{R}^{N \times N}$ is the eigenvectors of $\mathbf{L}$ and $\mathbf{U^T}$ is the transpose of $\mathbf{U}$. $g_\theta = diag(\theta)$ is the filter parameterised by $\theta \in \mathbb{R}^N$ and $\mathbf{SGC} \in \mathbb{R}^{B \times T \times N}$ is the output of spectral GCN.

- **Diffusion GCN** was proposed in (Li et al. 2018) and performed on a directed graph. Diffusion GCN models the bidirectional diffusion process, which enables the model to capture the influence of upstream and downstream traffic. It can be defined as Eq. (2.6) and Eq. (2.7):

$$\mathrm{X}_{\mathrm{t}*_{\mathcal{G}}\mathrm{f}_{\theta_{\mathbf{d}}}} = \sum_{k=0}^{K-1} \left(\theta_{\mathrm{k},1} (\mathbf{D_o^{-1}} \mathbf{A_w})^k + \theta_{\mathrm{k},2} (\mathbf{D_I^{-1}} \mathbf{A_w}^{\mathbf{T}})^k \right) \mathrm{x}_{\mathrm{t}} \quad for\ t = 1, ...T \tag{2.6}$$

where $\mathbf{D_o}$ and $\mathbf{D_I}$ are the out-degree and in-degree diagonal matrices respectively, and $\theta_{\mathrm{k},1} \in \mathbb{R}^K$ and $\theta_{\mathrm{k},2} \in \mathbb{R}^K$ are responding weight vectors. $\mathbf{A_w}$ is the weighted adjacent matrix and $\mathbf{A_w}^{\mathbf{T}}$ is the transpose of $\mathbf{A_w}$. $\theta_{\mathbf{d}} \in \mathbb{R}^{K \times 2}$ is the parameters for the filter, and $(\mathbf{D_o^{-1}} \mathbf{A_w})$ and $(\mathbf{D_I^{-1}} \mathbf{A_w}^{\mathbf{T}})$ represent the transition matrices of the diffusion process and the reverse one, respectively.

$$\mathbf{DGC} = \sigma \left(\sum_{t=1}^{T} \mathrm{X}_{\mathrm{t}*_{\mathcal{G}}\mathrm{f}\theta_{\mathbf{d}_{\mathbf{t},\mathbf{t}'}}} \right) for\ t' = 1, ...T' \tag{2.7}$$

where $\mathbf{DGC} \in \mathbb{R}^{B \times T' \times N}$ is the output of diffusion GCN and σ is the activation function like Sigmoid or ReLU. $f\theta_{d_{t;t'}}$ are the filters and its parameters are $\theta \in \mathbb{R}^{T' \times T \times K \times 2}$.

- **Traffic GCN** in (Cui et al. 2019) directly conducts convolutional operation on the road graph by Eq. (2.8).

$$\mathbf{GC} = \sigma \left((\mathbf{W_{gc}} * \tilde{\mathbf{A}}^k) \mathbf{X} \right) \tag{2.8}$$

where $*$ is the Hadamard product operator (i.e., element-wise matrix multiplication operator). $\mathbf{W_{gc}} \in \mathbb{R}^{N \times N}$ is a trainable weight matrix and $\mathbf{GC} \in \mathbb{R}^{B \times T \times N}$ is the output of GCN. σ is the activation function like ReLU (Li & Yuan 2017).

Modelling Features in Time Domain

Long Short-Term Memory (LSTM): LSTM is an extension of the RNN model (Hochreiter & Schmidhuber 1997). Compared to RNN that has one part (i.e., the $tanh$ layer), LSTM consists of four parts: three gates (namely, input gate $\mathrm{I_t}$, output gate $\mathrm{O_t}$ and forget gate $\mathrm{F_t}$) and a cell state ($\mathrm{C_t}$). The forget gate $\mathrm{F_t}$ with a sigmoid layer σ_g firstly determines the part of information in current traffic data $\mathrm{x_t}$ and in the last hidden state $\mathrm{h_{t-1}}$ that needs to be forgotten and updated to the cell state $\mathrm{C_t}$ via Eq. (2.9). To supplement the forgotten

information by forget gate $\mathrm{F_t}$, the input gate $\mathrm{I_t}$ with a sigmoid layer σ_g is used to decide the information from current traffic data $\mathrm{x_t}$ to be added into the cell state $\mathrm{C_t}$ via Eq. (2.10). Then, the cell state $\mathrm{C_t}$ is updated using the tangent layer σ_C (cf. Eq. (2.12)) for integrating the traffic information provided from the forget gate, the input gate and the last cell state $\mathrm{C_{t-1}}$. Meanwhile, the output gate with a sigmoid layer σ_g selects previous information remembered by $\mathrm{h_{t-1}}$ and the current information $\mathrm{x_t}$ by Eq. (2.11) for contributing to final output. Finally, the predicted result is computed by combining remembered information from the output gate $\mathrm{O_t}$ and the cell state $\mathrm{C_t}$ with a tangent layer σ_H by Eq. (2.13).

$$\mathrm{F_t} = \sigma_g(\mathrm{W_F} \times \mathrm{x_t} + \mathrm{U_F} \times \mathrm{h_{t-1}} + b_F) \tag{2.9}$$

$$\mathrm{I_t} = \sigma_g(\mathrm{W_I} \times \mathrm{x_t} + \mathrm{U_I} \times \mathrm{h_{t-1}} + b_I) \tag{2.10}$$

$$\mathrm{O_t} = \sigma_g(\mathrm{W_O} \times \mathrm{x_t} + \mathrm{U_O} \times \mathrm{h_{t-1}} + b_O) \tag{2.11}$$

$$\mathrm{C_t} = \mathrm{F_t} * \mathrm{C_{t-1}} + \mathrm{I_t} * \sigma_C(\mathrm{W_C} \times \mathrm{x_t} + \mathrm{U_C} \times \mathrm{h_{t-1}} + b_C) \tag{2.12}$$

$$\mathrm{h_t} = \mathrm{O_t} * \sigma_H \times (\mathrm{C_t}). \tag{2.13}$$

where $\mathrm{W_F}, \mathrm{W_I}, \mathrm{W_O}$ and $\mathrm{W_C}$ are the weight matrices of the forget gate, the input gate, the output gate and the cell state respectively while b_F, b_I, b_O and b_C are the corresponding bias for each gate and state. Furthermore, $\mathrm{U_F}, \mathrm{U_I}, \mathrm{U_O}$ and $\mathrm{U_C}$ are the weight vectors of the last hidden state $\mathrm{h_{t-1}}$. σ_g is used to denote a sigmoid function (= $\frac{1}{1+e^{-x}}$) in three gates and the operator $*$ denotes Hadamard product. σ_C and σ_H are hyperbolic tangent functions ($tanh(x)$) for the cell state and the final output.

Gated Recurrent Unit (GRU): GRU (Chung et al. 2014) is developed based on LSTM, and it incurs shorter processing time and less Central Processing Unit (CPU) cycles. The reason is that GRU combines LSTM's forget and input gates into a single "update gate", and also merges the memory cell and hidden state. This makes GRU simpler than the standard LSTM but still retain good performance. GRU consists of three parts: the update gate $\mathrm{z_t}$, the reset gate $\mathrm{r_t}$ and the hidden state $\mathrm{h_t}$. The update gate, $\mathrm{z_t}$, extracts the long-term dependency of the traffic data. It decides how much information it needs to update from

the input $\mathrm{x_t}$ and the hidden state at the previous time interval $\mathrm{h_{t-1}}$ (cf. Eq. (2.14)).

$$\mathrm{z_t} = \sigma(\mathrm{W_z} \times \mathrm{x_t} + \mathrm{U_z} \times \mathrm{h_{t-1}} + b_z) \tag{2.14}$$

The reset gate, $\mathrm{r_t}$, captures the short-term dependency of traffic features. It decides how much information from the hidden state at the previous time interval is retained for updating the current hidden state. It is computed in a similar manner as the update gate by Eq. (2.15).

$$\mathrm{r_t} = \sigma(\mathrm{W_r} \times \mathrm{x_t} + \mathrm{U_r} \times \mathrm{h_{t-1}} + b_r) \tag{2.15}$$

Then, the input $\mathrm{x_t}$, the reset gate $\mathrm{r_t}$ and the hidden state at the previous time interval $\mathrm{h_{t-1}}$ are used to activate the candidate hidden state $\tilde{\mathrm{h}}_\mathrm{t}$ via Eq. (2.16).

$$\tilde{\mathrm{h}}_\mathrm{t} = tanh(\mathrm{W_h} \times \mathrm{x_t} + \mathrm{U_h} \times (\mathrm{r_t} * \mathrm{h_{t-1}}) + b_h) \tag{2.16}$$

where $\mathrm{W_z}$, $\mathrm{W_r}$ and $\mathrm{W_h}$ are the weight vectors of the update gate, the reset gate and the candidate hidden state respectively while b_z, b_r and b_h are the corresponding bias for each gate and state. Furthermore, $\mathrm{U_z}$, $\mathrm{U_r}$ and $\mathrm{U_h}$ are the weight vectors of the hidden state at the previous time interval $\mathrm{h_{t-1}}$ in the update gate, the reset gate and the candidate hidden state, respectively. Finally, the current hidden state can be calculated using the update gate $\mathrm{z_t}$, the hidden state at the previous time interval $\mathrm{h_{t-1}}$ and the current candidate hidden state $\tilde{\mathrm{h}}_\mathrm{t}$ using Eq. (2.17).

$$\mathrm{h_t} = (1 - \mathrm{z_t}) * \mathrm{h_{t-1}} + \mathrm{z_t} * \tilde{\mathrm{h}}_\mathrm{t} \tag{2.17}$$

Sequence-to-Sequence Architecture:

The Sequence-to-Sequence (Seq2Seq) architecture was developed in (Sutskever et al. 2014) and has already been found to offer good performance in the area of natural language processing. The architecture consists of an encoder and a decoder with a context connecting the two. The encoder firstly encodes the input information into a context and then it is decoded into the output by the decoder. For traffic prediction problem, it encodes the spatially-fused time series using Eq. (2.18).

$$\mathrm{h}^{\mathrm{e}}_{\mathrm{t}-\mathrm{t_e}} = \begin{cases} f_{encoder}(\mathrm{h}^{\mathrm{e}}_{0}, \mathrm{x}_{\mathrm{t}-\mathrm{t_e}}), & t_e = T \\ f_{encoder}(\mathrm{h}^{\mathrm{e}}_{\mathrm{t}-\mathrm{t_e}-1}, \mathrm{x}_{\mathrm{t}-\mathrm{t_e}}), & t_e \in 0, \ldots, T-1 \end{cases} \tag{2.18}$$

where $\mathrm{h}^{\mathrm{e}}_{\mathrm{t}-\mathrm{t_e}}$ is the hidden state in the encoder at $(t-t_e)^{th}$ time interval. The initial hidden state is $\mathrm{h}^{\mathrm{e}}_{0}$. The hidden state $\mathrm{h}^{\mathrm{e}}_{\mathrm{t}-\mathrm{t_e}-1}$ at $(t-t_e-1)^{th}$ time interval and the spatially-fused time series $\mathrm{x}_{\mathrm{t}-\mathrm{t_e}}$ at $(t-t_e)^{th}$ time interval are used to calculate the hidden state $\mathrm{h}^{\mathrm{e}}_{\mathrm{t}-\mathrm{t_e}}$ at $(t-t_e)^{th}$ time interval.

The hidden state $\mathrm{h}^{\mathrm{e}}_{\mathrm{t}}$ ($t_e = 0$) at the t^{th} time interval is considered as the context vector C (cf. Eq. (2.19)) which encodes all information from the input $\mathbf{X}$ in the encoder.

$$\mathrm{C} = \mathrm{h}^{\mathrm{e}}_{\mathrm{t}} \tag{2.19}$$

In the decoder, the context vector C as the initial hidden state $\mathrm{h}^{\mathrm{d}}_{0}$ is decoded to the target sequence. The hidden state $\mathrm{h}^{\mathrm{d}}_{\mathrm{t}+\mathrm{t_d}-1}$ at $(t+t_d-1)^{th}$ time interval and the target traffic speed $\mathrm{x}_{\mathrm{t}+\mathrm{t_d}}$ at $(t+t_d)^{th}$ time interval are utilised to calculate the hidden state $\mathrm{h}^{\mathrm{d}}_{\mathrm{t}+\mathrm{t_d}}$ at $(t+t_d)^{th}$ time interval. The hidden state $\mathrm{h}^{\mathrm{d}}_{\mathrm{t}+\mathrm{t_d}}$ at $(t+t_d)^{th}$ time interval in the decoder is considered as the final prediction $\tilde{\mathrm{x}}_{\mathrm{t}+\mathrm{t_d}}, t_d = 1, \ldots, T'$.

Modelling Features in Space and/or Time Domain

Attention Mechanism:

To address translation accuracy as the length of an input sentence increases in field of neural machine translation (Kalchbrenner & Blunsom 2013), Bahdanau *et. al* (Bahdanau et al. 2015) developed an extension of the encoder–decoder model which learns to align and translate jointly. This method solves the long-term dependencies by measuring the similarity of the input at each observed position and the output at the targeted position. Meanwhile, this approach has evolved and used to address long-term dependencies in many areas such as video captioning (Yan et al. 2019), image classification (Wang et al. 2017), speech recognition (Chorowski et al. 2015), traffic flow prediction (Guo et al. 2019) and so on. For solving traffic prediction problem, the attention mechanism has usually been used for strengthening important features and weakening unimportant features in space and/or time domains towards to final prediction. We elaborate below the working principle of the attention mechanism in the decoder of the Seq2Seq architecture (Zhang, Li, Lin, Wang & He 2019) as an example.

The similarity of traffic data between observed time interval t and targeted time interval t', $\mathrm{u}_{\mathrm{t},\mathrm{t}'}$, is computed via a $tanh$ function as Eq. (2.20).

$$\mathrm{u}_{\mathrm{t},\mathrm{t}'} = q^T tanh\left(\mathrm{h}_{\mathrm{t}}^{\mathrm{e}} \mathrm{W}_{\mathrm{t},\mathrm{t}'} \mathrm{h}_{\mathrm{t}'}^{\mathrm{d}}\right); t = 1, 2, \ldots, T \tag{2.20}$$

where $\mathrm{W}_{\mathrm{t},\mathrm{t}'}$ is a trainable weight vector and q^T represents the transposition or reshaping operations that are utilised to adjust the dimensions. We then compute the attention weights as probabilities (i.e., $\mathrm{a}_{\mathrm{t},\mathrm{t}'} \in [0.0, 1.0]$) via a $softmax$ function given in Eq. (2.21).

$$\mathrm{a}_{\mathrm{t},\mathrm{t}'} = softmax\left(\mathrm{u}_{\mathrm{t},\mathrm{t}'}\right) = \frac{exp\left(\mathrm{u}_{\mathrm{t},\mathrm{t}'}\right)}{\sum_{t=1}^{T} exp\left(\mathrm{u}_{\mathrm{t},\mathrm{t}'}\right)} \tag{2.21}$$

After obtaining the attention weights $\mathrm{a}_{\mathrm{t},\mathrm{t}'}$, it is used to map to hidden state $\mathrm{h}_{\mathrm{t}}^{\mathrm{e}}$ for achieving targeted traffic state by Eq. (2.22).

$$\mathrm{x}_{\mathrm{t}'} = \sum_{t=1}^{T} \mathrm{a}_{\mathrm{t},\mathrm{t}'} \times \mathrm{h}_{\mathrm{t}}^{\mathrm{e}}. \tag{2.22}$$

Transformer: Transformer was proposed in the paper "Attention is All You Need", which makes use of the self-attention mechanism under the encoder and decoder architecture and outperforms the Google Neural Machine Translation model in specific tasks (Vaswani et al. 2017). Based on such success, transformer is applied into solving network-wide traffic prediction problem by using self-attention mechanism to capture dynamic-spatial and -temporal dependencies (e.g. (Xu et al. 2020) and (Zheng et al. 2020)). For a single-head self-attention mechanism used to analyse dynamic-spatial dependencies as an example, the input consists of queries $\mathbf{Q^s} \in \mathbb{R}^{N \times d_q}$ of dimension d_q, keys $\mathbf{K^s} \in \mathbb{R}^{N \times d_k}$ of dimension d_k, and values $\mathbf{V^s} \in \mathbb{R}^{N \times d_v}$ of dimension d_v, and they are computed by Eq. (2.23).

$$\begin{aligned} \mathbf{Q^s} &= \mathbf{W_q^s} \mathrm{x}_{\mathrm{t}} \\ \mathbf{K^s} &= \mathbf{W_k^s} \mathrm{x}_{\mathrm{t}} \\ \mathbf{V^s} &= \mathbf{W_v^s} \mathrm{x}_{\mathrm{t}} \end{aligned} \tag{2.23}$$

where $\mathbf{W_q^s}$, $\mathbf{W_k^s}$ and $\mathbf{W_v^s}$ are learnable weight matrices for $\mathbf{Q^s}$, $\mathbf{K^s}$, and $\mathbf{V^s}$, respectively,

and its random initial values are updated by the back propagation. After obtaining the three high dimensional spatial features, dynamic-spatial dependencies $\mathbf{S^s} \in \mathbb{R}^{N\times N}$ are calculated by dot-product as Eq. (2.24)

$$\mathbf{S^s} = softmax(\frac{\mathbf{Q^s}(\mathbf{K^s})^\mathbf{T}}{\sqrt{d_k}}) \tag{2.24}$$

where $\mathbf{S^s}$ is the learned dynamic dependency matrix and defined by how each sensor is influenced by all the other sensors in the road network. Then the new spatial features $\mathbf{M^s} \in \mathbb{R}^{N\times N}$ are updated by Eq. (2.25).

$$\mathbf{M^s} = \mathbf{S^s}\mathbf{V^s} \tag{2.25}$$

Note that multiple spatial dependencies can be learned with multi-heads attention mechanism by learning multiple pairs so as to reveal different hidden spatial dependencies from various latent spaces. Furthermore, two feed-forward neural network layers with non-linear activation are used to improve the prediction ability, and its output is added to the input of the self-attention mechanism to build the residual connection for stable training as Eq. (2.26).

$$\mathrm{Y^s} = ReLU(\mathrm{W_1^s}ReLU(\mathrm{W_0^s}\mathbf{M^s})) + \mathrm{x_t} \tag{2.26}$$

where $\mathrm{W_0^s}$ and $\mathrm{W_1^s}$ are weight vectors of two feed-forward neural network layers. Similar to the analysis of dynamic-spatial dependencies $\mathrm{Y^s}$, dynamic-temporal dependencies $\mathrm{Y^t}$ can be obtained by the same idea. To fuse dynamic-spatial and -temporal features, the gated mechanism (cf. Eq. (2.27)) is introduced based on the gated mechanism in GRU and the final output is calculated by Eq. (2.28).

$$\mathrm{z} = Sigmoid(\mathrm{Y^s}\mathrm{W_{zs}} + \mathrm{Y^t}\mathrm{W_{zt}} + b_z) \tag{2.27}$$

$$\tilde{\mathrm{x}}'_\mathrm{t} = \mathrm{z} * \mathrm{Y^s} + (1 - \mathrm{z}) * \mathrm{Y^t} \tag{2.28}$$

where $\mathrm{W_{zs}}$ and $\mathrm{W_{zt}}$ are weight matrices of the dynamic-spatial and -temporal features respectively and b_z is the bias term. z is the gate used to fuse both dynamic features. $\tilde{\mathrm{x}}'_\mathrm{t}$ is the output at t'^{th} time interval.

2.6 Public Data Sources

2.6.1 List of Public Data Sources

In this section, we summarise well-known public data sources that are commonly used for solving various traffic prediction problems including traffic flow, traffic speed and travel time prediction problems, in literature. Besides, we also elaborate the three large datasets of them that are used for our research aiming to predict traffic states on large urban road networks.

- **Caltrans Performance Measurement System (PeMS):** The traffic data from PeMS is collected in real-time from over 39,000 individual detectors across all major metropolitan areas of the State of California. It includes traffic speed, flow, occupancy, vehicle miles travelled (VMT), vehicle hours travelled (VHT) and Q (VMT/VHT). The time interval in PeMS is 5 minutes (Caltrans n.d.). There are several public traffic datasets from PeMS as follows.

  1. `PEMS03` (Song et al. 2020): This dataset is collected from 358 sensors with the period of 1^{st} of September and 30^{th} of November in 2018.
  2. `PEMS04` (Song et al. 2020): This dataset is collected from 307 sensors with the period of 1^{st} of January and 28^{th} of February in 2018.
  3. `PEMS07` (Song et al. 2020): This dataset is collected from 883 sensors with the period of 1^{st} of May and 31^{st} of August in 2017.
  4. `PEMS08` (Song et al. 2020): This dataset is collected from 170 sensors with the period of 1^{st} of July and 31^{st} of August in 2016.
  5. `PEMS-SF` (Cuturi 2011): This dataset covers 15 months worth of daily data from 1^{st} of January in 2008 to 30^{th} of March in 2009, and describes the occupancy rate, between 0 and 1, of different car lanes of San Francisco Bay Area freeways. The internal data is sampled every 10 minutes.
  6. `PEMS-BAY` (Li et al. 2018): This traffic speed dataset is collected from 325 sensors in the Bay Area and covers 6 months of data ranging from 1^{st} of January to 3^{th} of May in 2017.

- **Highways of Los Angeles:** One of traffic datasets collected from loop detectors in the highways of Los Angeles is named `METR-LA` (Li et al. 2018). This dataset includes traffic speed from 207 sensors ranging from 1^{st} of March to 30^{th} of June in 2012.

- **Seattle area traffic data:** The traffic dataset collected from freeways of I-50, I-405, I-90 and SR520 in Seattle area is named `LOOP-SEATTLE` (Cui et al. 2018, 2019). It describes traffic speed data from 323 loop detectors over the entirely of 2015 at a 5-minute time interval.
- **UK traffic data:** This data source provides average journey time, traffic speed and flow information for 15-minute periods since April 2015 on all motorways and "A" roads managed by Highways England, known as the Strategic Road Network, in England (England n.d.).
- **Beijing traffic data:** This is a large-scale real-world traffic speed prediction dataset (named `Q-traffic-dataset`) collected through Baidu map. It consists of three sub-datasets: 1) query sub-dataset recording the starting timestamp, coordination of the starting location and the destination, and the estimated travel time; 2) traffic speed sub-dataset recording traffic speed from 15,073 road segments covering approximately 738.91 km; 3) road network sub-dataset recording the topology of the road network (Liao et al. 2018).
- **NYC traffic data:** This dataset collected from taxi trips in New York City includes pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances itemised dares, rate types, payment types and driver-reported passenger counts from 2009 to now (Government n.d.).
- **San Francisco taxi traffic data:** This dataset contains mobility traces of taxi cabs in San Francisco and is collected from about 500 taxis over 30 days. It can be used for tasks like NYC traffic data (Michal et al. n.d.).
- **Traffic data from the Illinois Department of Transportation:** This system provides traffic flow, speed and travel time, and can be used for tasks like UK traffic data (webmaster n.d.).
- **DiDi traffic data:** DiDi GAIA data open program provides real-world traffic data including travel time index, travel and trajectory datasets (Alumni n.d.).

2.6.2 Three Public Datasets Used in Our Research

`PEMS-BAY`

The `PEMS-BAY` dataset is collected from California Transportation Agencies (CalTrans) Performance Measurement System (PeMS) (Chen 1994) and mentioned in Section 2.6.1.

It contains traffic speed data from 325 sensor stations in the Bay Area. The locations of loop detectors from this road networks is presented in Figure 2.3. This dataset covers traffic measurements for a period of six months between the 1^{st} of January and 30^{th} of June in 2017. The time interval for the data is 5 minutes and the total number of observed traffic data points is 16,937,700 (= 52,116 $\times$ 325).

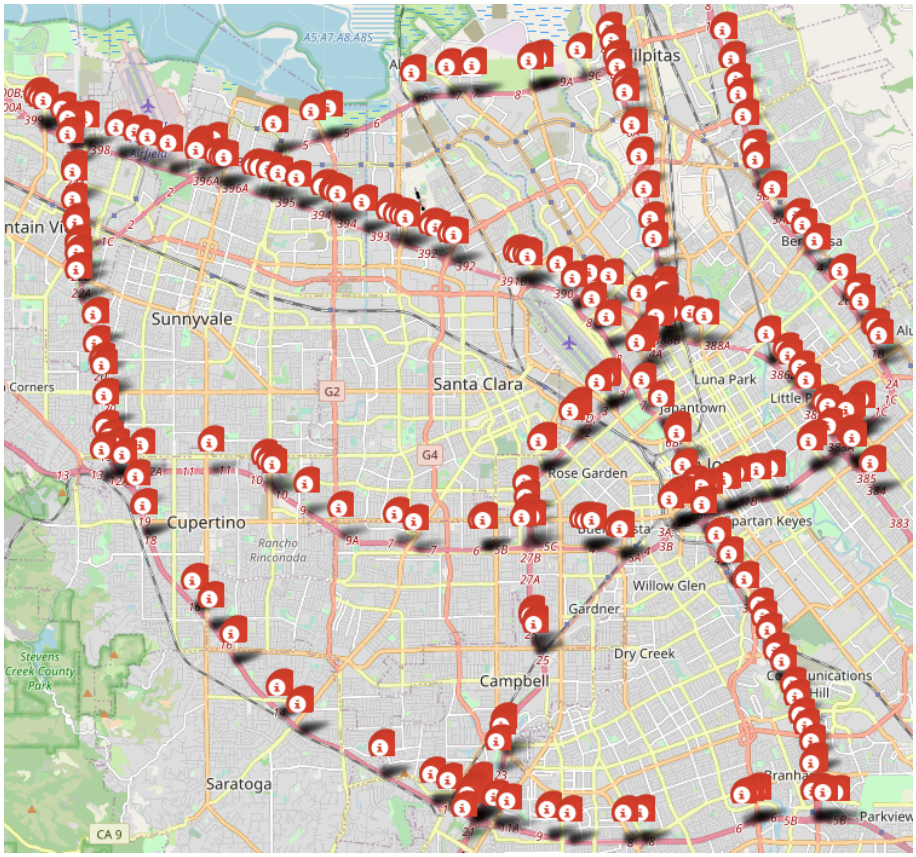


Figure 2.3: Locations of loop detectors in `PEMS-BAY` dataset.

`LOOP-SEATTLE`

The `LOOP-SEATTLE` dataset is collected from inductive loop detectors deployed on four connected freeways (i.e. I-5, I-405, I-90 and SR-520) in the Greater Seattle Area, and the locations of the loop detectors are presented by the red pins in Figure 2.4. This dataset records traffic information from 323 detectors over the entirety of year 2015 at a 5-min time interval. The unweighted and non-directional graph is used to represent this road network.

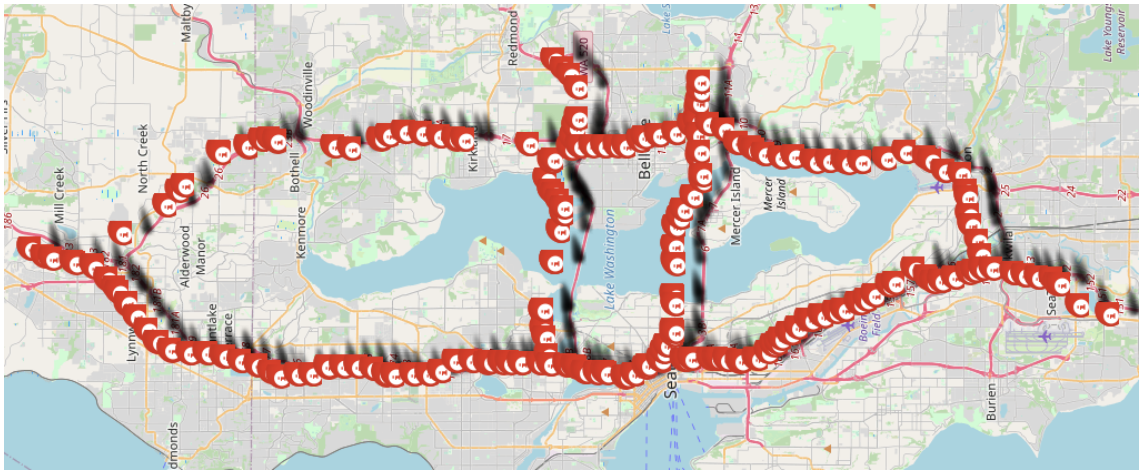


Figure 2.4: Locations of loop detectors in LOOP−SEATTLE dataset.

METR−LA

The METR−LA dataset is collected from loop detectors in the highways of Los Angeles County (Jagadish et al. 2014). Similarly, the locations of detectors in this road network are shown via the red pins in Figure 2.5. It includes 207 detectors and covers 4 months of traffic speed data from the 1^{st} of March to the 30^{th} of June in 2012. In this dataset, a non-directional graph with edge weights is used to construct the adjacency matrix. The pairwise road distances between detectors are first computed and then a thresholded Gaussian Kernel (Shuman et al. 2013) is used to build the adjacency matrix. The edge weights are calculated by Eq. (2.29) below:

$$W_{i,j} = \begin{cases} exp(-\frac{dist(i,j)^2}{2\sigma^2}), & if\ dist(i,j) < d_{threshold} \\ 0, & otherwise \end{cases} \tag{2.29}$$

where $W_{i,j}$ is the edge weight between node i and node j, and $dist(i,j)$ represents the actual physical road distance between node i and node j in the road network. The standard deviation of the distances is denoted by σ and $d_{threshold}$ is the threshold.

Table 2.2 provides the basic statistics of all three datasets including maximum (Max), minimum (Min), mean value (Mean), standard deviation (Std) and variance (Var) of traffic speed data as well as the size of dataset (in MByte). From the table, we note that METR−LA has higher traffic fluctuations with larger standard deviation and variance than PEMS−BAY followed by LOOP−SEATTLE.

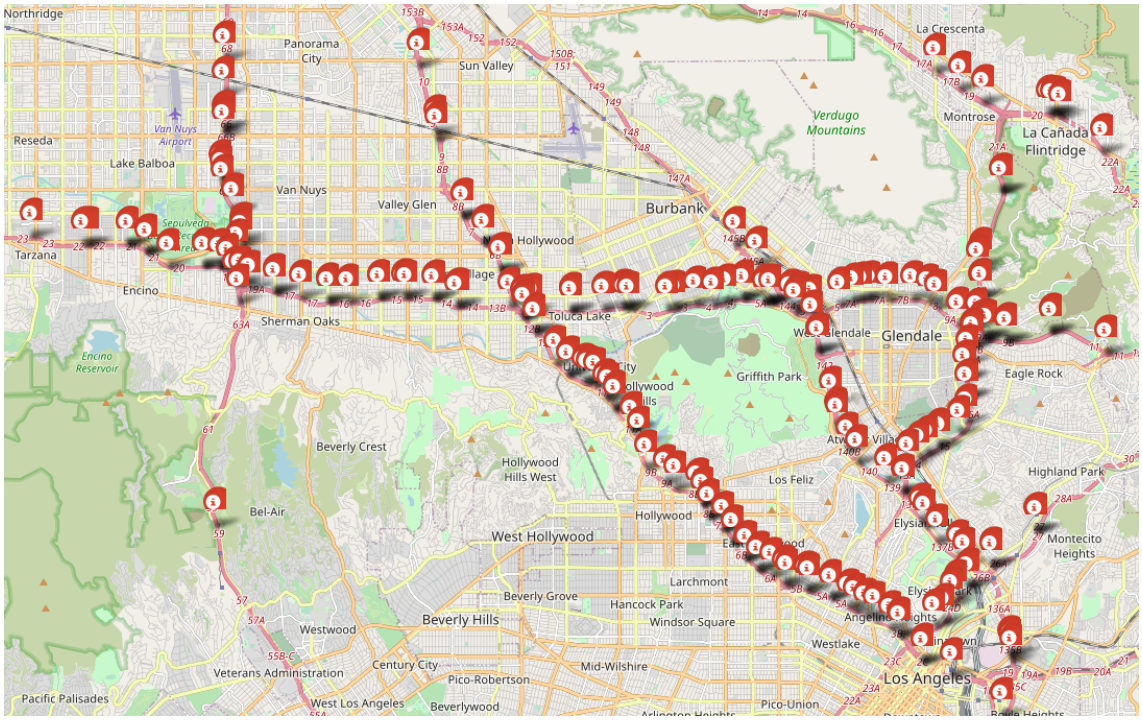


Figure 2.5: Locations of loop detectors in METR-LA dataset.

Table 2.2: Characteristics of three public traffic speed (miles/hour) datasets.

Dataset	Max	Min	Mean	Std	Var	Size
PEMS-BAY	85.10	0.00	62.62	8.56	85.41	135.90 MB
LOOP-SEATTLE	158.19	0.74	56.57	11.43	147.25	274.40 MB
METR-LA	70.00	0.00	53.72	19.19	374.85	57.00 MB

2.7 Performance Evaluation Methods

Three conventional performance metrics commonly are used to evaluate the models or approaches of traffic prediction in literature (Tan et al. 2009, Huang et al. 2014, Lv et al. 2015), namely Mean Absolute Error (MAE), Root-Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). They are computed as Eq. (2.30), Eq. (2.31) and Eq. (2.32), respectively.

$$MAE = \frac{1}{N \times T'} \cdot \sum_{i=1}^{N} \sum_{t=1}^{T'} |x_t^i - \tilde{x}_t^i| \tag{2.30}$$

$$RMSE = \left[\frac{1}{N \times T'} \cdot \sum_{i=1}^{N} \sum_{t=1}^{T'} (x_t^i - \tilde{x}_t^i)^2 \right]^{\frac{1}{2}} \tag{2.31}$$

$$MAPE = \frac{1}{N \times T'} \cdot \sum_{i=1}^{N} \sum_{t=1}^{T'} \frac{|x_t^i - \tilde{x}_t^i|}{x_t^i} \times 100\%. \tag{2.32}$$

where MAE presents the average absolute difference between the real and predicted traffic data. It is used to measure absolute prediction error. RMSE is the standard deviation of the residuals, which is the difference between the real and predicted traffic data. MAPE is the percentage of MAE to the real traffic data and is utilised to measure the prediction error. Furthermore, some existing works, such as (Lv et al. 2015, Huang et al. 2014), also define the accuracy of traffic prediction as $(100 - MAPE)\%$.

2.8 Summary

In this chapter, we first discussed the general formulation of the traffic prediction problems solved in literature and then review their solutions based on their methodologies. Models from those solutions could be categorised into three groups including statistical models, ML models and DL models. Therefore, we review existing works from the perspectives of these three groups.

The latest technology used for solving traffic prediction problem is based on DL, specifically hybrid DL models. We review and explain the latest hybrid DL models that are grouped into three: CNN-based, GCN-based and Transformer-based models. Besides, we also summarise modules used in hybrid DL models based on its function on extracting different types of features for final prediction.

Furthermore, we also review and list public data sources that have been used for various traffic prediction tasks such as traffic flow, speed, travel time and so on. Three of public datasets have been explained in greater detail and used in our research work here. Finally, we present the performance evaluation methods used in literature.

Chapter 3

Spatial-Temporal Feature-based Short-Term Traffic Flow Prediction on Linear Roadways

3.1 Introduction

This chapter focuses on solving the short-term traffic flow prediction problem on linear roadways (i.e., segment of road without intersections or junctions), and the aim is to predict the number of vehicles in a targeted region on linear roadways over a short time interval. Traditionally, short-term traffic flow prediction problem on linear roadways was addressed as a pure temporal process. Recently, spatial information related to the road network has been found to be important in improving traffic prediction accuracy (Lv et al. 2015, Wu & Tan 2016). Therefore, we consider the problem as a joint temporal and spatial process.

The literature reviewed in Chapter 2 indicates (1) better quality data (in volume and granularity), (2) combinations of prediction models and (3) joint consideration of temporal and spatial data all contribute to better short-term traffic flow prediction. In this chapter, we exploit all these three elements and develop a new ensemble model (EM) to predict short-term traffic on linear roadways based on three modules, namely long short term memory (LSTM), deep autoencoder (DAE) and convolution neural network (CNN). For building EM model, we consider both memory-based and memory-less models due to aiming to extract both temporal and spatial features for traffic prediction. Memory-based model of LSTM is used to remember important temporal information from historical data

while two memory-less models of DAE and CNN are utilised to analyse spatial patterns of historical data.

The rest of this chapter is organised as follows. We elaborate our approach for spatial-temporal feature-based short-term traffic flow prediction on linear roadways, including the problem definition, the architecture of our proposed model and its constituent modules, in Section 3.2. Furthermore, real-world traffic data collected from two different locations used in the evaluation of our model is described in Section 3.3 while the performance of our model compared against several existing models are shown in Section 3.4. Finally, Section 3.5 summarises our research work for short-term traffic flow prediction on linear roadways.

3.2 Methodology

In this Section, we show our approach for solving the short-term traffic flow prediction on linear roadways, which includes the problem definition, the input matrix, the architecture of our novel EM and its individual modules.

3.2.1 Problem Definition

Considering a linear roadway with a set of monitoring locations (e.g., loop detectors), let x_t^i be the number of vehicles recorded (i.e., traffic flow) at the i^{th} observation location at the $(t-1, t]$ time interval. The full traffic flow information across a linear roadway over a period can then be described with a sequence $x_t^i : \forall i, t$. The problem is then to predict the traffic flow at location i at the $(t + \Delta)$ time interval given the set of x_t^i and the prediction horizon (Δ) (Manual 2000). Typically, the time interval could be 15, 30, 45, and 60 minutes (Lv et al. 2015).

Furthermore, we also consider vehicle speed as another indicator for the future traffic flow. In fact, (Zhao et al. 2014) found that there exists a non-linear relationship between traffic flow and vehicle speed. As such, we also use vehicle speed data as input to our problem. We denote average vehicle speed at t^{th} time interval at location i as s_t^i.

3.2.2 Input Matrix

As prior mentioned in Section 3.2.1, our approach takes both the raw data of traffic flow (x_t^i) and average vehicle speed (s_t^i) over time at different locations as input. These inputs are first de-noised via a moving average (MA) filter producing the pre-processed traffic data on traffic flow ($\hat{x}_t^i = (\sum_{j=0}^{n} x_{t+j}^i)/(n+1); \forall i$) and vehicle speed ($\hat{s}_t^i = (\sum_{j=0}^{n} s_{t+j}^i)/(n+1); \forall i$).

The outputs of MA filter (both $\hat{x}_t^i$ and $\hat{s}_t^i$) are then integrated into a single matrix, $\mathbf{A_t}$ with a number of features. Specific to our work here, we extract 36 features from the input data (including for example traffic flow, traffic flow differences and speed at different observation stations over time) as shown in Table 3.1. The traffic flow, traffic speed and traffic flow difference of $i-2$, $i-1$, i, $i+1$ and $i+2$ in three previous time intervals are used to predict traffic flow at i at $t+\Delta$. We consider Δ = 15 minutes here. For example, from Table 3.1, x_{t-1}^{i-1} and s_{t-1}^{i-1} are respectively the traffic flow and traffic speed in the time interval $t-1$ of station $i-1$. δ_{t-1}^{i-1} is the traffic flow difference between the time interval $t-1$ and the time interval $t-2$.

3.2.3 The Architecture of EM

To account for both temporal and spatial effects on traffic flow, the framework of our prediction model (hereafter simply referred to as the Ensemble Model (EM)) is designed to draw insights from both sequential and spatial data and their correlation to make its prediction. To this end, we exploit specific prediction capabilities of three modules, namely the long-short term memory (LSTM), deep autoencoder (DAE) and convolution neural network (CNN). Specifically, we use the long memory property of LSTM to extract temporal features of traffic conditions. In fact, LSTM has already been found to be effective in dealing with sequential data (Hochreiter & Schmidhuber 1997). Furthermore, (Finn et al. 2016) have found that DAE has good capability in extracting global spatial features while in (Yao et al. 2018), CNN is found to be well-suited in relation to local spatial features. As such, we also integrate both DAE and CNN into our EM model.

Figure 3.1 shows the architecture of our EM. We can directly use our input matrix $\mathbf{A_t}$ for DAE module while, for LSTM and CNN modules, we first need to reshape their respective input to the required data input dimensions (in our case, 1×36 and $6 \times 6 \times 1$ respectively). The three outputs from the individual modules are used as three separate inputs for EM hidden layers. These hidden layers have the same number of neural units and are used to

Table 3.1: The features of Input Data ($\mathbf{A_t}$).

x_{t-1}^{i-2}	x_{t-2}^{i-2}	x_{t-3}^{i-2}	x_{t-1}^{i-1}	x_{t-2}^{i-1}	x_{t-3}^{i-1}	x_{t-1}^{i}	x_{t-2}^{i}	x_{t-3}^{i}	x_{t-1}^{i+1}	x_{t-2}^{i+1}	x_{t-3}^{i+1}	x_{t-1}^{i+2}	x_{t-2}^{i+2}	x_{t-3}^{i+2}
62	50	47	53	36	31	76	74	64	41	38	26	39	40	37
50	47	42	36	31	31	74	64	65	38	26	18	40	37	28
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
s_{t-1}^{i-2}	s_{t-2}^{i-2}	s_{t-3}^{i-2}	s_{t-1}^{i-1}	s_{t-2}^{i-1}	s_{t-3}^{i-1}	s_{t-1}^{i}	s_{t-2}^{i}	s_{t-3}^{i}	s_{t-1}^{i+1}	s_{t-2}^{i+1}	s_{t-3}^{i+1}	s_{t-1}^{i+2}	s_{t-2}^{i+2}	s_{t-3}^{i+2}
67.2	65.8	66.8	67.8	68.3	67.5	64	64.1	64.5	67.7	68.3	68.8	68.1	69.3	68.8
65.8	66.8	64.7	68.3	67.5	66.4	64.1	64.5	64.5	68.3	68.8	67.3	69.3	68.8	70.4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
δ_{t-1}^{i-1}	δ_{t-2}^{i-1}	δ_{t-1}^{i}	δ_{t-2}^{i}	δ_{t-1}^{i+1}	δ_{t-2}^{i+1}									
-17	-5	-2	-10	-3	-12									
-5	0	-10	1	-12	-8									
⋮	⋮	⋮	⋮	⋮	⋮									

map the outputs of the three individual modules to the same dimensions. This guarantees every module provides the same number of features for the $softmax$ layer. This ensures that each module's output is equally treated in the next EM layer. The three outputs from respective hidden layer are merged (as $\mathbf{B_t}$) before being fed to the $softmax$ layer as the final predictor.

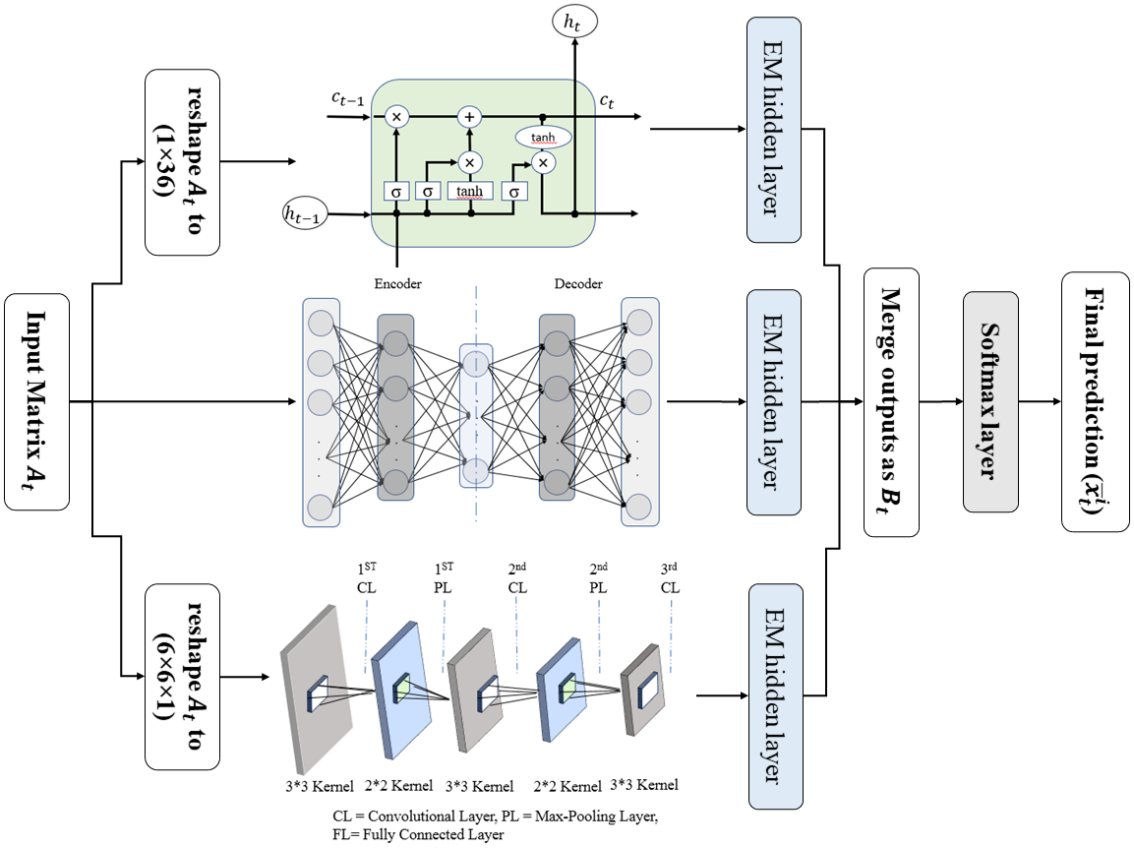


Figure 3.1: The architecture of our EM model.

3.2.4 Individual Modules

LSTM Module

The explanation of LSTM module can be found in Chapter 2.5.3.

DAE Module

Deep AutoEncoder (DAE) is an unsupervised learning technique that is capable of learning non-linear relationship of data (Zhang, Yao, Hu, Zhao, Li & Hu 2019). Figure 3.2

presents the architecture of the DAE module used in our work for short-term traffic flow prediction in linear roadways. It consists of an encoder for encoding the input matrix $\mathbf{A_t}$ into a context and a decoder for decoding the encoded context to the learned features $\bar{\mathbf{A}}_\mathbf{t}$. Note that a non-linear activation function is used in each layer for capturing non-linear relationships of traffic data.

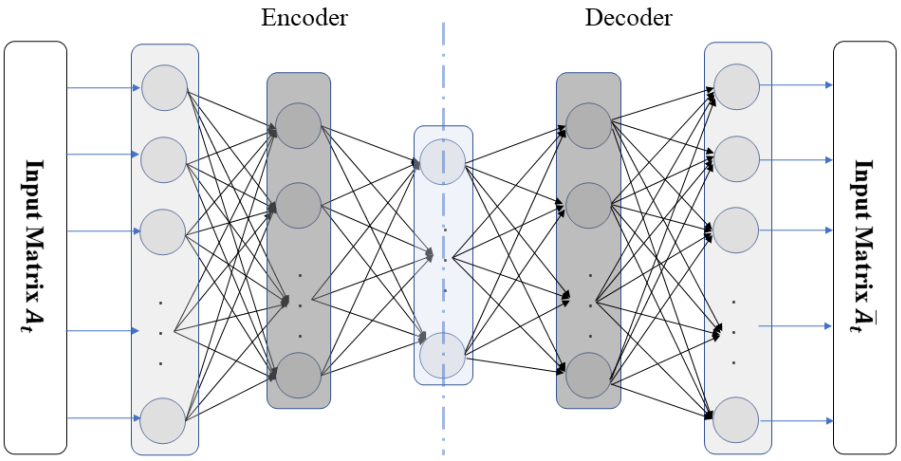


Figure 3.2: The architecture of DAE module.

CNN Module

The explanation of CNN module can be found in Chapter 2.5.3.

3.3 Data Description

To evaluate our EM, we choose two roadways: one in California (cf. Figure 3.3) and the other in London (cf. Figure 3.4). We focus on the traffic in one direction of the roadways. Specifically, we use the relevant data of five observation points along each chosen road and the objective is to predict the short-term traffic flow at the third observation points (i.e., the middle of the five points). The two observation points before and after the targeted location for prediction are taken into account for their spatial relevance.

The traffic data in California (hereafter referred to as "California-data") is collected from the California Department of Transportation (Caltrans) (Caltrans n.d.). The traffic data is collected every 30 seconds from more than 45,119 detectors in over 16,000 traffic stations, and aggregated into 5-min interval each for every station. The California-data we use covers the weekdays of the first three months of 2017.

Meanwhile, the traffic data in London (hereafter referred to as "London-data") is collected from the Highways England (England n.d.). For London, the traffic data is aggregated into 15-min interval each for every station. The chosen road is one near Heathrow airport and known to be prone to traffic congestion. This is as opposed to the California roadway which has comparatively much lower traffic flow. The two chosen roadways thus are chosen to represent two different traffic characteristics. We use the traffic data collected in the first three months in 2018 for London-data. In both cases, we use the first two months of the data for training and the third month for testing.

We show sample raw data from both datasets in Figure 3.5. Visually, especially in Figure 3.5(a), we already can see a relationship between traffic flow and traffic speed in which they are inversely proportional to each other (i.e., when traffic volume is high, the traffic speed recorded drops). This corroborates with the finding in (Zhao et al. 2014) which also found a non-linear relationship between traffic flow and traffic speed.

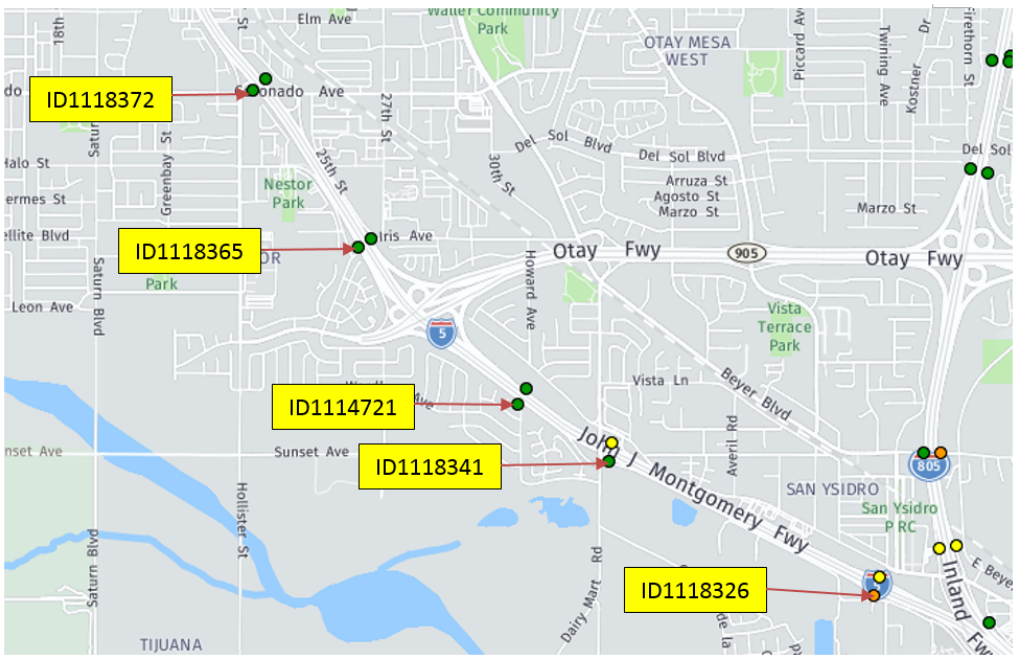


Figure 3.3: Five observation points for the chosen road in California. We predict the traffic flow at location ID1114721.

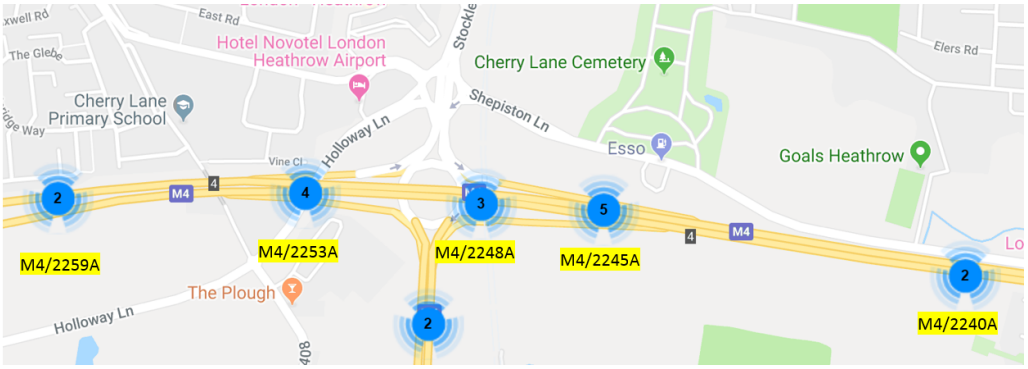


Figure 3.4: Five observation points for the chosen road in London. We predict the traffic flow at location MA/2248A.

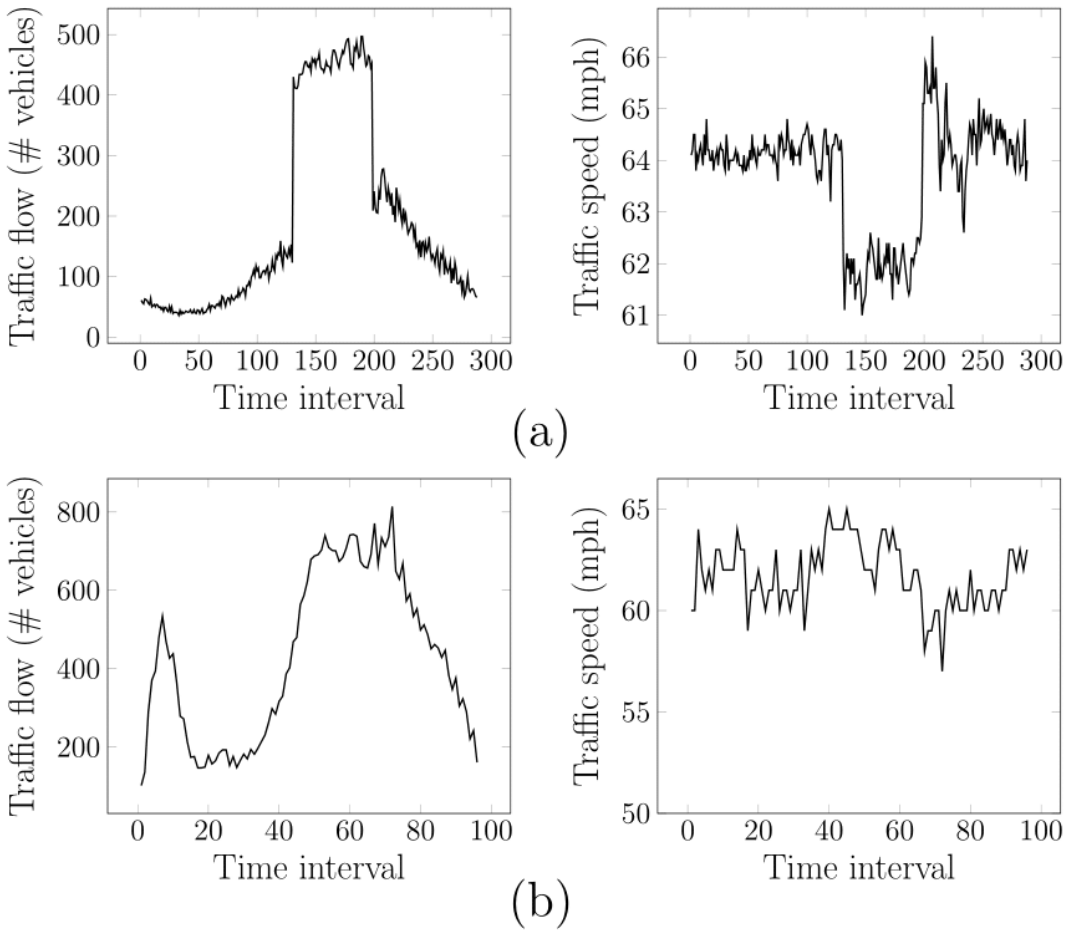


Figure 3.5: Raw traffic flow and traffic speed data in 24 hours for (a) California-data with 1 unit time interval = 5 mins and (b) London-data with 1 unit time interval = 15 mins.

3.4 Experiments

3.4.1 Model Setting

To achieve the best prediction accuracy of our EM, there are five parameters that we need to tune:

1. ***Number of data points,*** n***:*** Used in the MA filter to denoise. A suitable n can remove noise for providing accurate prediction, but large n can result in data distortion. For this parameter, we search the optimal value within the range 2 and 12 with unit step size.

2. ***Memory units:*** Used in the LSTM module to memorise historical traffic data. Too much memory units can result in over-fitting, which means the model obtains high accuracy on training data but low accuracy on testing data. For this parameter, we search the optimal value within the range 1 and 101 (step size = 5).

3. ***Number of layers:*** Used in the DAE and CNN modules. The number of layers in the DAE module means the number of hidden layers, and in the CNN module means the number of the convolutional layers and pooling layers. For both modules, more layers can capture more information from input, but it costs more time. We search the optimal number of layers between 1 and 6.

4. ***Batch size:*** Used in the EM model. It refers to the number of training samples utilised in one iteration. A large batch size requires less memory than small batch size. It also helps update the network's parameters more frequently. For this parameter, we search the optimal value within the range 32 and 1024 (step size = 32).

5. ***Epoch:*** Used in the EM model to update weights on the whole training data. Few epochs can result in under-fitting while the converse results in over-fitting. We search the optimal epoch size between 1 and 501 (step size = 50).

We follow the grid search method described in (Everaers & Kremer 1994) to explore and search for the optimal parameter combinations. In our case here, the number of possible parameter combinations in our model is (number of data $\times$ memory units $\times$ number of layers $\times$ batch size $\times$ epoch) which results in ($11 \times 21 \times 6 \times 6 \times 33 \times 11$) = 3,018,708 combinations.

3.4.2 Results and Discussion

To evaluate our proposed EM model, we compare our EM against the three individual constituent modules (i.e., LSTM, DAE and CNN separately in isolation) in our proposed EM model, two existing ensemble models, namely the Data Aggregation (DA) approach (Tan et al. 2009) and the CNN-LSTM traffic flow prediction (CLTFP) model (Wu & Tan 2016), and an integration of DA and CLTFP. In addition, three metrics mentioned in Chapter 2.7 including MAE, MAPE and RMSE are used for evaluating models.

We first compare the prediction of our EM against real traffic flow (raw data). Figure 3.6 presents the predicted traffic flow for the California-data and London-data over the period of one week from our proposed EM model and the real traffic flow. Overall, EM can efficiently predict the trend of the traffic flow changes, following the clear diurnal pattern of the roadways. However, on close inspection, we note that sudden and rapid fluctuations at short time interval present a much harder task for EM (e.g., the spikes at peak times).

Table 3.2 presents the results achieved by the considered models on the California-data and London-data. We also compute and present the (1-MAPE)% results in Figure 3.7.

Table 3.2: Comparison of all models for both linear roadways

Model Name	California-data			London-data		
	MAE	MAPE	RSME	MAE	MAPE	RSME
DAE	14.69	0.0973	31.66	89.68	0.1805	128.33
CNN	13.01	0.0880	24.52	62.20	0.1311	90.77
LSTM	11.44	0.0845	19.76	110.42	0.2968	167.87
DA	70.84	0.0460	80.66	348.04	0.1350	487.98
CLTFP	6.51	0.0436	10.75	41.13	0.0886	55.09
DA+CLTFP	5.23	0.0396	9.21	42.36	0.0763	43.02
EM	4.10	0.0255	16.86	27.59	0.0514	38.59

Comparing among the individual modules (i.e., LSTM, DAE and CNN), we found that LSTM performs the best for predicting the traffic in California though the three modules offers very similar accuracy (only 1.28% accuracy difference between the best and worst). However, when dealing with highly congested roadway (i.e., the London-data), LSTM performs significantly worse at 70.32% accuracy as opposed to 81.95% and 86.89% accuracy achieved by DAE and CNN respectively. This implies the importance of spatial information in short-term traffic prediction which LSTM does not consider. Another reason for this large difference in prediction accuracy may be because of large variance in the

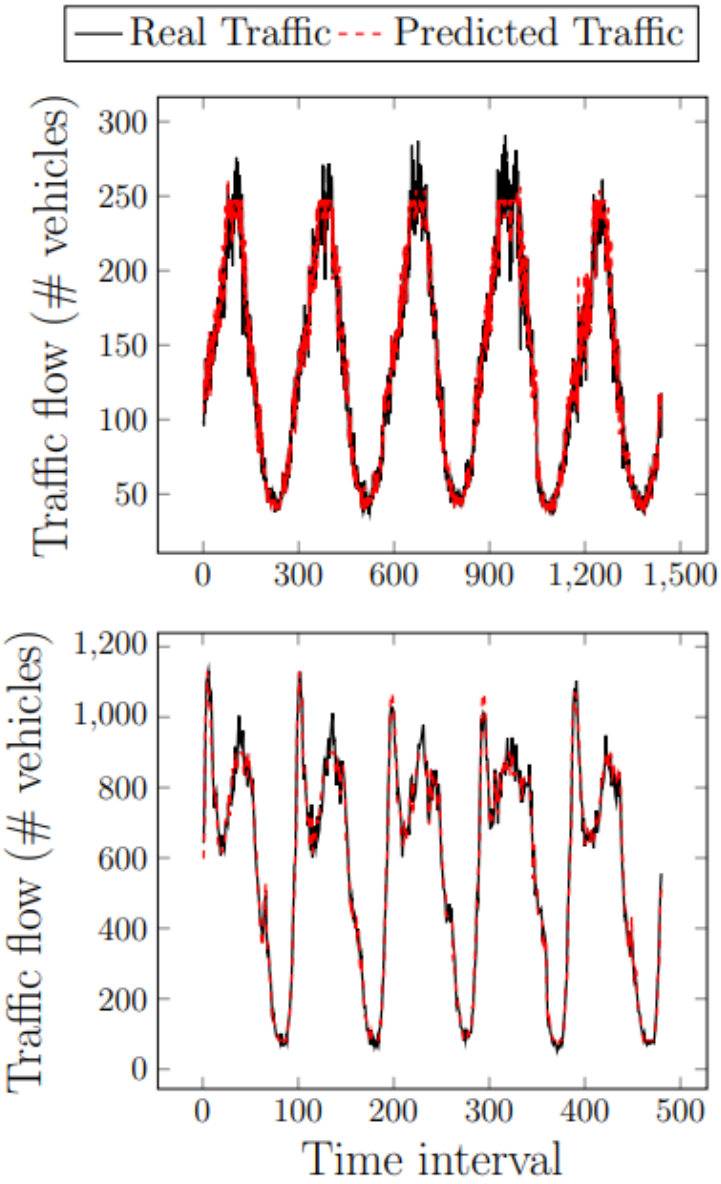


Figure 3.6: EM Prediction vs. real traffic (raw data) at the middle station over one week period: (top) California-data and (bottom) London-data.

London-data. For congestion prone roadway, CNN achieved the best prediction ($\approx 5\%$ better than DAE).

The DA, CLTFP and DA+CLTFP models exhibit better results than any of the single model above for both cases. Both of them achieved approximately 95.5% accuracy ($\approx 4.5\%$ better than the single models). This corroborates the past findings that ensemble approaches can produce better predictions than single models. Nevertheless, the absolute error of the DA model is higher than the CLTFP model. This may be due to the fact that the constituent modules of DA, which are MA, ES and ARIMA, can only capture

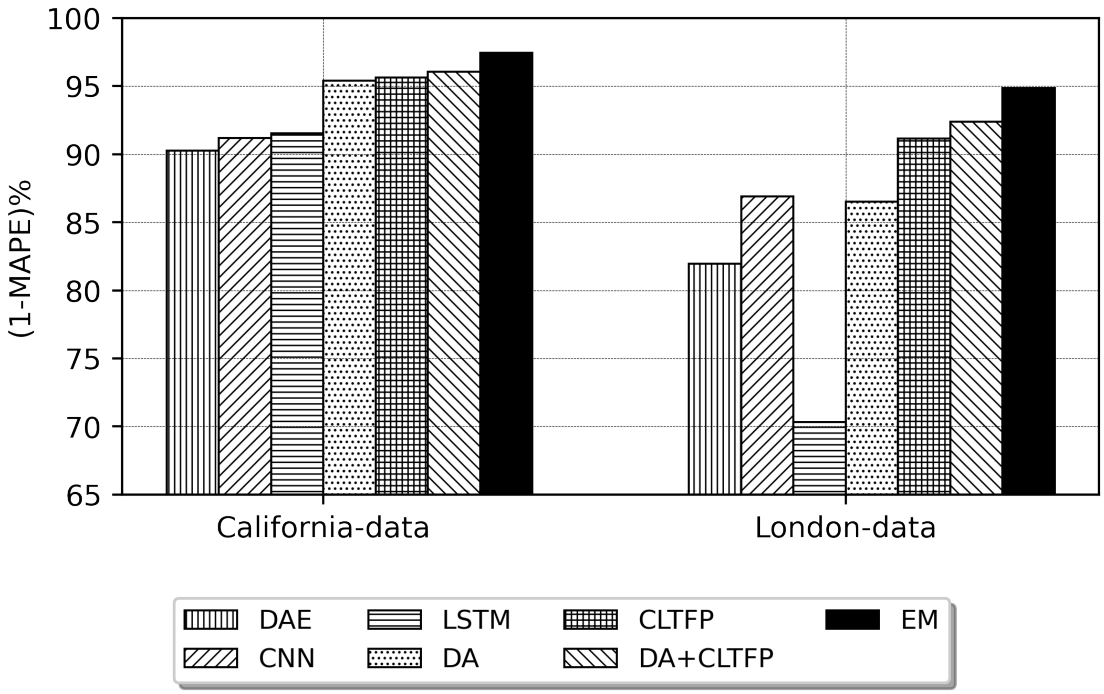


Figure 3.7: (1-MAPE)% on California-data (left) and London-data (right). Our EM model achieves the best accuracy for both cases.

temporal features of the data while on the other hand, CLTFP uses two LSTM modules for extracting temporal and periodic features and additionally, a CNN module for extracting spatial features. Taking the advantages of both DA and CLTFP, prediction accuracy of DA+CLTFP is higher than both DA and CLTFP.

Our EM model achieves the best accuracy amongst all the competing models for both smooth (California) and congestion prone (London) linear roadways. For California, EM achieves 97.45% accuracy (i.e., 7.18% and 1.41% better than the worst (DAE) and the next best competitor (DA+CLTFP) respectively). The performance improvement achieved by EM is even higher for busy roadway in London whereby EM obtained 94.86% accuracy (i.e., 24.54% and 2.49% better than the worst (LSTM) and the next best competitor (DA+CLTFP) respectively).

The two chosen linear roadways exhibit different traffic characteristics. The California roadway has comparatively lower traffic flow and thus less congestion prone while the London roadway is busier with higher traffic fluctuation. From our results, all models are better at predicting the traffic flow for California but perform significantly worse for London. The most affected model is LSTM when its prediction falls from 91.55% to 70.32% while DA+CLTFP's performance, the second best model, worsens by $\approx 3.67\%$. Our EM model on the hand is the least affected with only 2.59% accuracy degradation.

This suggests that our EM approach, while being consistently more accurate than others, it is also more robust against changes.

3.5 Chapter Summary

In this chapter, we address the problem of short-term traffic flow prediction problem on linear roadways. Owing to the various smart city initiatives, the problem has received renewed attention. Coupled with the increasing uptake of IoT technology and monitoring platforms offering better quality monitoring data, we develop a new ensemble model (EM) that predicts short-term traffic flow on linear roadways taking into account both temporal and spatial traffic information.

To this end, our EM proposal exploits three modules, including LSTM, DAE and CNN to make our predictions. We compare our proposal against its constituent modules, two existing ensemble models in the literature, namely DA and CLTFP, and an integration of those two existing ensemble models, DA+CLTFP, in predicting traffic at two linear roadways (located in California and London) exhibiting different characteristics (stable vs high variance).

Our EM achieved the highest accuracy among the seven considered models, computing 97.45% on California-data and 94.86% on London-data prediction accuracy. Our results also show that EM is not only accurate but also the most robust, recording the least accuracy degradation when making predictions for the more challenging London-data.

Chapter 4

Global- and Local-Spatial with Short-, Medium- and Long-Temporal Feature-based Short-Term Traffic Flow Prediction on Intersections

4.1 Introduction

This Chapter focuses on solving the short-term traffic flow prediction problem at intersections. The aim is to predict the number of vehicles in a targeted region over a short time interval. Compared to Chapter 3 where we solve short-term traffic flow prediction in linear roadways based on 1) better quality data (in volume and granularity), 2) combinations of prediction models and 3) joint consideration of temporal and spatial data, patterns or characteristics of traffic flow at intersections are more complex, especially for major intersections with multiple entrances and exits. Therefore, more detailed information and features should be considered for providing high prediction accuracy. The considered features in this chapter include 1) short-, 2) medium-, and 3) long-term temporal features as well as 4) global- and 5) local-spatial features. Taking these features into account, we develop a novel model based on both memory-based and memory-less approaches, named ALLSCP, for short-term traffic flow prediction at intersections.

The contributions for this chapter are as follows:

- Our ALLSCP model extracts short-, medium- and long-term temporal as well as

global and local spatial features to compute the final prediction.

- We consider real-world traffic flows at road segments with the intersection at two locations (i.e., Los Angeles and London).
- The literature has found that traffic flow is also correlated with vehicle speed (e.g., (Zhao et al. 2014)). In our work, both metrics are taken as input to our model.
- We conduct comparative study across other well-known existing machine/deep learning models and other hybrid ones against our model and found that ALLSCP performs better than the rest and also robust in different scenarios.

The rest of this chapter is organised as follows. We define the problem solved in our approach and detail the temporal-spatial input generation, our novel ALLSCP model, its architecture and constituent modules in Section 4.2. We compare the performance of our proposed model against existing models in Section 4.4 using real traffic data described in Section 4.3 before concluding our work in Section 4.5.

4.2 Methodology

For solving the short-term traffic flow prediction at intersections and providing high prediction accuracy, we firstly define the problem, and then design the input matrices based on the characteristics of traffic flow and speed data and the framework based on the advantages of individual modules. The following sections explain the problem definition (cf. Section 4.2.1), the input matrix generation (cf. Section 4.2.2) and our proposed ALLSCP model (cf. Section 4.2.3) in detail.

4.2.1 Problem Definition

Considering a road segment of interest with m observation stations located at different points, each station continuously monitors the traffic flow (i.e., vehicle count) and the average vehicle speed at a fixed time interval. Let traffic flow and average traffic speed of the i^{th} station at time interval t be x_t^i and s_t^i respectively. Then the sequences for the traffic flow and speed can be written as $\mathrm{x^i} = \{x_1^i, x_2^i, \ldots, x_t^i, \ldots, x_T^i\}$ and $\mathrm{s^i} = \{s_1^i, s_2^i, \ldots, s_t^i, \ldots, s_T^i\}$ where $t = 1, 2, 3, \ldots, T$ and $i = 1, 2, 3, \ldots, m$. We

consider both traffic flow and speed as input since we found in our datasets a linear relationship between the two (i.e., when traffic volume is high, traffic speed recorded drops) which corroborates the findings in (Van Aerde & Rakha 1995, Zhao et al. 2014).

Generally, traffic flow and speed data are collected from sensors installed on the roadsides with k minutes as a time interval (e.g., 5 minutes in (Caltrans n.d.) and 15 minutes in (England n.d.)). Here, we use $k = 15$ minutes whereby we manually integrate three intervals of traffic flow and speed from (Caltrans n.d.) into the same time interval as in (England n.d.) to ensure the uniformity of experiments.

Assume that station i^* is the selected station. Then, given previous measured traffic flow and speed at i^* and its neighbouring stations at the t^{th} time interval, the aim is to predict future traffic flow, $\hat{x}^{i^*}_{t+\Delta}$, at station, i^*, for the $(t+\Delta)^{th}$ time interval where Δ is the prediction horizon which is typically equal to 1, 2, 3, or 4 time intervals (Lv et al. 2015).

Due to the problem solved at intersections, we consider consecutive stations along road segments that meet or cross at the junction. Details of experiment are given in Section 4.3.

4.2.2 Temporal-Spatial Input Matrix Generation

Three types of temporal features (short-, medium- and long-term temporal features) and two spatial features (global and local spatial features) are extracted from real world traffic data. In (Tan et al. 2009), it is indicated that the current traffic flow is not only related to the traffic flow in the several previous time intervals but also related to traffic conditions at the same time in previous days and even weeks. Thus, we define: (1) traffic data in the several previous time intervals as short-term temporal features, (2) traffic data at the same time interval in the previous days as medium-term temporal features and (3) traffic data at the same time interval in previous weeks as long-term temporal features. Correspondingly, three temporal vectors, the time-interval temporal vector $\mathrm{T}^{\mathrm{i}}_{\mathrm{t}}$, the daily temporal vector $\mathrm{D}^{\mathrm{i}}_{\mathrm{t}}$ and the weekly temporal vector $\mathrm{W}^{\mathrm{i}}_{\mathrm{t}}$, are generated from original traffic flow data as follows: Eq. (4.1), Eq. (4.2), and Eq. (4.3) respectively.

$$\mathrm{T}^{\mathrm{i}}_{\mathrm{t}} = \{x^i_{t-(k_1-1)}, x^i_{t-(k_1-2)}, \ldots, x^i_{t-2}, x^i_{t-1}, x^i_t\} \tag{4.1}$$

$$\mathrm{D}^{\mathrm{i}}_{\mathrm{t}} = \{x^i_{(t+\Delta)-\frac{24\times60}{k}\times k_2}, x^i_{(t+\Delta)-\frac{24\times60}{k}\times(k_2-1)}, \ldots, x^i_{(t+\Delta)-\frac{24\times60}{k}\times 2}, x^i_{(t+\Delta)-\frac{24\times60}{k}\times 1}\} \tag{4.2}$$

$$\mathrm{W_t^i} = \{x^i_{(t+\Delta)-\frac{7\times24\times60}{k}\times k_3}, x^i_{(t+\Delta)-\frac{7\times24\times60}{k}\times(k_3-1)}, \ldots, x^i_{(t+\Delta)-\frac{7\times24\times60}{k}\times 2}, x^i_{(t+\Delta)-\frac{7\times24\times60}{k}\times 1}\} \tag{4.3}$$

Traffic measurements obtained from neighbouring stations could also be used to improve the prediction accuracy (Lv et al. 2014). Thus, we define the difference of traffic data between the targeted station and all its neighbouring stations as global spatial features. Furthermore, we define the difference of traffic data between adjacent stations in the neighbourhood as local-spatial features. As such, to capture spatial-temporal traffic relationships in the transportation network, we generate Eq. (4.4) in which m is the number of all stations and n is the number of previous time intervals before the $(t+\Delta)^{th}$. In $\mathbf{TS_t}$, besides x_t^i and s_t^i, we also include the traffic flow and speed changes as additional features. Specifically, we follow (Blandin et al. 2012) and define the traffic flow and speed changes at the i^{th} station between time interval t and $(t-1)$ as $\delta_t^{ix} = x_t^i - x_{t-1}^i$ and $\delta_t^{is} = s_t^i - s_{t-1}^i$ respectively. In this matrix, the information of the traffic flow, traffic speed, traffic flow difference and traffic speed difference in rows along the time dimension as temporal features, and the same information in columns along the space dimension corresponding to m neighbouring stations is regarded as spatial features.

4.2.3 The ALLSCP Model

Our novel model, named ALLSCP, exploits the strengths of several modules in capturing specific types of temporal and spatial features that contribute to the final prediction. Figure 4.1 shows the architecture of ALLSCP. Vectors $\mathrm{T_t^i}$, $\mathrm{D_t^i}$, $\mathrm{W_t^i}$ and Matrix $\mathbf{TS_t}$ are utilised for short-, medium-, long-term temporal as well as global and local spatial feature extraction by specific modules as follows:

Short-term Temporal Feature Extraction

In our model, $\mathrm{T_t^i}$ is used to extract short-term temporal features. We exploit a memory-based model, ARIMA, for this purpose. The key idea is that while ARIMA does not consider the non-linearity of traffic data and inherently assumes that traffic data to be linearly correlated with time, changes of traffic status over very short duration are continuous and can be considered as linear. As such, we use ARIMA model to analyse short-term

$$\mathbf{TS_t} = \begin{bmatrix} x^1_{t-(n-1)} & \cdots & x^1_t & s^1_{t-(n-1)} & \cdots & s^1_t & \delta^{1x}_{t-(n-2)} & \cdots & \delta^{1x}_t \\ \delta^{1s}_{t-(n-2)} & \cdots & \delta^{1s}_t & & & & & & \\ x^2_{t-(n-1)} & \cdots & x^2_t & s^1_{t-(n-1)} & \cdots & s^2_t & \delta^{2x}_{t-(n-2)} & \cdots & \delta^{2x}_t \\ \delta^{2s}_{t-(n-2)} & \cdots & \delta^{2s}_t & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & & & & & & \\ x^i_{t-(n-1)} & \cdots & x^i_t & s^i_{t-(n-1)} & \cdots & s^i_t & \delta^{ix}_{t-(n-2)} & \cdots & \delta^{ix}_t \\ \delta^{is}_{t-(n-2)} & \cdots & \delta^{is}_t & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & & & & & & \\ x^m_{t-(n-1)} & \cdots & x^m_t & s^m_{t-(n-1)} & \cdots & s^m_t & \delta^{mx}_{t-(n-2)} & \cdots & \delta^{mx}_t \\ \delta^{ms}_{t-(n-2)} & \cdots & \delta^{ms}_t & & & & & & \end{bmatrix} \tag{4.4}$$

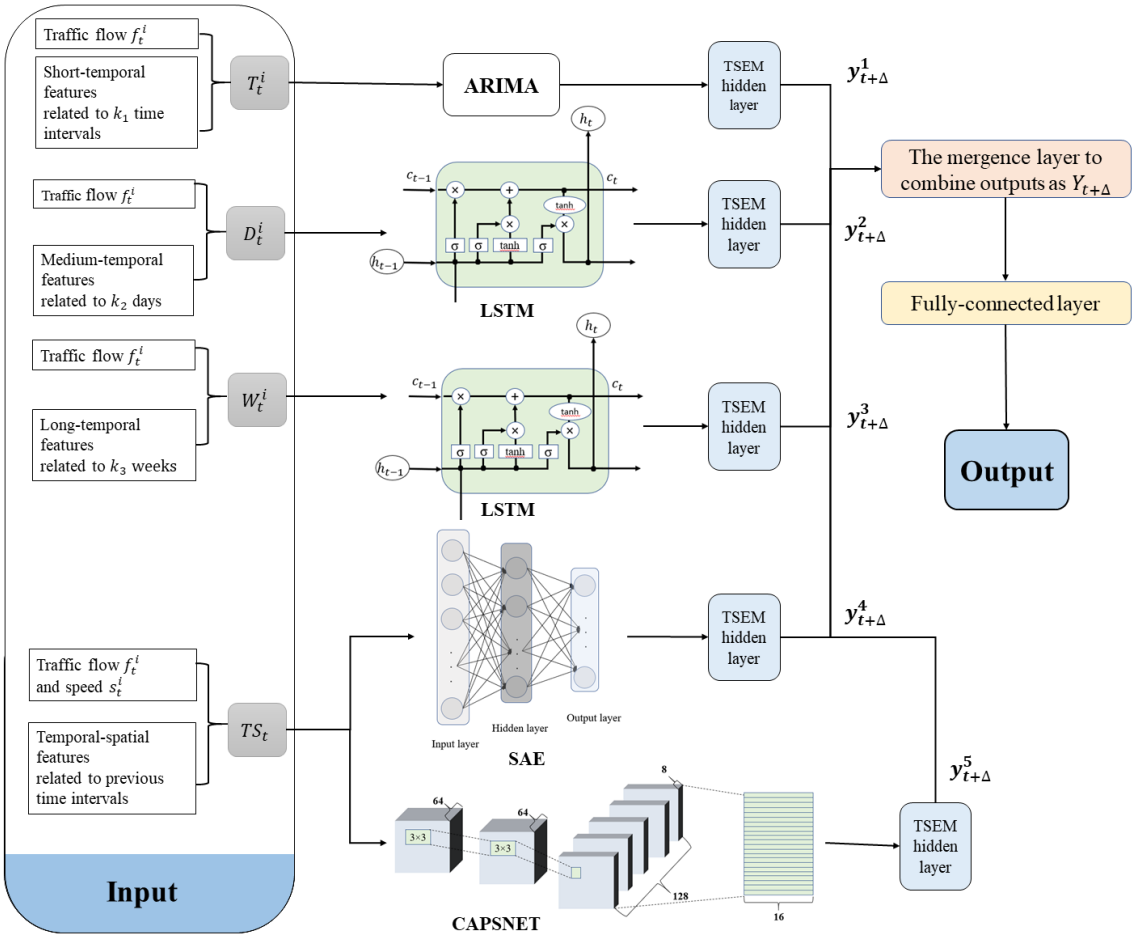


Figure 4.1: The architecture of the ALLSCP model.

temporal features and for longer temporal features, we propose to use LSTM (cf. Section 4.2.3). ARIMA is widely used as a statistical model for the prediction of time series data. It requires time series data to be stationary or stationary after differentiating (Jensen

1990). Traffic flow is a periodic time series and usually has slight changes in a very short time interval. Thus, traffic flow is considered as stationary time series and this is also why many existing works used ARIMA for short-term traffic flow prediction (e.g., (Ahmed & Cook 1979, Van Der Voort et al. 1996, Lee & Fambro 1999) and (Williams & Hoel 2003)). There are three important parameters: (1) p – the number of autoregressive terms, (2) d – the number of non-seasonal differences for converting data to be stationary and (3) q – the number of lagged forecast errors in the prediction equation. Firstly, we ensure our input data fulfil the stationary property by differentiating the input d times. Then short-term temporal features are extracted from previous p time intervals with the number of lagged prediction errors q. Next, short-term temporal features are extracted from the vector $\mathrm{T}_{\mathrm{t}}^{\mathrm{i}}$ in the ARIMA module using Eq. (4.5).

$$x_{t+\Delta}^{i} = c + \phi_1 x_t^i + \phi_2 x_{t-1}^i + \cdots + \phi_{p-1} x_{t-(k_1-2)}^i + \phi_p x_{t-(k_1-1)}^i \tag{4.5}$$

where ϕ_p is the parameter of the autoregressive part of ARIMA, and c is a constant.

As a part of input for the final prediction, the output from ARIMA is formatted to be in the same dimension with other features via the ensuing TSEM hidden layer (see Figure 4.1).

As discussed Chapter 2, ARIMA does not capture non-linear relationship. For this, we use the following modules to capture the non-linear relationship in traffic flow.

Medium- and Long-term Temporal Feature Extraction

We use a memory-based model, LSTM, which is capable of learning long-term relationship from historical data (Ma et al. 2015), to extract medium- (daily) and long-term temporal (weekly) features from $\mathrm{D}_{\mathrm{t}}^{\mathrm{i}}$ and $\mathrm{W}_{\mathrm{t}}^{\mathrm{i}}$, respectively. LSTM is an extension of the RNN (Hochreiter & Schmidhuber 1997). Compared to RNN that has only one part (i.e., the $tanh$ layer). LSTM consists of four parts: three gates (namely, input gate I_{t}, output gate O_{t} and forget gate F_{t}) and a cell state (C_{t}).

Taking $\mathrm{D}_{\mathrm{t}}^{\mathrm{i}}$ as an example on how the LSTM module extracts medium-temporal features in our framework, the forget gate F_{t} with a sigmoid layer, σ_g, firstly determines the part of information in current traffic flow x_t^i and in the last hidden state, H_{t-1} that it needs to forget and update it to the cell state, C_t, via Eq. (4.6). To supplement the forgotten information by forget gate F_t, the input gate, I_t, with a sigmoid layer σ_g is used to decide the information from current traffic flow x_t^i to be added into the cell state C_t via Eq.

(4.7). Then, the cell state, C_t, is updated using the tangent layer σ_C (cf. Eq. (4.9),) for integrating the traffic flow information provided from the forget gate, the input gate and the last cell state C_{t-1}. Meanwhile, the output gate with a sigmoid layer σ_g selects previous information remembered by H_{t-1} and the current information x_t^i by Eq. (4.8) for contributing to the final output. Finally, the predicted result is computed by combining remembered information from the output gate O_t and the cell state C_t with a tangent layer σ_H by Eq. (4.10).

$$F_t = \sigma_g(W_F \times x_t^i + U_F \times H_{t-1} + b_F) \tag{4.6}$$

$$I_t = \sigma_g(W_I \times x_t^i + U_I \times H_{t-1} + b_I) \tag{4.7}$$

$$O_t = \sigma_g(W_O \times x_t^i + U_O \times H_{t-1} + b_O) \tag{4.8}$$

$$C_t = F_t * C_{t-1} + I_t * \sigma_C(W_C \times x_t^i + U_C \times H_{t-1} + b_C) \tag{4.9}$$

$$H_t = O_t * \sigma_H \times (C_t). \tag{4.10}$$

where W_F, W_I, W_O and W_C are the weights of the forget gate, the input gate, the output gate and the cell state respectively while b_F, b_I, b_O and b_C are the corresponding bias for each gate and state. Furthermore, U_F, U_I, U_O and U_C are the weights of the last hidden state H_{t-1}. We use σ_g to denote a sigmoid function (= $\frac{1}{1+e^{-x}}$) in three gates and the operator $*$ to denote Hadamard product. σ_C and σ_H are hyperbolic tangent functions ($tanh(x)$) for the cell state and the final output. In our case, one LSTM module, for extracting medium-temporal features from $\mathrm{D_t^i}$, has k_2 memory units and another LSTM module, for extracting long-temporal features from $\mathrm{W_t^i}$, has k_3 memory units. This means we use traffic flow in k_2 time intervals in previous days and in k_3 time intervals in previous weeks to extract medium- and long-temporal features.

Global-Spatial Feature Extraction

The input matrix $\mathbf{TS_t}$ that records historical traffic flow at the targeted station and their neighbouring stations is used to extract global spatial features by utilising the Stacked Autoencoder (SAE) considered as a memory-less module in our framework as shown in Figure 4.1. The SAE neural network (Singh & Mohan 2018), as an unsupervised learning algorithm, can learn features from high dimension data and then encode it into low dimension data. If a predictor (e.g., a logistic regression layer (Menard 2002) or a softmax layer (Dunne & Campbell 1997)) is added on the top of SAE model, it can be used for prediction problems. The SAE module in our framework includes one input layer and n_s hidden layers for global spatial feature extraction. It first takes the input matrix $\mathbf{TS_t}$ into the SAE module via the input layer. Then, it encodes the output of the input layer to the 1^{st} hidden layer representation $\mathrm{y}^1(\mathrm{TS_t})$ via Eq. (4.11) and finally it decodes the representation $\mathrm{y}^1(\mathrm{TS_t})$ back into a reconstruction $\mathrm{z}^1(\mathrm{TS_t})$ calculated via Eq. (4.12). The shape of the input matrix $\mathbf{TS_t}$ is $(4n-2)\times m$, i.e., the input layer has $(4n-2)\times m$ neural units without any weighted inputs. The number of neural units in the hidden layers is set as n_u that is decided by the grid search detailed in (Everaers & Kremer 1994) from a limited range. We consider logistic sigmoid function for each hidden layer for extracting global spatial features from the input matrix $\mathbf{TS_t}$ by the fully connection between layers.

$$\mathrm{y}^1(\mathrm{TS_t}) = G\Big(\mathrm{w}_1 \times \mathbf{TS_t} + b_1\Big) \tag{4.11}$$

$$\mathrm{z}^1(\mathrm{TS_t}) = Z\Big(\mathrm{w}_2 \times \mathbf{y}(\mathbf{TS_t}) + b_2\Big) \tag{4.12}$$

where G is the logistic sigmoid function as an encoder, w_1 is the weight vector of the encoder, and b_1 is the bias vector. Correspondingly, Z is the logistic sigmoid function as a decoder, and w_2 and b_2 are respectively the weight vector and the related bias of the decoder.

Local-Spatial Feature Extraction

We extract local-spatial features via a memory-less module, CAPSNET. Compared to SAE that extracts global-spatial features via full connection between layers, CAPSNET focuses on local-spatial features via the local connections implemented by the kernel functions in convolutional layers. For example, if the kernel size was set as 3×3, convolu-

tional value only includes features of three continuous points inside the kernel. This can be considered as the local feature. Capsule network (Sabour et al. 2017) is based on CNN. CAPSNET is characterised by "capsules" in vector form. When extracting local features in images, important local information that the capsules detect is encapsulated in a vector form. The length of an output vector encodes the probability of a feature and the direction of the vector encodes the gesture of features, such as rotation angle and direction. Compared to CNN, CAPSNET can effectively extract more detailed local-spatial features because of the vector form. Here, the input matrix $\mathbf{TS_t}$ is regarded as a image matrix in our CAPSNET module, and its shape is $(4n-2) \times m$.

Our CAPSNET module consists of two convolutional layers and a fully connected layer, TrafficCaps. The first convolutional layer is the same as the convolutional layer in conventional CNN layers. It is used to extract spatial features of traffic flow between neighbouring stations. The $ReLU$ function (cf. Eq. (4.13)) is used as the activation function in this layer. The second convolutional layer is the primary capsule layer to capture the local-spatial features and used for converting the single scalar output of the first convolutional layer into vector form with a dimension of 8 by "capsules". Finally, the TrafficCaps layer extracts the spatial relationship between the local-spatial features obtained from primary capsules and outputs the features to a set of advanced capsules with a dimension of 16. Eq. (4.14) is the novel nonlinear "squashing" activation function for the vector form of capsules used in the primary convolutional and TrafficCaps layers.

$$ReLU(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \tag{4.13}$$

$$\mathrm{v_j} = \frac{||\mathrm{s_j}||^2}{1 + ||\mathrm{s_j}||^2} \frac{\mathrm{s_j}}{||\mathrm{s_j}||} \tag{4.14}$$

where $\mathrm{v_j}$ and $\mathrm{s_j}$ are the output and input vector of capsule j respectively. The final output of the module is a vector.

Final Prediction

After extracting short-, medium-, long-term temporal as well as global and local spatial features, we use a fully-connected layer as a predictor in our framework for the final short-term traffic flow prediction in intersections. In addition, on the top of the fully-connected layer, there are two hidden layers (namely TSEM hidden layer and Merging layer). The

function of the TSEM hidden layer is to convert outputs from five modules into the related tensors of the same dimension: $y^1_{t+\Delta}$ from the ARIMA module for short-term temporal features, $y^2_{t+\Delta}$ and $y^3_{t+\Delta}$ from the LSTM module for medium- (daily) and long-term (weekly) temporal features, $y^4_{t+\Delta}$ from the SAE module for global-spatial features and $y^5_{t+\Delta}$ from the CAPSNET module for local-spatial features. Then the following mergence layer concatenates the five outputs ($y^1_{t+\Delta}$, $y^2_{t+\Delta}$, $y^3_{t+\Delta}$, $y^4_{t+\Delta}$, $y^5_{t+\Delta}$) in the last dimension and generates temporal and spatial-fused features as Eq. (4.15). The fully-connected layer matches the temporal and spatial-fused features $Y_{t+\Delta}$ to the output.

$$Y_{t+\Delta} = \{y^1_{t+\Delta}, y^2_{t+\Delta}, y^3_{t+\Delta}, y^4_{t+\Delta}, y^5_{t+\Delta}\}. \tag{4.15}$$

The loss function employed to assess our model is the Mean Squared Error (MSE) and the optimiser $Adam$ (Kingma & Ba 2014) is utilised to minimise the loss function MSE. The grid search method (Everaers & Kremer 1994) is used to find the optimal parameter combinations in the limited range detailed in Section 4.4.1. In addition, following the convention in the literature, we use 70% of each dataset for training, 20% for validation and 10% for testing.

4.3 Data Description

In our experiments, we use real-world traffic data collected from intersection road segments at two locations: Los Angeles, USA and London, United Kingdom. The Los Angeles traffic data is collected from the California Department of Transportation (Caltrans) (PeMS) (Caltrans n.d.). PeMS aggregates traffic data into 5-min interval for every station. Meanwhile, the London traffic data is collected from the Highways England (England n.d.) where the traffic data is aggregated into 15-min interval each for every station. For both, we use data period between January and June 2018. As prior mentioned, we use 15-min interval for both locations to ensure the uniformity of experiments. We choose roadways prone to heavy congestion and frequent incidents.

In addition, we also collect traffic data from two linear roadways in Los Angeles, USA and London, United Kingdom to show the generality and robustness of the performance of our novel ALLSCP model on different types of roadways.

4.3.1 Data Collection from Linear Roadways

For linear roadway in Los Angeles, traffic data is collected from roadway 605 shown in Figure 4.2 while, for London, traffic data is collected on the M4 motorway shown in Figure 4.3. Hereinafter, we label our two datasets from linear roadways as follows: `L-Los` in Los Angeles and `L-London` in London. Specifically, data from five observation stations are used to predict the traffic flow at the third observation station (i.e., the middle of the five stations). Figure 4.2 shows the five observation stations labelled as a_l, b_l, c_l, d_l and e_l, and we predict short-term traffic flow at station c_l. Thus, the number of observation stations inside of temporal-spatial matrix $\mathbf{TS_t}$, $m = 5$. The two observation stations before and after station c_l are taken into consideration for spatial features.

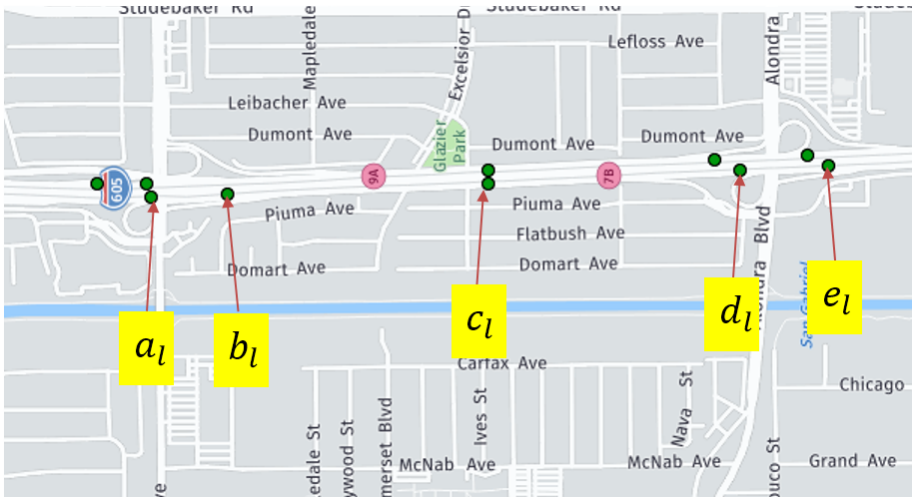


Figure 4.2: Five observation stations for the chosen road in Los Angeles. We predict the traffic flow at station c_l.

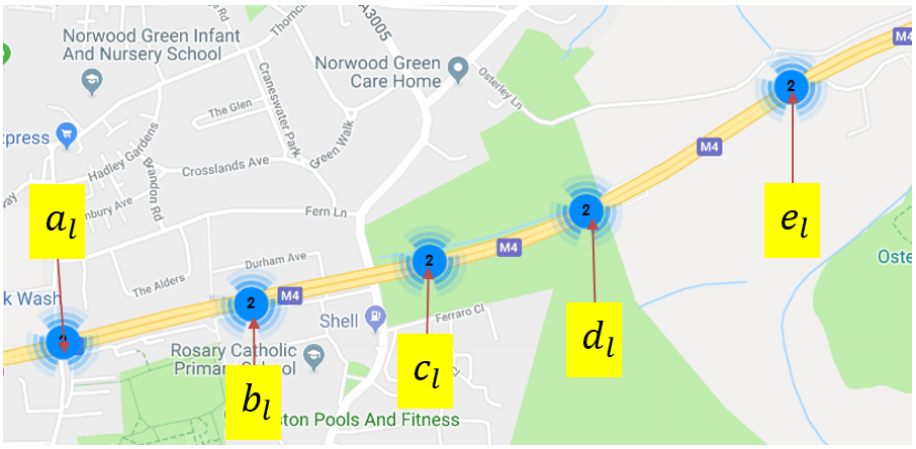


Figure 4.3: Five observation stations for the chosen road in London. We predict the traffic flow at station c_l.

Figure 4.4 and Figure 4.5 show the real traffic flow and speed data in a randomly selected week from L-Los and L-London, respectively. It is obvious that traffic flow and speed have daily period. On both datasets, traffic speed becomes lower when traffic flow is higher, which means the traffic flow data has very close relationship with the traffic speed data. Besides, figures show traffic flow is larger and traffic speed is lower from L-Los, compared to the data from L-London. Both traffic flow and speed have larger and more frequent fluctuations from L-Los, which indicates the traffic situations are worse in L-Los.

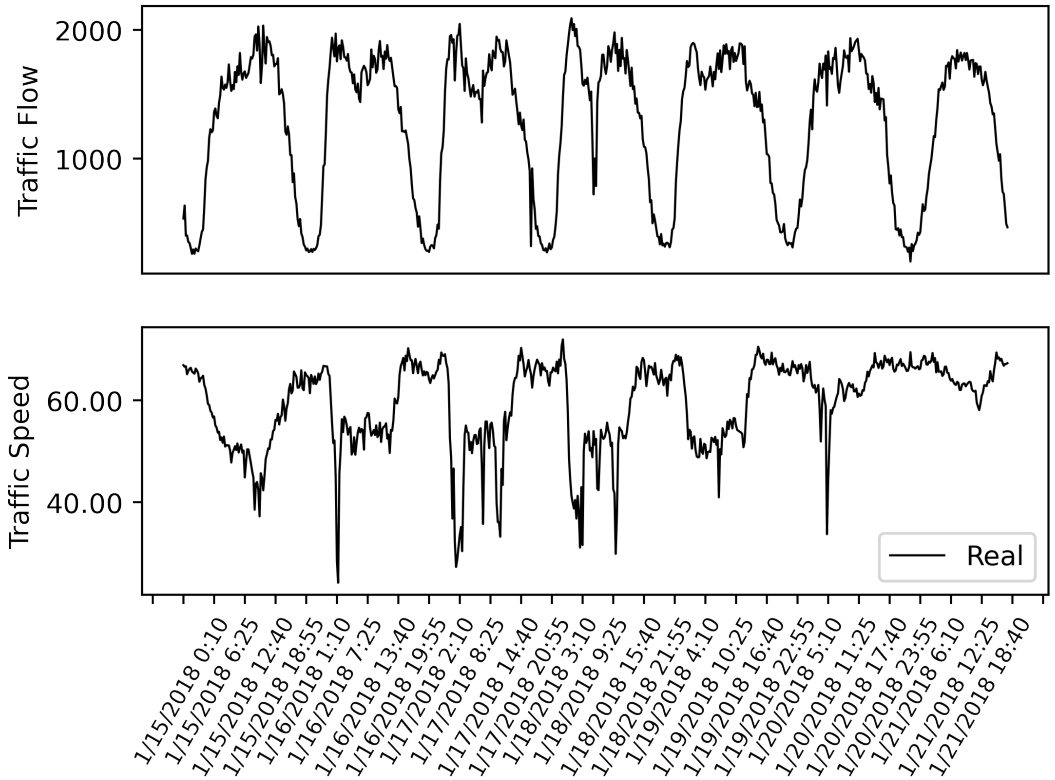


Figure 4.4: Real traffic flow and speed data in a randomly selected week from L-Los. The x-axis and y-axis represent time and traffic flow (top) and/or speed (bottom), respectively. The unit of traffic flow is the number of vehicles and the unit of traffic speed is "miles/hour".

4.3.2 Data Collection from Intersections

For intersections in Los Angeles, we focus on the junction between road 605 and 105. Eight contiguous observation stations are utilised to predict two targeted stations. Figure 4.6 shows the eight selected observation stations (labelled as a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i). We consider the traffic flow in the direction from a_i moving towards d_i, f_i and h_i.

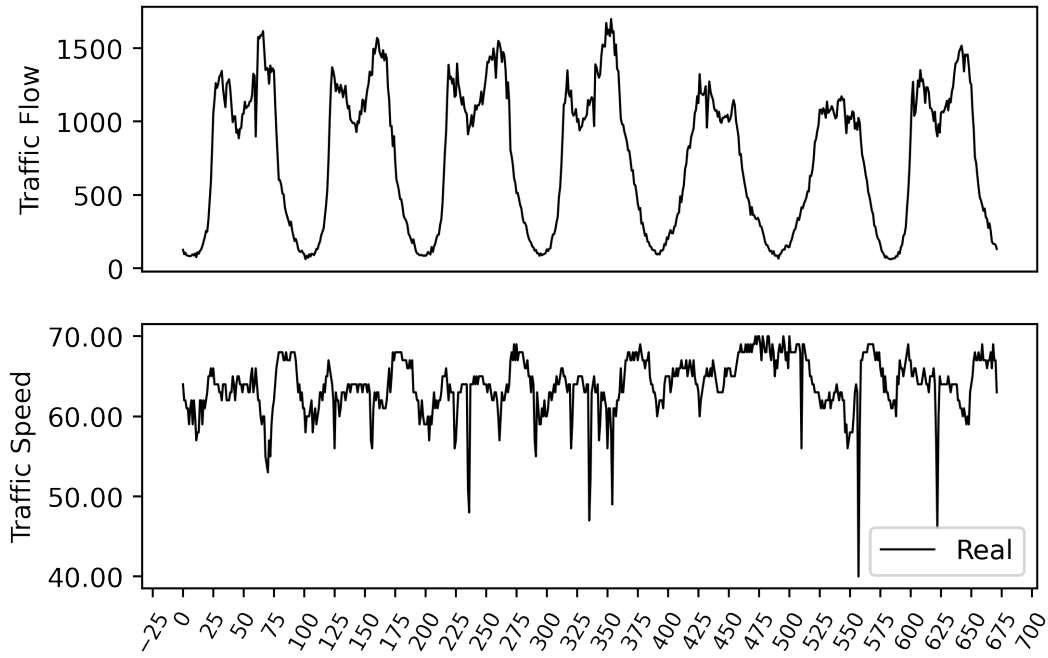


Figure 4.5: Real traffic flow and speed data in a randomly selected week from `L-London`. The x-axis and y-axis represent time interval and traffic flow (top) and/or speed (bottom), respectively. The unit of traffic flow is the number of vehicles and the unit of traffic speed is "miles/hour". A time interval is considered as 15 minutes and a week has $\frac{7\times24\times60}{15} = 672$ time intervals.

Our aim is to predict short-term traffic flow at e_i and g_i. The number of observation stations inside of temporal-spatial matrix $\mathbf{TS_t}$, m is 8. Compared to the previous case for linear roadway, the difference is that the data at the exit and entrance corresponding to the current direction is considered into prediction. This is important as drivers may avoid congested road segments by exiting at the junction and return back further down the road. For example, in Figure 4.6, if b_i and/or g_i is congested, drivers approaching h_i from a_i can choose to exit to c_i or e_i rather than going directly from a_i to h_i. Correspondingly, similar intersection scenario is considered for London, and we collect the data at the junction between M4 and M25 displayed in Figure 4.7. Hereinafter, we label our two datasets from intersections as follows: `I-Los` in Los Angeles and `I-London` in London.

Figure 4.8 and Figure 4.9 show the real traffic flow and speed data in a randomly selected week from `I-Los` and `I-London`, respectively. For intersections, the patterns of traffic data can be found similar to these for linear roadways in Chapter 3.3. For example, when the traffic flow is higher, the traffic speed is lower, and both traffic flow and speed have a daily pattern.

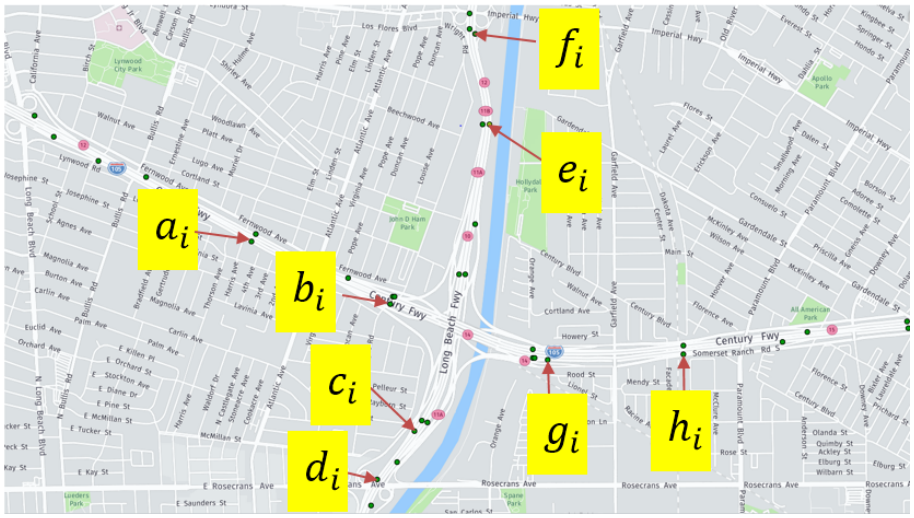


Figure 4.6: Eight observation stations for the chosen road in Los Angeles. We predict the traffic flow at station: e_i and g_i.

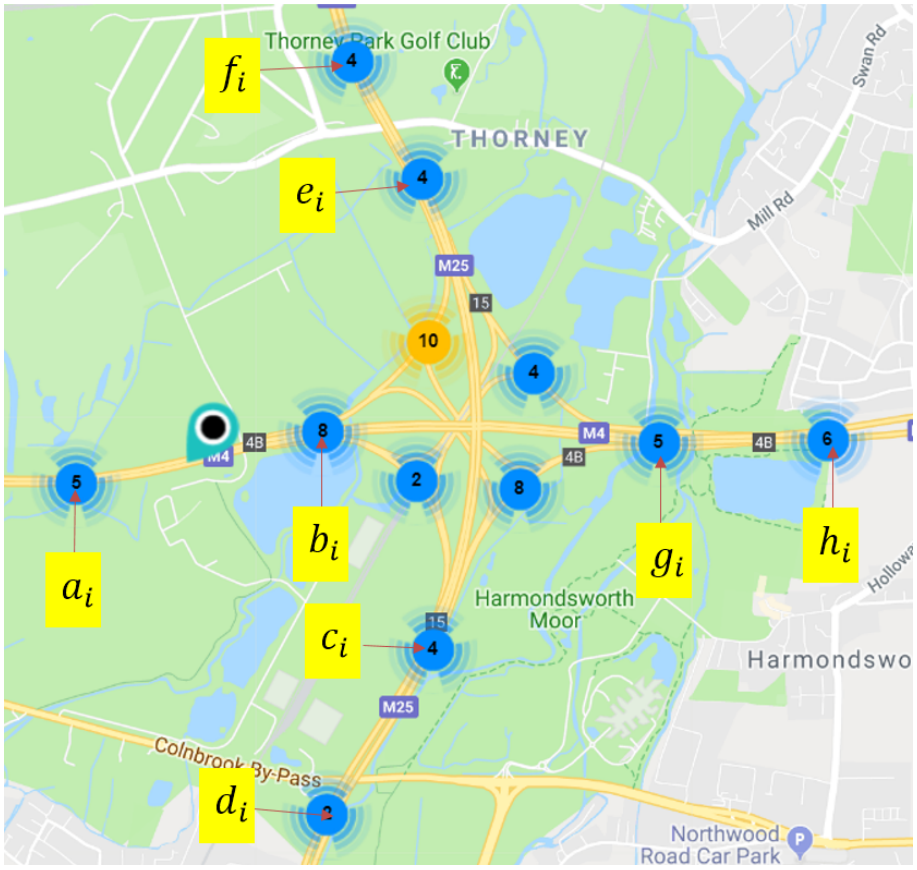


Figure 4.7: Eight observation stations for the chosen road in London. We predict the traffic flow at stations: e_i and g_i.

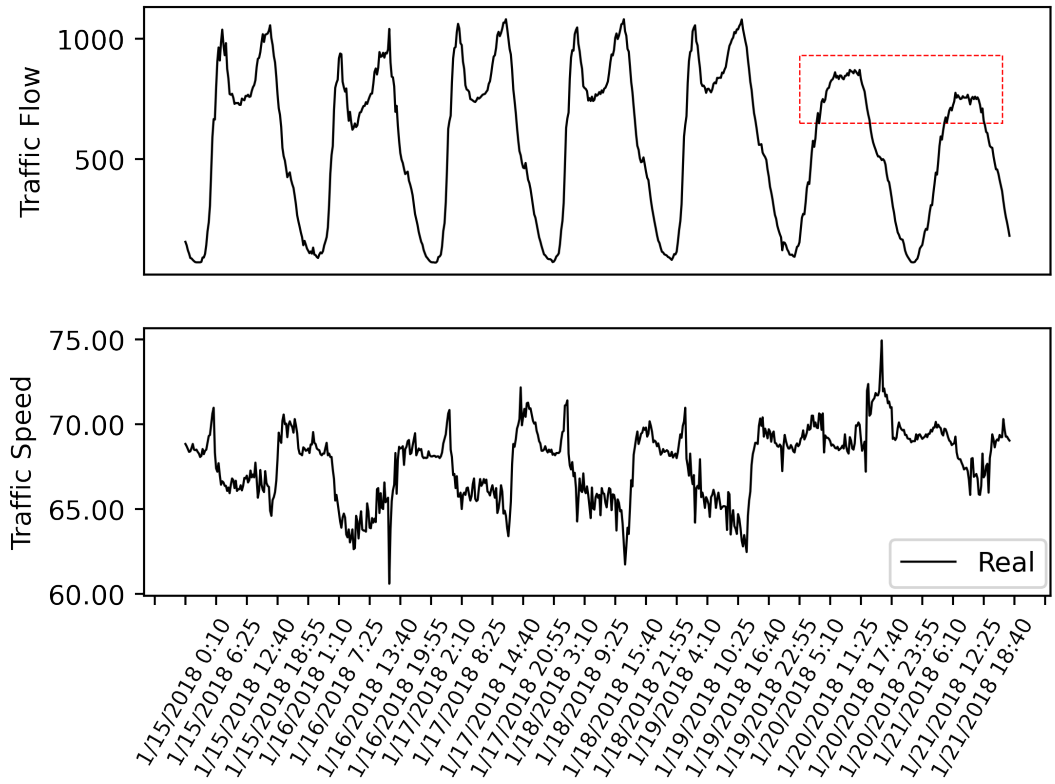


Figure 4.8: Real traffic flow and speed data in a random selected week from I-Los. The x-axis and y-axis represent time and traffic flow (top) and/or speed (bottom), respectively. The unit of traffic flow is the number of vehicles and the unit of traffic speed is "miles/hour". Traffic flow with red box is for the weekend.

4.3.3 Relations of Traffic Flow and Speed

Table 4.1 presents the characteristics of traffic flow and speed from four datasets, including maximum (Max), minimum (Min), mean (Mean), standard deviation (Std), variance (Var) and size (KB). The names of datasets with subscript f and s describe the traffic flow data and the traffic speed data from the related datasets, respectively. For intersections, based on the larger standard deviations and variances of traffic flow from I-Los, traffic fluctuations for I-Los is more complicated than I-London. However, in terms of traffic speed, I-Los, with smaller standard deviation and variance, appears to be less complicated than I-London. This phenomenon probably results from the different physical characteristics of roadways in the United States and the United Kingdom. For linear roadways, the standard deviations and variances of both traffic flow and speed from L-Los are larger than those from L-London, which indicate that L-Los is more complicated than L-London.

Figure 4.10 and Figure 4.11 present the relationships between traffic flow and speed in

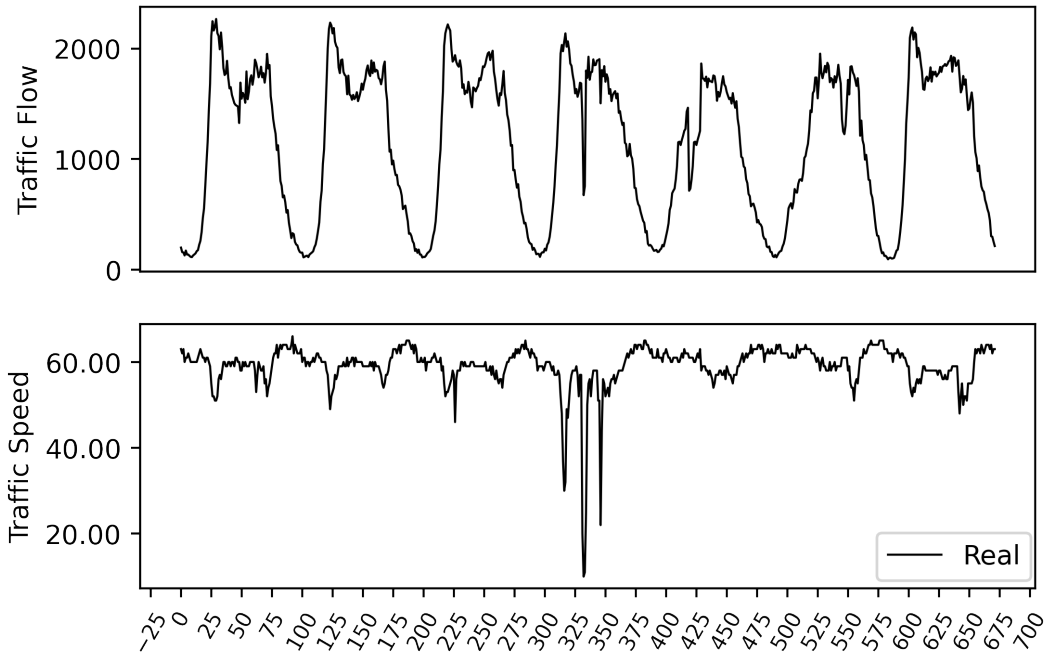


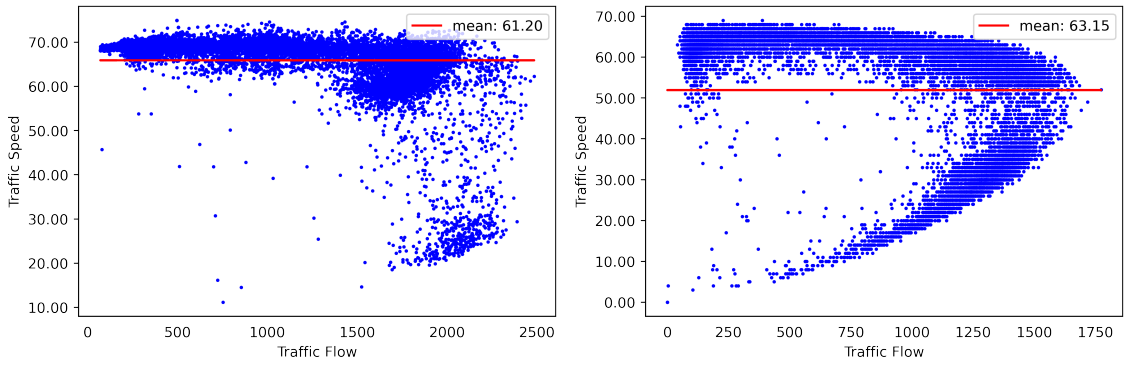
Figure 4.9: Real traffic flow and speed data in a randomly selected week from I-London. The x-axis and y-axis represent time interval and traffic flow (top) and/or speed (bottom), respectively. The unit of traffic flow is the number of vehicles and the unit of traffic speed is "miles/hour". A time interval is considered as 15 minutes and a week has $\frac{7\times24\times60}{15} = 672$ time intervals.

Table 4.1: Characteristics of four types of datasets

Dataset	Max	Min	Mean	Std	Var	Size (KB)
L-Los$_f$	2186.00	155.00	1318.74	555.54	308627.53	1358
L-London$_f$	1750.00	0.00	814.87	494.30	244331.83	1328
I-Los$_f$	2484.00	71.00	971.62	613.02	375788.54	2172
I-London$_f$	1773.00	0.00	863.27	503.25	253257.98	2124
L-Los$_s$	76.43	13.17	61.20	8.94	79.91	1358
L-London$_s$	77.00	0.00	63.15	6.25	39.06	1328
I-Los$_s$	74.93	11.17	65.89	7.03	49.48	2172
I-London$_s$	69.00	0.00	51.92	15.23	231.93	2124

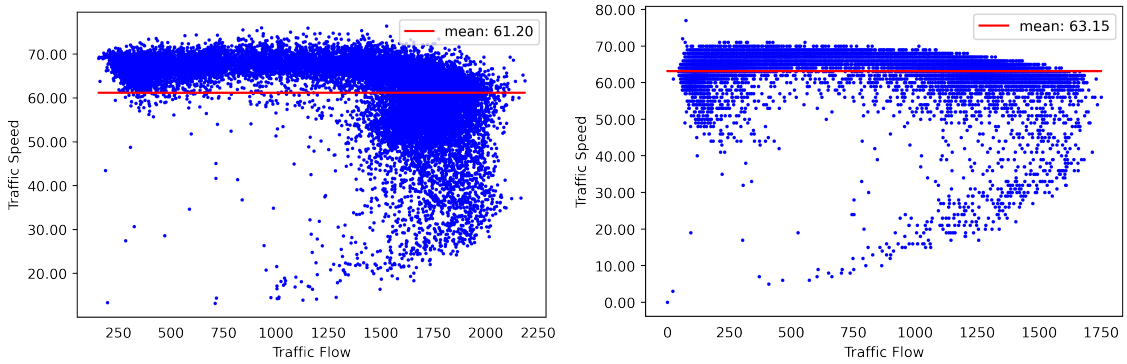
intersections and in linear roadways, respectively. The x-axis describes traffic flow and the y-axis represents traffic speed. It shows a non-linear relation exists between traffic flow and speed. In fundamental relations of traffic flow (Mathew & Rao 2006), the relationships of traffic flow and speed can be presumed that the traffic flow is zero either

because there is no vehicles or there are too many vehicles without mobility.This assumption fits the relationships of traffic flow and speed shown in Figure 4.10 and Figure 4.11. The traffic speed decreases while the traffic flow increases when the traffic speed is higher than the averaged speed. However, the traffic speed increases as the traffic flow increases when the traffic speed is lower than its average.



(a) I−Los (b) I−London

Figure 4.10: Relations of traffic flow and speed from I−Los (a) and I−London (b). The x-axis and the y-axis represent traffic flow and speed, respectively. The unit of traffic flow is the number of vehicles and the unit of traffic speed is "miles/hour". The red line represents the average of traffic speed.



(a) L−Los (b) L−London

Figure 4.11: Relations of traffic flow and speed from L−Los (a) and L−London (b). The x-axis and the y-axis represent traffic flow and speed, respectively. The unit of traffic flow is the number of vehicles and the unit of traffic speed is "miles/hour". The red line represents the average of traffic speed.

4.3.4 Data Preparation

To test our model from different dimension, processed datasets generated by Hodrick Prescott (HP) filter (Maravall et al. 2001) is also used to evaluate our model. HP filter is a type of data-smoothing technique to remove short-term fluctuations and reveal long-term trend. For using HP filter to denoise and smooth data, the traffic flow (or speed) in our work is decoded into two parts: a trend component ${\tau_{xt}}^i$ (or ${\tau_{st}}^i$) and a cyclical part ${c_{xt}}^i$ (or ${c_{st}}^i$), by Eq. (4.16) and Eq. (4.17). ${\tau_{xt}}^i$ and ${\tau_{st}}^i$ are the trend components of traffic flow and speed, and ${c_{xt}}^i$ and ${c_{st}}^i$ are the cyclical parts of traffic flow and speed. These trend components and cyclical parts are optimised by Eq. (4.18) and Eq. (4.19), and then used to calculate processed traffic flow ($\overline{x}_t^i$) and speed ($\overline{s}_t^i$) by Eq. (4.20) and Eq. (4.21).

$$x_t^i = {\tau_{xt}}^i + {c_{xt}}^i \tag{4.16}$$

$$s_t^i = {\tau_{st}}^i + {c_{st}}^i \tag{4.17}$$

$$\min_{\tau} \left(\sum_{t=1}^{T} \left(x_t^i - {\tau_{xt}}^i\right)^2 + \lambda \sum_{t=2}^{T-1} \left[\left({\tau_{xt+\Delta}}^i - \tau_{xt}\right) - \left(\tau_{xt} - \tau_{xt-1}\right) \right]^2 \right) \tag{4.18}$$

$$\min_{\tau} \left(\sum_{t=1}^{T} \left(s_t^i - {\tau_{st}}^i\right)^2 + \lambda \sum_{t=2}^{T-1} \left[\left({\tau_{st+\Delta}}^i - \tau_{st}\right) - \left(\tau_{st} - \tau_{st-1}\right) \right]^2 \right) \tag{4.19}$$

$$\overline{x}_t^i = {\tau_{xt}}^i + {c_{xt}}^i \tag{4.20}$$

$$\overline{s}_t^i = {\tau_{st}}^i + {c_{st}}^i \tag{4.21}$$

where λ is the signal-to-noise ratio. A reasonable λ can remove noise for providing accurate prediction, but too large or too small a λ value can result in data distortion. Hereafter, pre-processed (or de-noised) datasets corresponding to original datasets are labeled as `PI-Los`, `PI-London`, `PL-Los` and `PL-London`. Figure 4.12, Figure 4.13, Figure 4.14 and Figure 4.15 present the pre-processed traffic flow and speed for `I-Los`, `I-London`, `L-Los` and `L-London`, respectively. Compared to real data shown in Fig-

ure 4.8, Figure 4.9, Figure 4.4 and Figure 4.5, sharp changes resulted from noises or incidents are removed and data becomes more smoothing.

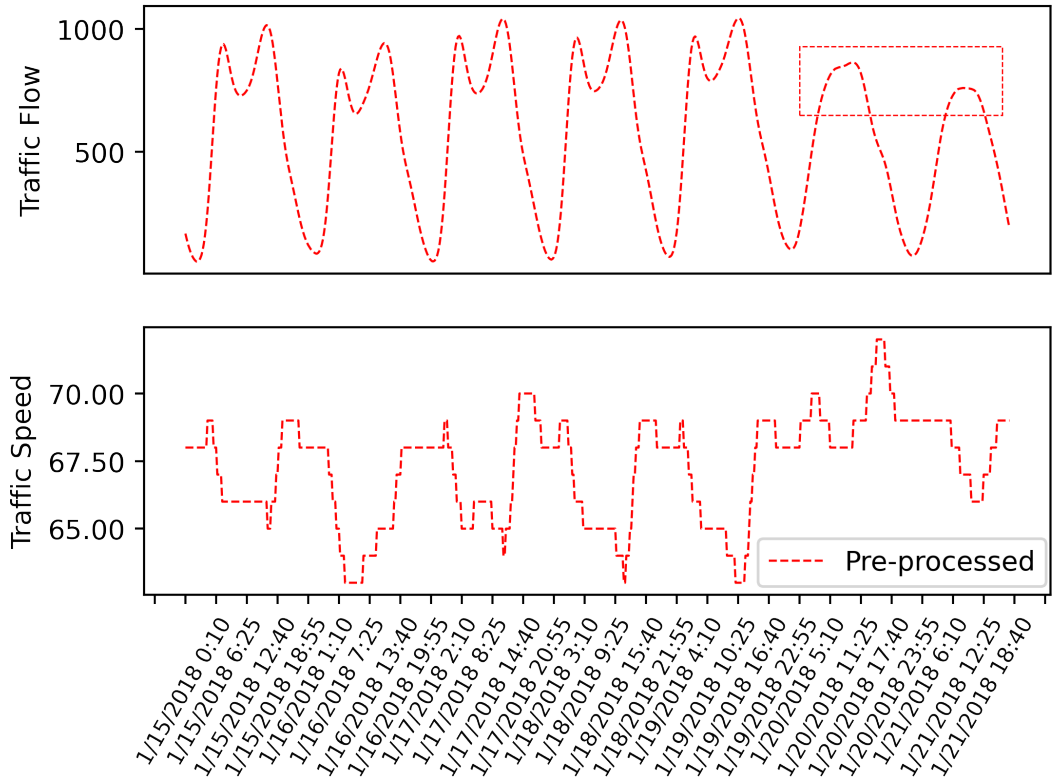


Figure 4.12: Pre-processed traffic flow and speed in a randomly selected week from `PI-Los`. The x-axis and y-axis represent time and traffic flow (top) and/or speed (bottom), respectively. The unit of traffic flow is the number of vehicles and the unit of traffic speed is "miles/hour". Traffic flow with red box is from the weekend.

4.4 Experiments

4.4.1 Model Setting

For achieving the best results, the ALLSCP parameters that we need to tune are as follows:

1. ***Parameters in the temporal-spatial input matrix:*** By analysing the autocorrelation function (ACF) and partial autocorrelation function (PACF) of traffic flow sequences, we set k_1 in $\mathrm{T}_{\mathrm{t}}^{\mathrm{i}}$ to 9 as we found traffic flow in the next time interval highly depends on traffic flow in 9 previous time intervals. Both k_2 in the daily temporal matrix $\mathrm{D}_{\mathrm{t}}^{\mathrm{i}}$ and k_3 in the weekly temporal matrix $\mathrm{W}_{\mathrm{t}}^{\mathrm{i}}$ are set to 7, taking into account

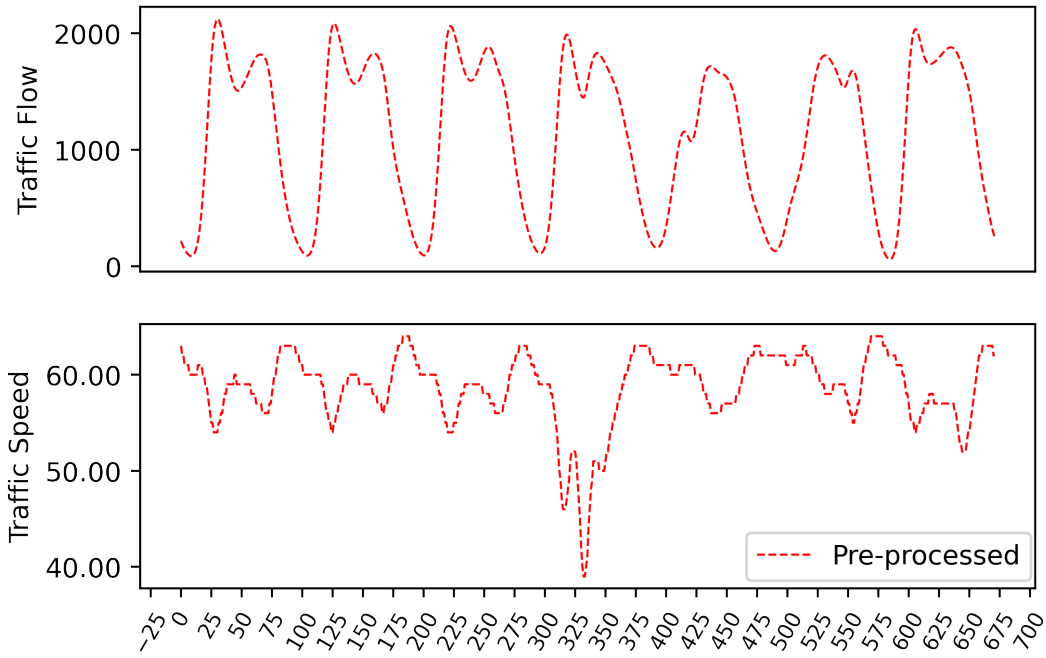


Figure 4.13: Pre-processed traffic flow and speed in a randomly selected week from `PI-London`. The x-axis and y-axis represent time interval and traffic flow (top) and/or speed (bottom), respectively. The unit of traffic flow is the number of vehicles and the unit of traffic speed is "miles/hour". A time interval is considered as 15 minutes and a week has $\frac{7\times24\times60}{15} = 672$ time intervals.

7 previous days and weeks respectively. The number of previous time intervals n and the number of stations m in the $\mathbf{TS_t}$ are respectively set as 3 and 8 in the intersections and 3 and 5 in the linear roadways (See Section 4.3).

2. ***Parameters in the ARIMA module:*** Lag order, p, differentiating times, d, and moving average window, q are respectively set to 9 (equal to k_1), 1 and 0, which are decided by analysing the autocorrelation coefficient and partial autocorrelation coefficient of matrix $\mathrm{T}_{\mathrm{t}}^{\mathrm{i}}$.

3. ***Parameters in the LSTM module:*** The parameters needed to tune in our LSTM modules are memory units n_d for the input matrix $\mathrm{D}_{\mathrm{t}}^{\mathrm{i}}$ and n_w for the input matrix $\mathrm{W}_{\mathrm{t}}^{\mathrm{i}}$. Based on the number of time intervals defined in the input matrix $\mathrm{D}_{\mathrm{t}}^{\mathrm{i}}$ and $\mathrm{W}_{\mathrm{t}}^{\mathrm{i}}$, we set $n_d = k_2$ and $n_w = k_3$.

4. ***Parameters in the SAE module:*** For SAE, we search the optimal number of hidden layers n_s between 1 and 6, and the number of neurons n_{su} in each of hidden layers from 200 to 400 (step size = 50). More layers can capture more information from input, but it costs more time.

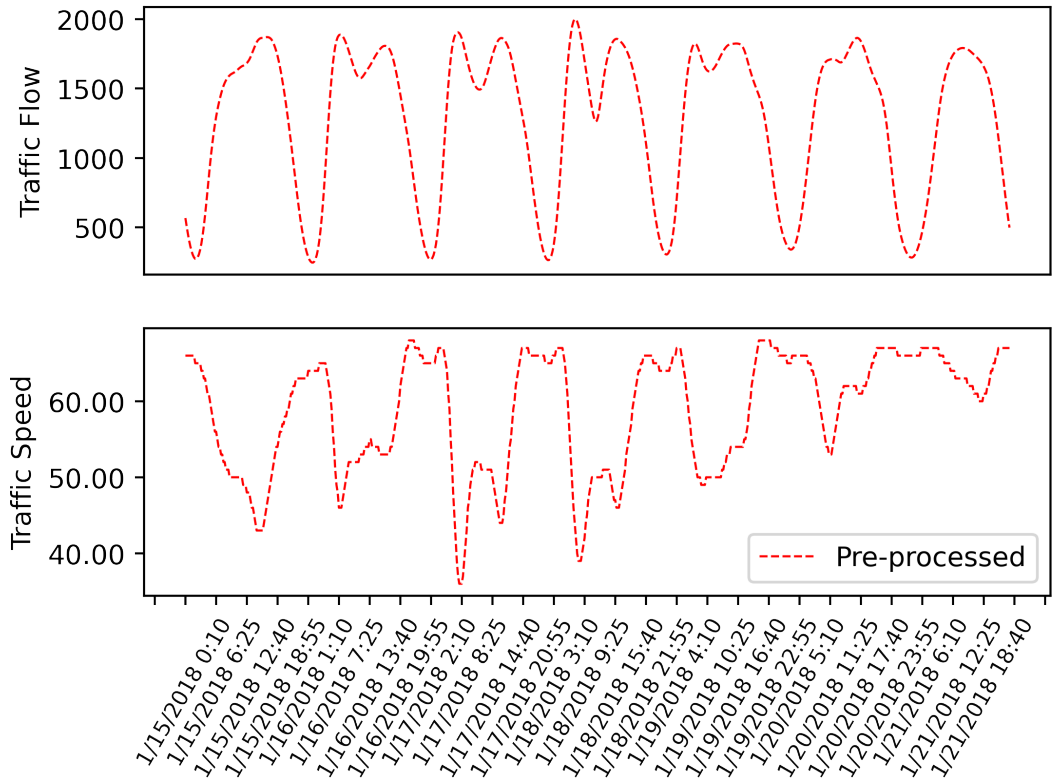


Figure 4.14: Pre-processed traffic flow and speed in a randomly selected week from `PL-Los`. The x-axis and y-axis represent time and traffic flow (top) and/or speed (bottom), respectively. The unit of traffic flow is the number of vehicles and the unit of traffic speed is "miles/hour".

5. ***Parameters in the CAPSNET module:*** The parameters for CAPSNET is given in Table 4.2. There are four layers including two conventional convolutional layers, one primary capsule layer (namely PrimaryCaps) and one traffic capsule layer (namely TrafficCaps). The two conventional convolutional layers are used to capture the temporal-spatial features of short-term traffic flow, in which the kernel size in both conventional convolutional layers and activation function are respectively 3×3 and "ReLU". Convolution operations are performed with 2 as the stride and zero padding. The PrimaryCaps layer is a convolutional layer with 128 channels with 3×3 kernel size. It has 16 $(128/8)$ capsules and each capsule is an 8-dimensional vector. The difference when compared to conventional convolutional layers is that the activation function in this layer is "Squashing" function (cf. Eq. (4.14)) rather than "ReLU". This activation function is also used in the TrafficCaps layer with 16 advanced capsules and each of capsules has a 16-dimensional vector. The advantage of using this in our work is that it produces output in a vector consisting of 16 values to allow us taking more traffic information than a scalar

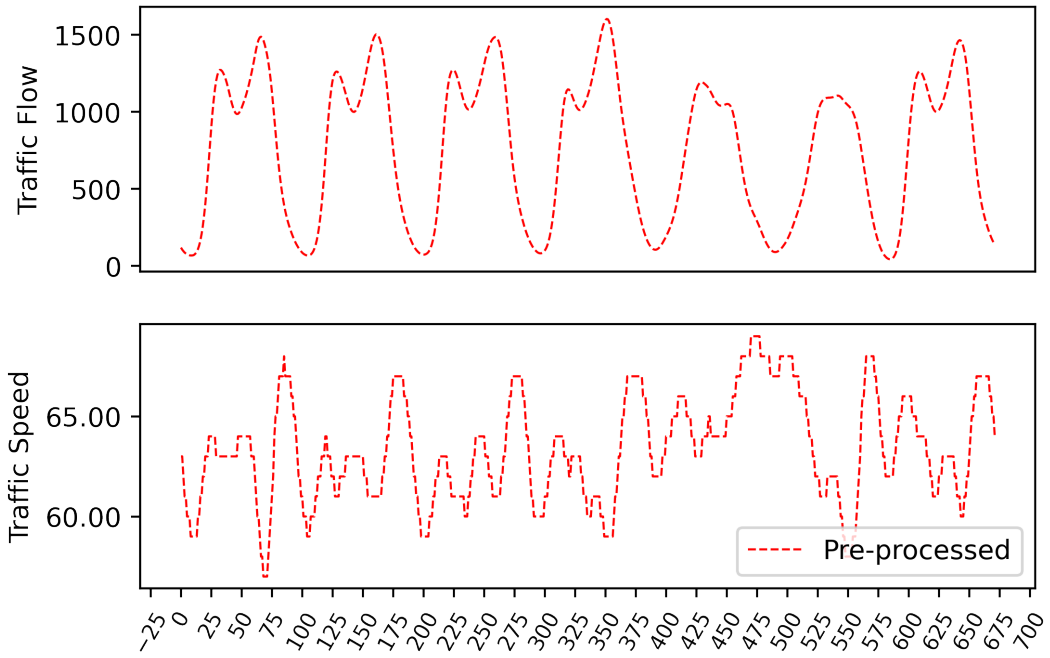


Figure 4.15: Pre-processed traffic flow and speed in a randomly selected week from `PL-London`. The x-axis and y-axis represent time interval and traffic flow (top) and/or speed (bottom), respectively. The unit of traffic flow is the number of vehicles and the unit of traffic speed is "miles/hour". A time interval is considered as 15 minutes and a week has $\frac{7\times24\times60}{15} = 672$ time intervals.

value obtained by other activation functions.

Table 4.2: Parameter Setting in the CAPSNET module

Layer name	Parameter	Activation
Convolution	(3, 3, 64)	ReLU
PrimaryCaps	(3, 3, 128) Capsule dimension = 8	Squashing
TrafficCaps (Fully connected) (Flattened)	Advanced capsule = 16 Capsule dimension = 16 256	Squashing

4.4.2 Results and Discussion

We compare the performance of our ALLSCP against a time-series prediction model (ARIMA), a simple machine learning model (SVR), four deep learning models (LSTM,

SAE, CNN and CAPSNET) and two ensemble models (DA and CLTFP) on intersections and also linear roadways.

Linear Roadways

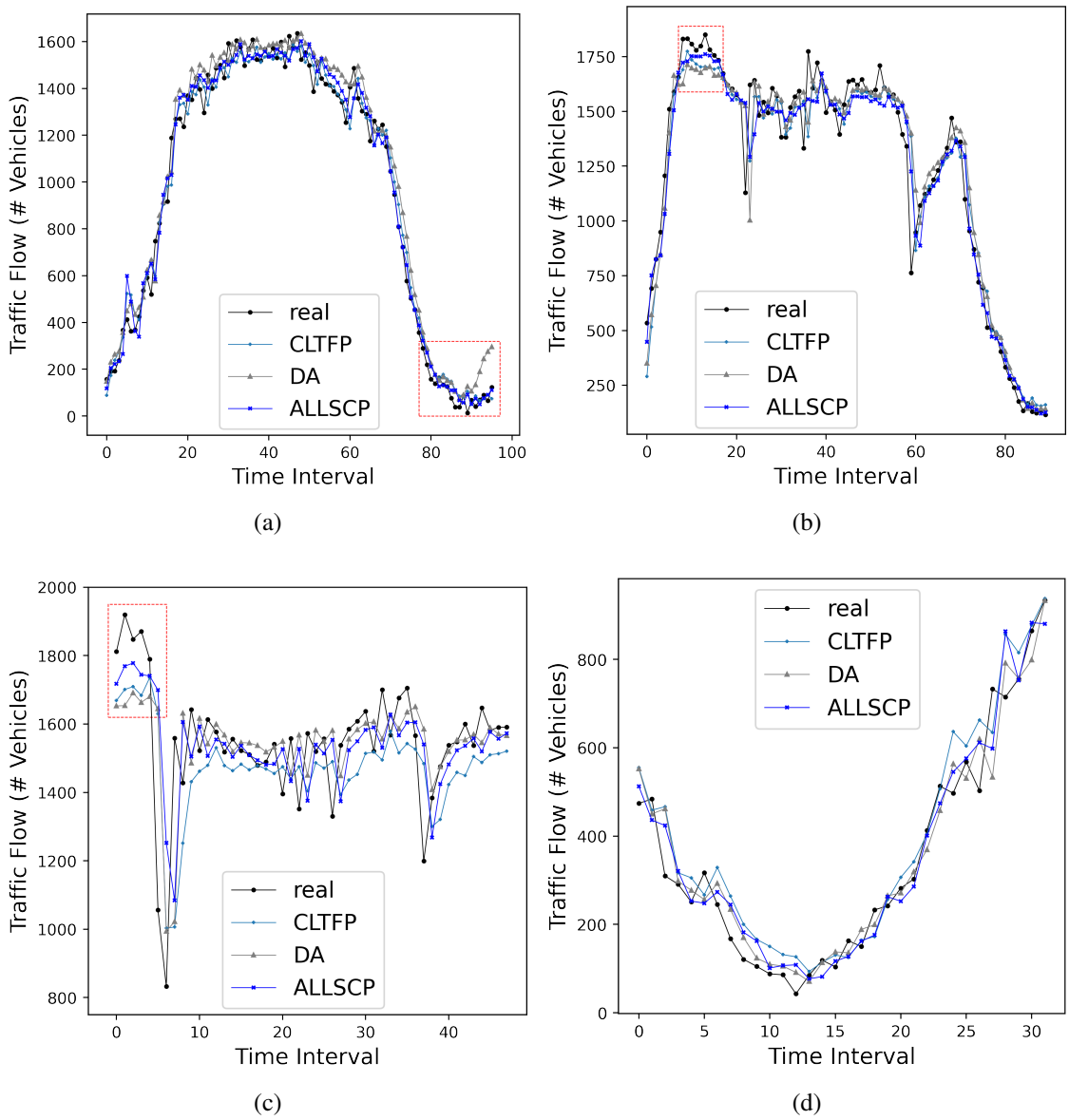


Figure 4.16: Real traffic flow (black) and predicted traffic flow predicted by our ALLSCP (blue) and other two existing ensemble models (other colours) in different traffic situations for `L-Los`. Notes that each sub-figure shows a different traffic situation. Red boxes in sub-figures show obvious difference between ALLSCP and other two existing models.

We compare the original traffic flow against the predicted traffic flow from ALLSCP and other two existing ensemble models (CLTFP ad DA) for `L-Los` in Figure 4.16 and

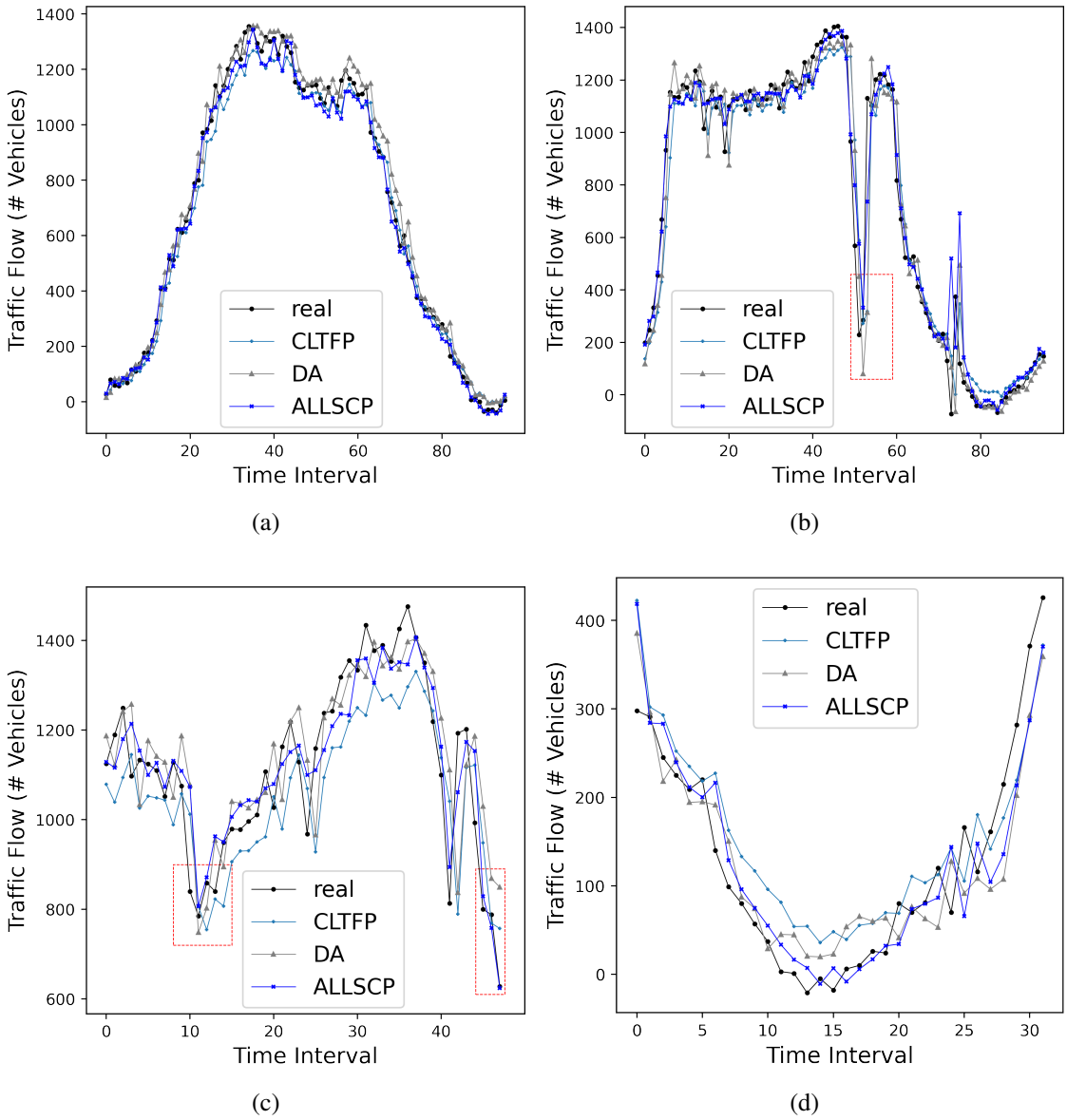


Figure 4.17: Real traffic flow (black) and predicted traffic flow predicted by our ALLSCP (blue) and other two existing ensemble (other colours) models in different traffic situations for `L-London`. Notes that each sub-figure shows a different traffic situation. Red boxes in sub-figures show obvious difference between ALLSCP and other two existing models.

`L-London` in Figure 4.17 under different traffic situations. Figure 4.16(a) and Figure 4.17(a) present the traffic flow over a 24-hour period with normal traffic condition from `L-Los` and `L-London` respectively, while Figure 4.16(b) and Figure 4.17(b) show the traffic flow over a 24-hour period with an abnormal traffic situation (i.e., with various traffic incidents). In addition, Figure 4.16(c) and Figure 4.17(c) show the traffic flow during rush hours over a period of 12 hours, and Figure 4.16(d) and Figure 4.17(d) show the off-peak time over a period of 8 hours. Especially for Figure 4.16(c) and Figure 4.16(c)

during rush hours, our model can both capture sudden changes as well as finer traffic changes. Red boxes in all sub-figures show obvious differences between ALLSCP and other two existing ensemble models. Overall, ALLSCP captures the traffic flow changes, following closely the diurnal pattern exhibited in road traffic, even under abnormal traffic conditions.

Figure 4.18 shows the accuracy (i.e., $(100\% - MAPE)$) of the models across `L-Los`, `PL-Los`, `L-London` and `PL-London` datasets. For all cases, our ALLSCP achieves the best accuracy. Specifically for cases using original traffic data, ALLSCP achieves accuracy of 93.86% and 95.05% for Los Angeles and London respectively while the rest of the models on average only achieve accuracy of 91.84% and 92.59%. Amongst the considered models, DA and CAPSNET are the second best models for `L-Los` and `L-London` respectively with a performance gap of 1.11% and 1.78% when compared to ALLSCP. On the other end of the spectrum, the worst performing model for both cases are CNN as it is only capable of capturing local-spatial features. Furthermore, when we use de-noised data, ALLSCP's accuracy further improved to 98.16% (4.3% improvement) and 97.50% (2.45% improvement) for `PL-Los` and `PL-London` respectively. DA remains to be the closest rival for prediction on Los Angeles traffic with 96.88% accuracy. However, for London, CNN's prediction accuracy improves significantly to become the second best (95.83%). LSTM which mainly focuses on temporal feature extraction is the worst performer when using de-noised London data, indicating the traffic pattern near Heathrow airport is more complex. In fact, we note that while for Los Angeles, all models achieve improved accuracy, this is not the case for London when the prediction accuracy for ARIMA, LSTM, SAE and SVR worsened, again mainly due to the higher volatility in traffic near Heathrow airport. Our ALLSCP model achieves overall higher prediction accuracy due to its ability to capture different temporal (short, medium and long) and spatial (global and local) features. For instance, we exploit CAPSNET's feature on encapsulating important information related to local features into a vector form that can carry more information than a scalar value. From our results, this encapsulation alone is not sufficient but using our ensemble model, we achieve better accuracy. Moreover, due to the periodic properties of traffic data, the carefully designed input (including short, medium and long temporal traffic data on the targeted station and on its neighbouring stations) also contributes to the enhancement of the prediction accuracy.

Table 4.3 presents the MAE, MAPE and RMSE achieved by all models on `L-Los`, `PL-Los`, `L-London` and `PL-London`. Between a statistic model and a simple machine learning model (i.e., ARIMA and SVR), SVR performs better. This is mainly due to the non-linear relationship between traffic flow in different time intervals which ARIMA

Table 4.3: Comparison of all models for both linear roadways

Model	L-Los			PL-Los			L-London			PL-London		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
ARIMA	81.89	0.0884	109.55	72.21	0.0662	97.62	59.37	0.0890	92.10	58.64	0.0743	92.32
LSTM	74.56	0.0778	97.42	79.42	0.0696	104.36	79.20	0.1027	106.03	50.25	0.0678	83.49
SAE	59.68	0.0829	113.42	62.35	0.0636	100.87	60.58	0.0815	104.93	53.83	0.0756	111.90
CAPSNET	85.74	0.0781	117.17	36.85	0.0329	48.96	44.69	0.0673	80.18	31.39	0.0436	53.14
CNN	95.91	0.0885	128.60	42.94	0.0403	68.00	58.58	0.0866	105.15	32.92	0.0417	99.62
SVR	76.39	0.0803	103.53	57.74	0.0559	81.95	51.70	0.0768	82.08	56.83	0.0728	95.58
CLTFP	**40.67**	0.0845	**50.46**	**13.28**	0.0378	**19.67**	**30.19**	0.0783	**40.62**	45.48	0.0763	88.52
DA	77.65	0.0725	103.51	14.32	0.0312	22.37	58.56	0.0703	92.21	28.45	0.0521	43.51
ALLSCP	71.64	**0.0614**	95.08	20.10	**0.0184**	26.84	36.29	**0.0495**	65.80	**18.22**	**0.0250**	**25.90**

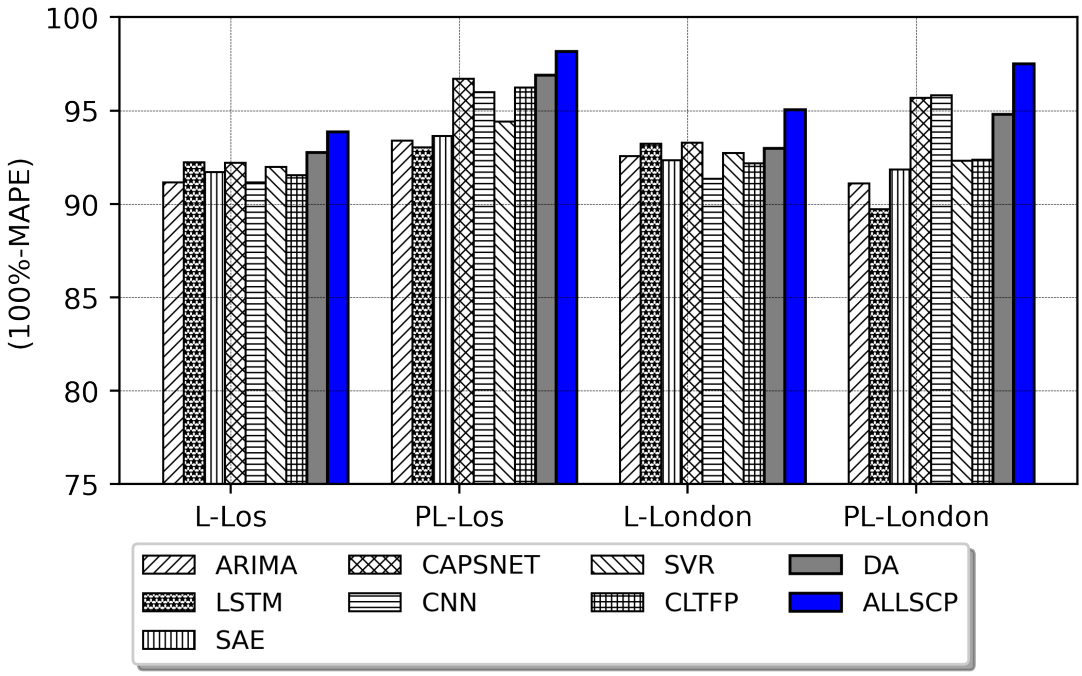


Figure 4.18: $(100\% - MAPE)$ achieved on `L-Los` (left), `PL-Los` (middle-left), `L-London` (middle-right) and `PL-London` (right) collected from linear roadways. ALLSCP achieves the best accuracy for all cases.

fails to take into account whereas SVR with a non-linear kernel function (i.e., RBF kernel function) is capable of mapping a non-linear vector to a high dimensional feature space for conducting linear regression. Meanwhile, the four deep learning models (i.e., LSTM, SAE, CNN and CAPSNET) generally achieve better predictions compared to simple machine learning models. For instance, LSTM focusing on time-series data using the cell state to store information on long-term dependencies of traffic data outperforms both ARIMA and SVR. For the SAE model, full connection is used between hidden layers. Therefore, it missed the contribution of local features for traffic prediction. Compared to SAE model, CNN can capture local-spatial feature for obtaining better result because of convolutional kernels. Based on CNN, CAPSNET converts scalar values representing features into a vector form to obtain more detailed information for traffic prediction. This is the reason that CAPSNET model obtains best results, especially on `PL-Los` (96.71%) and `PL-London` (95.64%). This implies the importance of spatial information for traffic prediction. Furthermore, between the two ensemble models (CLTFP and DA), MAPE of DA model is lower on four datasets, and the other two metrics (MAE and RMSE) of CLTFP model are lower except on `PL-London`. DA model mainly depends on temporal feature extraction for prediction while CLTFP model extracted temporal-spatial features for the final prediction. This again indicates the importance of simultaneously taking the temporal and spatial features on the problem of traffic prediction. Finally, while our

ALLSCP consistently achieves the best accuracy, CLTFP relegates it to second best in some cases for MAE and RMSE.

Intersections

For intersections, ALLSCP achieves the best accuracy across both original and pre-processed datasets at different locations (i.e., at e_i and g_i). We present the $(100\% - MAPE)$ results for e_i in Figure 4.19[1]. ALLSCP achieves an average of 95.53% accuracy for both e_i and g_i across all datasets while the average achieved accuracy by the other eight models is 91.10%. SAE seems particularly challenged for Los Angeles datasets with significantly lower accuracy achieved compared to other models (even dipping below 70% accuracy for g_i for `I-Los`). The main reason for this is that SAE model uses unsupervised learning method to reduce feature dimensions to obtain important hidden information instead of original data. Furthermore, we see a general trend of improved accuracy achieved when pre-processed datasets are used. For instance, we see a 12.57% improvement in accuracy for SAE for `PI-Los` compared to `I-Los`. This implies de-noised data offer better input for short-term traffic predictions.

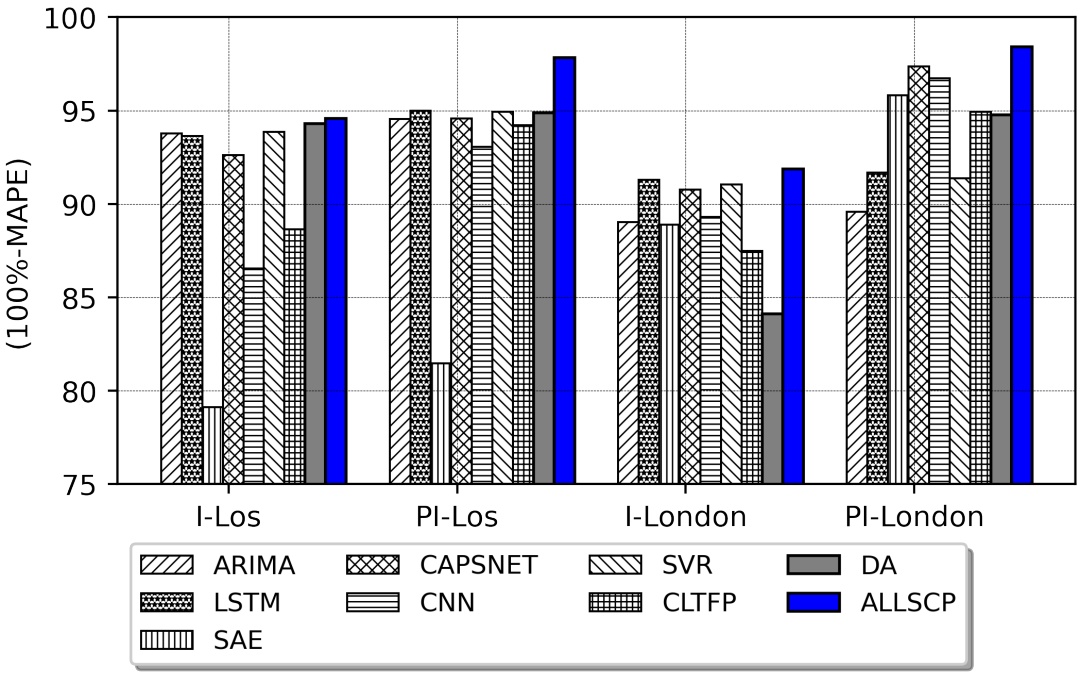


Figure 4.19: $(100\% - MAPE)$ for e_i on `I-Los` (left), on `PI-Los` (middle-left), on `I-London` (middle-right), and on `PI-London` (right). ALLSCP achieves the best accuracy for four cases.

[1]We omit the plot for g_i as it is qualitatively similar.

Table 4.4 and Table 4.5 respectively present the prediction results of stations e_i and g_i. We observe cases where CLTFP achieves lower prediction error in terms of MAE and RMSE when compared to ALLSCP. Our results thus suggest that CLTFP is capable of reducing errors in absolute terms while appears to be less accurate with relative errors when ALLSCP performs better. This could be due to the fact that for CLTFP ensemble model, it exploits both LSTM and CNN models as constituent models while for ALLSCP, we proposed to use CAPSNET in place of CNN which as prior mentioned represents features in vector form rather than scalar values.

Ablation Experiment Results of the Proposed Model

We conduct ablation experiments by removing one module one at a time, and then compare it to our full proposed model. Five variants of ALLSCP including 1) LLSCP by removing ARIMA module for short-term temporal feature analysis, 2) A-LSCP by removing LSTM module for medium-term temporal feature analysis, 3) AL-SCP by removing LSTM for long-term temporal feature analysis, 4) ALLCP by removing SAE for global spatial feature analysis and 5) ALLS by removing CAPSNET for local spatial feature analysis, are used for ablation experiments on four raw datasets.

Table 4.6 shows results of ablation experiments on two datasets (`L-Los` and `L-London`) from linear roadways. Overall, ALLSCP outperforms its five variants. On `L-Los`, LLSCP performs the worst. This indicates that ARIMA used for short-term temporal feature analysis is the most important module in our ALLSCP. The second worst variant is ALLS, which means CAPSNET used for analysing local spatial features is also very important for short-term traffic prediction and its importance only inferiors to ARIMA. Among five variants, AL-SCP achieves best results, followed by A-LSCP and ALLCP. This indicates that, for short-term traffic prediction, LSTM for long-term temporal feature analysis is less important than other four modules. Similar results can be found on `L-London`. Therefore, on linear roadways, the importance ranking of modules from most to less in our ALLSCP is {ARIMA, CAPSNET, SAE, LSTM for medium-term temporal feature analysis, LSTM for long-term temporal feature analysis}. It shows that the short-term temporal and local spatial features are two most important features for short-term traffic prediction on linear roadways.

Table 4.7 presents ablation experimental results on two datasets (`I-Los` and `I-London`) from intersections. All models show better results on intersections than on linear roadways, and our proposed model, ALLSCP, still achieves the best performance. The reason for this is that more features in $\mathbf{TS_t}$ from intersections can be used for improving pre-

Table 4.4: Comparison of all models for the intersection $x_t^{e_i}$

Model	$x_t^{e_i}$ (I-Los)			$x_t^{e_i}$ (PI-Los)			$x_t^{e_i}$ (I-London)			$x_t^{e_i}$ (PI-London)		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
ARIMA	66.11	0.0622	92.42	59.36	0.0545	87.75	80.85	0.1097	126.93	73.04	0.1042	114.03
LSTM	64.57	0.0636	87.17	54.16	0.0501	79.78	66.17	0.0871	111.02	65.78	0.0832	104.48
SAE	169.22	0.2088	277.82	150.86	0.1852	221.46	83.78	0.1102	143.76	26.82	0.0418	40.68
CAPSNET	77.79	0.0738	103.03	58.12	0.0543	73.00	72.47	0.0922	117.88	20.68	0.0263	29.24
CNN	126.52	0.1346	225.44	61.76	0.0694	89.86	81.92	0.1070	134.18	25.44	0.0327	35.83
SVR	63.99	0.0613	90.74	54.52	0.0508	80.40	67.77	0.0896	112.34	66.67	0.0863	105.32
CLTFP	**32.84**	0.1135	**43.59**	**19.72**	0.0579	**27.71**	89.24	0.1253	145.48	34.01	0.0507	46.46
DA	61.00	0.0570	83.20	55.32	0.0511	81.63	75.56	0.1589	119.14	21.32	0.0524	30.14
ALLSCP	58.91	**0.0542**	79.90	24.28	**0.0218**	31.49	**63.23**	**0.0813**	**105.09**	**13.31**	**0.0158**	**18.68**

Table 4.5: Comparison of all models for the intersection $x_t^{g_i}$

Model	$x_t^{g_i}$ (I-Los)			$x_t^{g_i}$ (PI-Los)			$x_t^{g_i}$ (I-London)			$x_t^{g_i}$ (PI-London)		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
ARIMA	45.09	0.0843	61.33	36.56	0.0676	49.93	37.85	0.1035	52.13	34.10	0.0899	51.86
LSTM	42.04	0.0779	57.19	33.67	0.0631	45.78	30.99	0.0829	43.34	26.65	0.0798	41.69
SAE	157.42	0.3194	228.09	91.84	0.1937	164.81	36.01	0.0979	49.25	34.10	0.0946	46.44
CAPSNET	62.79	0.1186	82.73	27.56	0.0476	34.06	34.03	0.0873	47.05	11.94	0.0295	16.02
CNN	82.85	0.1631	125.16	32.75	0.0639	46.74	38.17	0.0977	52.96	12.44	0.0309	20.32
SVR	43.62	0.0838	60.84	33.77	0.0634	45.86	30.92	0.0831	43.00	29.10	0.0816	41.35
CLTFP	**21.63**	0.1204	**29.30**	11.07	0.0603	15.10	45.43	0.1186	63.11	21.71	0.0627	28.77
DA	40.69	0.0746	54.88	33.00	0.0608	44.66	35.19	0.0960	48.76	32.47	0.0931	47.01
ALLSCP	40.26	**0.0740**	54.73	**9.38**	**0.0171**	**12.45**	**29.09**	**0.0761**	**40.60**	**7.07**	**0.0171**	**9.80**

Table 4.6: The results of ablation experiments on linear roadways.

Model	L-Los			L-London		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
LLSCP	83.08	0.0741	108.79	42.97	0.0723	107.89
A-LSCP	73.37	0.0664	97.02	38.57	0.0545	95.99
AL-SCP	71.71	0.0645	95.16	36.35	0.0496	95.11
ALLCP	73.66	0.0665	97.49	41.13	0.0556	100.26
ALLS	78.22	0.0693	104.68	57.78	0.0802	119.21
ALLSCP	**71.64**	**0.0614**	**95.08**	**36.29**	**0.0495**	**95.08**

diction accuracy. Among five variants, ALLS obtains the worst results, which indicates that CAPSNET module, that is responsible of analysing local spatial features, takes most important position in ALLSCP. AL-SCP is the best variant and its performance is very close to ALLSCP, which means LSTM used for long-term temporal feature analysis is less important than other four modules. Overall, the importance ranking of modules from most to less in our ALLSCP on intersections is {CAPSNET, ARIMA, SAE, LSTM for medium-term temporal feature analysis, LSTM for long-term temporal feature analysis}. It shows that the local spatial and short-term temporal features are two most important features for short-term traffic prediction on intersections, and the local and global spatial features become more important than on linear roadways.

Prediction Stability and Robustness

We have shown that ALLSCP is consistent in making the best prediction accuracy for different scenarios. Hence, ALLSCP behaves stably over the different prediction scenarios. We also observe that generally, models perform better using pre-processed datasets when compared to using raw traffic data. For instance, the average accuracy of ALLSCP improves from 94.46% to 97.83% on linear roadways and from 92.76% to 98.29% on intersections.

We proceed to compare the improvement of $(100\% - MAPE)$ between raw and pre-processed data. We present the results in Table 4.8. All models with convolutional layers (i.e., CNN, CLTFP, CAPSNET, ALLSCP) achieve improved accuracy after removing noises using HP filter. This is due to the use of convolutional operator that is commonly used to extract edge features in image recognition applications. In our case for traffic flow, the edge feature corresponds to the difference of traffic flow between two continuous

Table 4.7: The results of ablation experiments on intersections.

Model	$x_t^{e_i}$ (I-Los)			$x_t^{g_i}$ (I-Los)			$x_t^{e_i}$ (I-London)			$x_t^{g_i}$ (I-London)		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
LLSCP	62.30	0.0582	84.09	44.16	0.0834	59.50	63.67	0.0872	106.46	31.46	0.0882	43.04
A-LSCP	62.84	0.0584	84.83	43.55	0.0818	59.27	65.13	0.0901	111.79	31.21	0.0852	42.67
AL-SCP	62.86	0.0584	85.09	43.35	0.0813	59.11	63.65	0.0869	108.43	31.26	0.0858	43.11
ALLCP	63.85	0.0593	85.88	43.71	0.0822	59.44	66.13	0.0909	113.57	31.23	0.0850	42.82
ALLS	66.12	0.0615	92.09	45.11	0.0851	61.21	77.93	0.1087	125.22	36.63	0.1056	50.45
ALLSCP	**58.91**	**0.0542**	**79.90**	**40.26**	**0.0740**	**54.73**	**63.23**	**0.0813**	**105.09**	**29.09**	**0.0761**	**40.60**

time intervals. Sudden changes (e.g., due to traffic accidents) causes unbalanced learning (Pang et al. 2019) and negatively affects the prediction. Pre-processing data smooths and reduces the differences between two intervals to enable us to obtain better results. DA also achieves improvement since DA is fed with more detailed temporal information (e.g., hourly, daily and weekly).

Table 4.8: Accuracy improvement when comparing raw data against de-noised data for different models.

Model	L-Los	L-London	I-Los (e_i)	I-Los (g_i)	I-London (e_i)	I-London (g_i)	Variance
ARIMA	2.22	-1.47	0.77	1.67	0.55	1.36	1.38
LSTM	0.82	-3.49	1.35	1.48	0.39	0.31	2.83
SAE	1.93	-0.59	2.36	12.57	6.84	0.33	30.41
CAPSNET	4.52	2.37	1.95	7.10	6.59	5.78	3.92
CNN	4.82	4.49	6.52	9.92	7.43	6.68	3.22
SVR	2.44	-0.40	1.05	2.04	0.33	0.15	1.04
CLTFP	4.67	0.20	5.56	6.01	7.46	5.59	5.14
DA	4.13	1.82	0.59	1.38	10.65	0.29	12.81
ALLSCP	4.30	2.45	3.24	5.69	6.55	5.90	2.20

From Table 4.8, we see that when comparing using raw and de-noised data, ALLSCP is among the models achieving the best improvements (average improvement = 4.69%). CNN achieves the highest improvements when using de-noised datasets with an average improvement of 6.64% though as we have shown before, its accuracy is much worse than ALLSCP. Furthermore, ARIMA, LSTM and SVR only achieve minimal improvements (i.e., below 1%). Along with SAE, these models even achieve worse performance using de-noised datasets for linear roadways in London. In terms of prediction stability (from the perspective of variance of the prediction improvements), SVR is the most stable with lowest variance among all models (i.e., only 1.04). The second lowest variance is achieved by ARIMA (i.e., 1.38) followed closely by our ALLSCP with variance of 2.20. Although the variances of SVR and ARIMA on all datasets are lower than ALLSCP, the prediction accuracy of ALLSCP is consistently higher than those two models. Therefore, considering both the prediction accuracy and the variance of improvements on all datasets, our ALLSCP is more stable, robust and accurate.

4.5 Chapter Summary

In this chapter, we present a novel model for addressing the problem of short-term traffic flow prediction on intersections; a problem that has received renewed attention due to the development of smart city visions. Taking into account five important features: 1) short-term temporal features, 2) medium-term temporal features, 3) long-term temporal features, 4) global spatial features, and 5) local spatial features, our model exploits the strengths of four modules, namely ARIMA, LSTM, SAE and CAPSNET to make our predictions.

We examine our proposed model, ALLSCP, across intersections at two different locations (Los Angeles and London) where frequent congestion and accidents are expected. Meanwhile, we also validate our proposed model on two linear roadways at these two locations again for showing the generality of our model. We use both raw traffic data as well as pre-processed (i.e., de-noised) data, and compare our ALLSCP against existing models in the literature including its constituent modules, two single models (namely SVR and CNN) and two existing ensemble models in the literature (namely DA and CLTFP).

Our ALLSCP model achieved the highest accuracy among the nine considered models, achieving an average of 96.14% and 95.53% accuracy for linear and intersection roadways respectively while on average, the other competing models achieved 93.10% and 91.10% for the corresponding scenarios. Our results show that ALLSCP model is not only accurate but also the most robust, recording the least accuracy degradation when making predictions for the more challenging data with frequent congestion and accidents.

Chapter 5

Dynamic Spatial-Temporal Feature-Based Multi-Steps Traffic Speed Prediction on Large-Scale Road Networks

5.1 Introduction

In this chapter, we turn our attention to the road network as a whole, i.e., instead of focusing on specific road segment (either linear as studied in Chapter 3 or intersection as studied in Chapter 4), we are now considering the entire road network. Traffic speed prediction on large-scale road networks is known to be a more challenging task with complex inter-dependencies temporally and spatially. Both traffic speed and flow data exhibit such patterns and characteristics. For example, temporally, it is found that there is non-linear temporal dynamics of traffic speed over time depending on the changing road conditions (e.g., (Zhang 2003)). Furthermore, traffic speed data also show periodic patterns (e.g., weekly and seasonal changes). Thus, current traffic speed on a location depends not only on the immediate previous epochs but may also correlate with longer periodic patterns or trends. Hence, both long- and short-temporal dependencies in traffic time sequences should be taken into account when making traffic speed predictions.

Since road networks are inherently spatial networks, the traffic speed within the network logically depends on the topological structure of the road interconnections. Local spatial dependency can be seen from observing that the traffic speed at one road segment is

affected by both its immediate upstream and downstream road segments. Furthermore, since the vehicles are travelling within the same physical road network, the traffic condition of different road segments across the network are inter-dependent. This can be exemplified in various gridlock phenomena taking place in urban cities during peak hours whereby congestion at one road segment quickly cascade and spread to other locations.

Early attempts on traffic speed prediction focus on temporal dependencies of traffic over time. Both long- and short-term dependencies in traffic time sequences were studied. Spatial dependencies were later jointly considered, initially focusing on capturing dependencies from immediate upstream and downstream road segments and more recently, taking into account the influence of the entire road network topology. However, the literature has mostly considered that the physical road network topology (i.e., the road segments and their connectivity) is fixed and neighbouring nodes contribute equally to the future traffic status of the targeted node (we refer this as *fixed spatial* dependency).

In this chapter, we argue that each neighbouring node is distinct and has different influence to the targeted node (i.e., the degree of the spatial dependency varies from one neighbour to another). In the space dimension, intuitively, closer neighbouring nodes have stronger influence while traffic status at road segments further away gradually have less impact. However, this relationship is not strict as it also depends on the local connectivity of the different neighbouring nodes and it is possible to have a node further away having greater influence on the targeted node. In the time dimension, a congestion lasting longer period should have wider spatial impact and thus, a road segment will be more influenced by neighbouring nodes further away if the network suffers comparatively long congestion period and vice versa. As such, we assert that capturing and quantifying such *dynamic spatial* dependency is important to further improve traffic prediction accuracy.

Building on the most recent developments in deep learning, we thus develop a novel prediction model, named SAGCN-SST (Self-Attention Graph Convolutional Network with Spatial, Sub-spatial and Temporal blocks), and address multi-step traffic speed prediction problem for large-scale road networks. Apart from considering fixed spatial and long temporal dependencies, the learning architecture in our SAGCN-SST is designed to also simultaneously capture the dynamic spatial dependency for both short- and long-term traffic prediction. We join a self-attention mechanism into our graph convolutional layers to capture how different neighbouring nodes contribute to the future traffic status of the targeted node. In our framework, we use parallel sub-blocks for different neighbourhoods, avoiding increasing model depth and complexity. We validate our proposed framework using two real-world traffic datasets from large-scale road networks (Seattle and Los Angeles) and also conduct a comparative study across well-known existing models in the

recent literature. The results indicate our SAGCN-SST performs better than other leading models in literature.

The rest of this chapter is organised as follows. We first define our traffic prediction problem and then detail the design rationale and the architecture of our novel deep learning framework in Section 5.2. Section 5.3 describes two real-world datasets collected from large-scale road networks while Section 5.4 evaluates our SAGCN-SST and compares it with well-known existing models in the recent literature on these two datasets. Finally, we summary our work in Section 5.5.

5.2 Methodology

5.2.1 Problem Definition

Road Network Representation

Graph theory is an effective tool to capture spatial features of data by analysing the connectivity between detectors or road segments (May 1990). In this chapter, we model the road network as an undirected graph since the traffic speed depends on its downstream while traffic congestion propagates upstream (Long et al. 2008) and the datasets used in this chapter includes traffic congestion events.

Considering a road network represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes representing sensor locations or road segments with $|\mathcal{V}| = N$. $\mathcal{E}$ is the set of edges representing physical connectivity between road segments. $\mathcal{G}$ can be represented by $\mathbf{A} \in \mathbb{R}^{N \times N}$, the $N \times N$ symmetric adjacency matrix, with its element $A_{i,j} = 1$ if there exists a link between node i and j and 0 otherwise. The degree matrix of graph $\mathcal{G}$, $\mathbf{D} \in \mathbb{R}^{N \times N}$ is then defined as $D_{i,i} = \sum_j A_{i,j}$, which sums the number of edges connected to each node.

The future traffic status of a node is influenced by its own current status. For this, we further define $\tilde{\mathbf{A}} = (\mathbf{A} + \mathbf{I}) \in \mathbb{R}^{N \times N}$ where $\mathbf{I}$ is the $N \times N$ identity matrix. This accounts for the fact that each node is also self-influenced. As such, the trace $tr(\tilde{\mathbf{A}}) = \sum_{i=1}^{N} A_{i,i} = N$. $\tilde{\mathbf{A}}$ only describes the connectivity of neighbours one hop away from each node (i.e., $1 - hop$ neighbourhood). We further introduce the notion of $k - hop$ neighbourhood to represent the set of nodes that are reachable within k hops from the targeted node. We define the $k - hop$ neighbourhood matrix as $\tilde{\mathbf{A}}^{\mathbf{k}} \in \mathbb{R}^{N \times N}$. The reason for introducing the $k - hop$ matrix is to account for the fact that traffic congestion not only propagate to

its immediate upstream and downstream road segments but also often spread in a certain area in the network (Nguyen et al. 2016).

Traffic information

Let v_t^i denotes the traffic speed measured at node i at t^{th} time interval. Typically, a time interval can represent 5, 15, 30, 45 and 60 mins (Bickel et al. 2007). In this chapter, we use 5-min time interval. Given a large-scale road network, the traffic speed on N detectors is then written as $\mathrm{v_t} = \{v_t^1, v_t^2, \ldots, v_t^i, \ldots, v_t^{N-1}, v_t^N\}; \mathrm{v_t} \in \mathbb{R}^N$, $(i = 1, 2, 3, \ldots, N)$. Then $\mathbf{V} = \{\mathrm{v_{t-T+1}}, \mathrm{v_{t-T+2}}, \ldots, \mathrm{v_{t-1}}, \mathrm{v_t}\}; \mathbf{V} \in \mathbb{R}^{T \times N}$, $(T = 1, 2, 3, \ldots)$ gives the traffic speed collected from N detectors in the network for the past T time intervals. Conversely, the traffic speed for the future is written as $\mathbf{V}' = \{\mathrm{v_{t+1}}, \mathrm{v_{t+2}}, \ldots, \mathrm{v_{t+T'}}\} \in \mathbb{R}^{T' \times N}$ where T' is the prediction horizon. Generally, traffic prediction problems can be categorised into short- ($T' < 30$ mins) and long-term ($T' \geq 30$ mins). Since we are addressing multi-step prediction problem, our solution covers both timescales whereby $T' = \{1, 3\}$ for short-term and $T' = \{6, 9, 12\}$ for long-term traffic speed prediction corresponding to $\{5, 15\}$ mins and $\{30, 45, 60\}$ mins respectively (Min & Wynter 2011).

Problem Formulation

Given past traffic speed, $\mathbf{V}$ and the road network $\mathcal{G}$, our aim is to predict traffic speed, $\mathbf{V}'$, in the future T' time intervals. The problem can then be represented as in Eq. (5.1).

$$\mathbf{V}' = F\Big(\mathbf{V}; \mathcal{G}\Big(\mathcal{V}, \mathcal{E}, \tilde{\mathbf{A}}^{\mathbf{k}}\Big)\Big) \tag{5.1}$$

where the objective is to learn the mapping function $F(.)$ and compute the traffic speed in the next T' time intervals given the traffic speed in the past T time intervals and network information including the different k neighbourhood matrices as input.

5.2.2 Design Overview

Figure 5.1 presents the overall learning architecture of our SAGCN-SST framework. It consists of three main blocks:

- **Input block** – This block is responsible for preparing the raw traffic and graph data into a trainable format as input to the spatial block.
- **Spatial block** – This block extracts both fixed and dynamic spatial features. Based on GCN, a memory-less model, we construct $k-hop$ neighbourhoods for each node in the network and utilise a self-attention mechanism to learn the degree of influence of different individual neighbour to the targeted node. The k spatial features extracted in this block is concatenated (i.e., $\mathbf{SAGCN} = \{\mathbf{SAGCN^1}, \mathbf{SAGCN^2}, \ldots, \mathbf{SAGCN^k}\}$) as input to the temporal block.
- **Temporal block** – This block then captures the temporal features. Its inputs include the k spatial features and $\mathbf{V}'$. It aims to obtain the long-temporal relationship of the past and future data. Specifically, we propose to integrate a memory-based sequence-to-sequence model within an encoder-decoder architecture (Sutskever et al. 2014) for extracting the long-temporal dependency of the traffic speed. The output of this block is the final prediction.

5.2.3 Input Block

The two input data are the past traffic speed measurements and the road network graph data. The traffic speed data is $\mathbf{V} = \{\mathrm{v_{t-T+1}}, \mathrm{v_{t-T+2}}, \ldots, \mathrm{v_t}\}; \mathbf{V} \in \mathbb{R}^{T \times N \times B \times \mathcal{F}}$ where B and $\mathcal{F}$ represent the batch size and the number of considered traffic features respectively. In this chapter, without loss of generality, traffic feature only consists of traffic speed. Therefore, $\mathcal{F} = 1$.

The road network graph data refers to the $k-hop$ neighbourhood matrices, $\{\tilde{\mathbf{A}}^{\mathbf{1}}, \tilde{\mathbf{A}}^{\mathbf{2}}, \ldots, \tilde{\mathbf{A}}^{\mathbf{k}}\}$. Since we only need the connectivity information of nodes within the neighbourhood rather than the actual hop distance to the neighbours, we follow (Cui et al. 2019) and clip all elements in $\tilde{\mathbf{A}}^{\mathbf{k}}$ to be within {0,1}. We can then rewrite the $k-hop$ neighbourhood matrix $\tilde{A}^k$ hereafter as follows:

$$\tilde{\mathbf{A}}^{\mathbf{k}} = Ci(\tilde{\mathbf{A}}^{\mathbf{k}}) \tag{5.2}$$

where $Ci(.)$ is the clip function such that $\tilde{A}^k_{i,j} = \min\left(\tilde{A}^k_{i,j}, 1\right)$. Note that when $k = 1$, $\tilde{\mathbf{A}}^{\mathbf{1}} = \tilde{\mathbf{A}}$ reverts back to the adjacency matrix itself describing the connectivity relationship of nodes in the $1-hop$ neighbourhood. All spatial features from $k-hop$ neighbourhoods are concatenated as a matrix vector $\mathbf{SAGCN} \in \mathbb{R}^{B \times T \times N \times k}$.

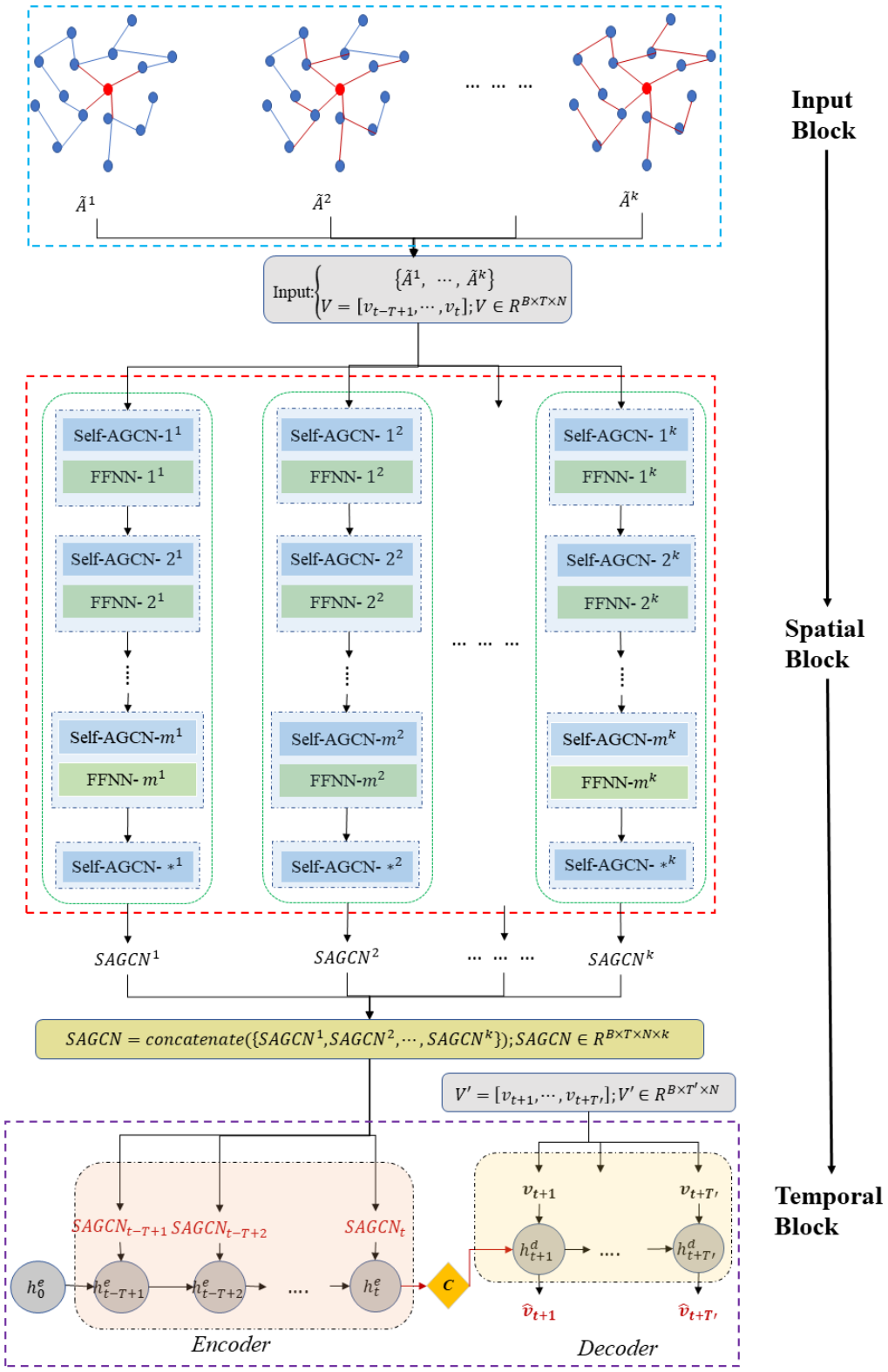


Figure 5.1: Our novel deep learning framework, SAGCN-SST.

5.2.4 The Spatial Block

The spatial block is composed of k parallel sub-spatial blocks corresponding to k different neighbourhoods (see Figure 5.1). Each sub-spatial block consists of m graph convolutional blocks (GCN blocks) where each GCN block comprises a graph convolutional layer with self-attention mechanism (Veličković et al. 2018) (namely Self-AGCN) and a feed forward neural network (FFNN) layer. The last GCN block is followed by an additional Self-AGCN layer. At the end of parallel sub-spatial blocks, the k spatial features (i.e., $\mathbf{SAGCN^1}$, $\mathbf{SAGCN^2}$, ..., $\mathbf{SAGCN^k}$) corresponding to k different neighbourhoods are passed to the temporal block for extracting temporal features.

Figure 5.2 illustrates the working principle of the m^{th} Self-AGCN layer on the $k-hop$ neighbourhood where it has n_k nodes. For example, traffic speed of node 1 in Figure 5.2, as the targeted node, is affected by other $(n_k - 1)$ nodes differently. As such, we need to quantify and compute the different weights of these neighbours with respect to the targeted node (i.e., $(a_{1-1}, a_{1-2}, \ldots, a_{1-n_k})$ (namely, attention weights)). To achieve this, the convolutional operation (i.e., Eq. (5.3)) is first conducted on the graph of the road network for fixed spatial feature extraction:

$$\mathrm{GC}_{\mathrm{t}}^{\mathrm{m;k}} = \left(\mathbf{W}_{\mathbf{gc}_{\mathbf{t}}^{\mathbf{m;k}}} * \tilde{\mathbf{A}}^{\mathbf{k}}\right) \mathrm{v}_{\mathrm{t}} \tag{5.3}$$

where $*$ is the Hadamard product operator, $\mathrm{v}_{\mathrm{t}} \in \mathbb{R}^N$ is the traffic speed at the t^{th} time interval. $\mathbf{W}_{\mathbf{gc}_{\mathbf{t}}^{\mathbf{m;k}}} \in \mathbb{R}^{N \times N}$ is a trainable weight matrix in the m^{th} Self-AGCN layer on the $k-hop$ neighbourhood. The output matrix $\mathrm{GC}_{\mathrm{t}}^{\mathrm{m;k}} \in \mathbb{R}^N$ represents the fixed spatial features at current time interval.

As prior mentioned, each neighbouring node contributes differently to the future traffic status of a targeted node due to its distance as well as its own spatial neighbourhood in relation with the targeted node. Intuitively, traffic status of a targeted node within a road network is more heavily influenced by their immediate adjacent neighbours and less affected by nodes further away. However, this is not strictly so. Moreover, traffic volume in a road network also affects the influence of neighbouring nodes. For instance, neighbouring nodes may not have strong influence on the targeted node in a relatively quiet road network with low traffic flow. Conversely, in a congestion-prone road network, the traffic status of neighbouring nodes will have impact on the future status of the targeted node. In fact, neighbouring nodes will have increasing influence as the congestion period lengthens. To extract such dynamic spatial features, we apply the self-attention mechanism at each GCN block to compute the contribution of each node in the $k-hop$ neighbourhood

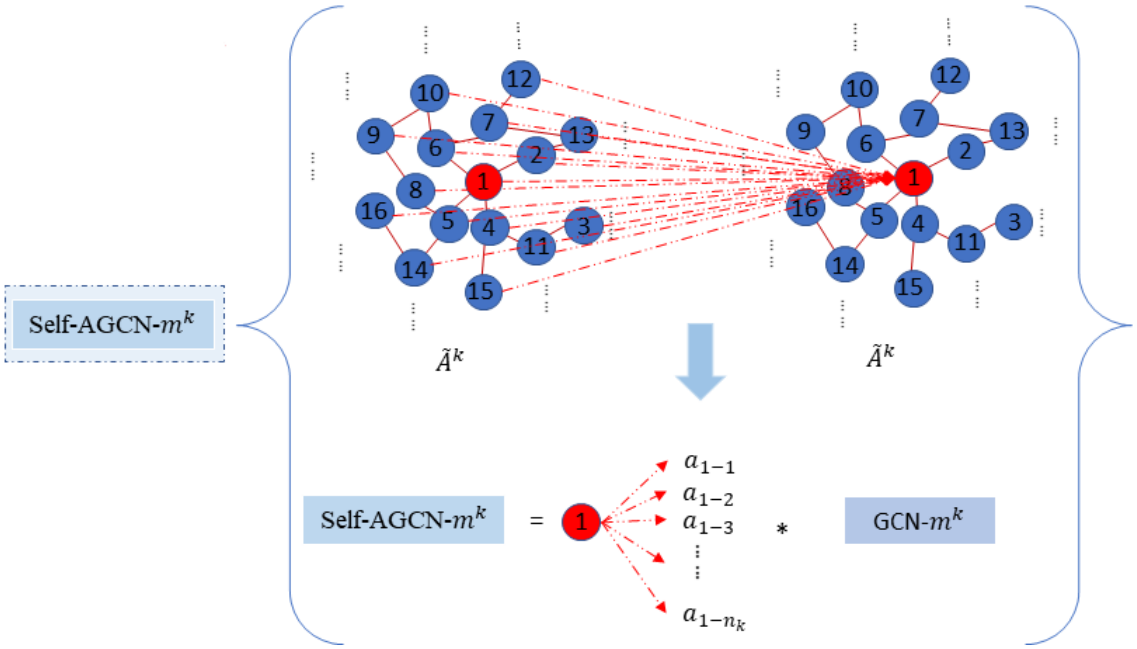


Figure 5.2: The graph convolutional layer with self-attention mechanism where the contribution of each neighbouring node to the future traffic status of the targeted node 1 is computed and represented via an attention weight.

and assign a weight to these neighbouring nodes. The weight is computed based on the similarity between the neighbouring node and the targeted node. Specifically, the similarity of traffic data between two nodes i and j, $\mathbf{u_t^{m;k}}(\mathbf{i},\mathbf{j}) \in \mathbb{R}^{N\times n_k}$, is computed via a $tanh$ function as follows:

$$\mathbf{u_t^{m;k}}(\mathbf{i},\mathbf{j}) = q^T tanh\left(\mathrm{GC_t^{m;k}(i)}\mathbf{W_f^{m;k}}\mathrm{GC_t^{m;k}(j)}\right); \quad j = 1, 2, \ldots, n_k \tag{5.4}$$

where $\mathbf{W_f^{m;k}} \in \mathbb{R}^{N\times N}$ is a trainable weight matrix and q^T represents the transposition or reshaping operations that are utilised to adjust the dimensions. We then compute the attention weights as probabilities (i.e., $\mathbf{a_t^{m;k}}(\mathbf{i},\mathbf{j}) \in [0.0, 1.0]$) via a $softmax$ function given in Eq. (5.5).

$$\begin{aligned}\mathbf{a_t^{m;k}}(\mathbf{i},\mathbf{j}) &= softmax\Big(\mathbf{u_t^{m;k}}(\mathbf{i},\mathbf{j})\Big) \\ &= \frac{exp\Big(\mathbf{u_t^{m;k}}(\mathbf{i},\mathbf{j})\Big)}{\sum_{j=1}^{n_k} exp\Big(\mathbf{u_t^{m;k}}(\mathbf{i},\mathbf{j})\Big)}\end{aligned} \tag{5.5}$$

After obtaining the attention weights $\mathbf{a_t^{m;k}}(\mathbf{i},\mathbf{j}) \in \mathbb{R}^{N\times n_k}$, it is used to map to fixed spatial traffic feature $\mathrm{GC_t^{m;k}(i)}$ for achieving dynamic spatial traffic feature $\mathrm{SGC_t^{m;k}} \in \mathbb{R}^N$.

$$\mathrm{SGC_t^{m;k}} = \sum_{j=1}^{n_k} \mathbf{a_t^{m;k}}(\mathbf{i},\mathbf{j})\mathrm{GC_t^{m;k}(i)} \tag{5.6}$$

Each Self-AGCN layer is followed by a FFNN layer for improving the prediction ability on learned traffic features. This layer consists of a $tanh$ layer as given in Eq. (5.7) and a $dropout$ layer as given in Eq. (5.8).

$$\mathrm{SGC_{tanh;t}^{m;k}} = tanh(\mathrm{SGC_t^{m;k}}) \tag{5.7}$$

$$\mathrm{SGC_{drop;t}^{m;k}} = dropout(\mathrm{SGC_{tanh;t}^{m;k}}) \tag{5.8}$$

Compared to (Kipf & Welling 2017) which uses a $ReLU$ layer and a $dropout$ layer, we propose to use a $tanh$ layer in place of the $ReLU$ layer. The main reason is that $ReLU$ function de-activates negative values and only retains positive values. As such, it may miss some important information hidden behind negative values.

5.2.5 The Temporal Block

For temporal feature extraction, we propose to use the sequence-to-sequence architecture (Sutskever et al. 2014) which has already been found to offer good performance in the area of natural language processing. The architecture consists of an encoder and a decoder with a context $\mathbf{C} \in \mathbb{R}^{B\times N}$ connecting the two (see Figure 5.1).

The encoder takes the $\mathbf{SAGCN}$ produced by the spatial block as the input. It encodes the spatially-fused time series using the following:

$$\mathbf{h}^{\mathbf{e}}_{\mathbf{t-t_e}} = \begin{cases} f_{encoder}(\mathbf{h}^{\mathbf{e}}_{\mathbf{0}}, \mathbf{SAGCN}_{\mathbf{t-t_e}}), & t_e = T \\ f_{encoder}(\mathbf{h}^{\mathbf{e}}_{\mathbf{t-t_e-1}}, \mathbf{SAGCN}_{\mathbf{t-t_e}}), & t_e \in 0, \ldots, T-1 \end{cases} \tag{5.9}$$

where $\mathbf{h}^{\mathbf{e}}_{\mathbf{t-t_e}} \in \mathbb{R}^{B \times N}$ is the hidden state in the encoder at $(t - t_e)^{th}$ time interval. The initial hidden state is $\mathbf{h}^{\mathbf{e}}_{\mathbf{0}}$. The hidden state $\mathbf{h}^{\mathbf{e}}_{\mathbf{t-t_e-1}} \in \mathbb{R}^{B \times N}$ at $(t - t_e - 1)^{th}$ time interval and the spatially-fused feature $\mathbf{SAGCN}_{\mathbf{t-t_e}} \in \mathbb{R}^{N \times k}$ at $(t - t_e)^{th}$ time interval are used to calculate the hidden state $\mathbf{h}^{\mathbf{e}}_{\mathbf{t-t_e}}$ at the $(t - t_e)^{th}$ time interval.

The hidden state $\mathbf{h}^{\mathbf{e}}_{\mathbf{t}}$ ($t_e = 0$) at the t^{th} time interval is the context vector $\mathbf{C}$ which encodes all information from the input $\mathbf{SAGCN}$ in the encoder.

$$\mathbf{C} = \mathbf{h}^{\mathbf{e}}_{\mathbf{t}} \tag{5.10}$$

In the decoder, the context vector $\mathbf{C}$ as the initial hidden state $\mathbf{h}^{\mathbf{d}}_{\mathbf{0}} \in \mathbb{R}^{B \times N}$ is decoded to the target sequence. The hidden state $\mathbf{h}^{\mathbf{d}}_{\mathbf{t+t_d-1}}$ at $(t + t_d - 1)^{th}$ time interval and the target traffic speed $\mathrm{v}_{\mathrm{t+t_d}}$ at $(t + t_d)^{th}$ time interval are utilised to calculate the hidden state $\mathbf{h}^{\mathbf{d}}_{\mathbf{t+t_d}}$ at the $(t + t_d)^{th}$ time interval. The hidden state $\mathbf{h}^{\mathbf{d}}_{\mathbf{t+t_d}}$ at the $(t + t_d)^{th}$ time interval in the decoder is the final prediction $\tilde{\mathrm{v}}_{\mathrm{t+t_d}}$.

The $f_{encoder}$ and $f_{decoder}$ functions are two GRU modules (Chung et al. 2014). While GRU is based on LSTM, it incurs shorter processing time and less central processing unit (CPU) cycles. The reason is that GRU combines LSTM's forget and input gates into a single "update gate", and also merges the memory cell and hidden state. This makes GRU simpler than the standard LSTM but still efficient. Figure 5.3 depicts the working process and data flow in a recycled unit of the GRU module. GRU consists of three parts: the update gate $\mathbf{z_t}$, the reset gate $\mathbf{r_t}$ and the hidden state $\mathbf{h}^{\mathbf{e}}_{\mathbf{t}}$. The update gate, $\mathbf{z_t}$, extracts the long-term dependency of the data. It decides how much information it needs to update from the input $\mathbf{SAGCN_t}$ and the hidden state at the previous time interval $\mathbf{h}^{\mathbf{e}}_{\mathbf{t-1}}$ (see Eq. (5.11)).

$$\mathbf{z_t} = \sigma(\mathbf{W_z} \times \mathbf{SAGCN_t} + \mathbf{U_z} \times \mathbf{h}^{\mathbf{e}}_{\mathbf{t-1}} + \mathrm{b_z}) \tag{5.11}$$

The reset gate, $\mathbf{r_t}$, captures the short-term dependency of traffic features. It decides how much information from the hidden state at the previous time interval is retained for updating the current hidden state. It is computed in a similar manner as the update gate by Eq. (5.12).

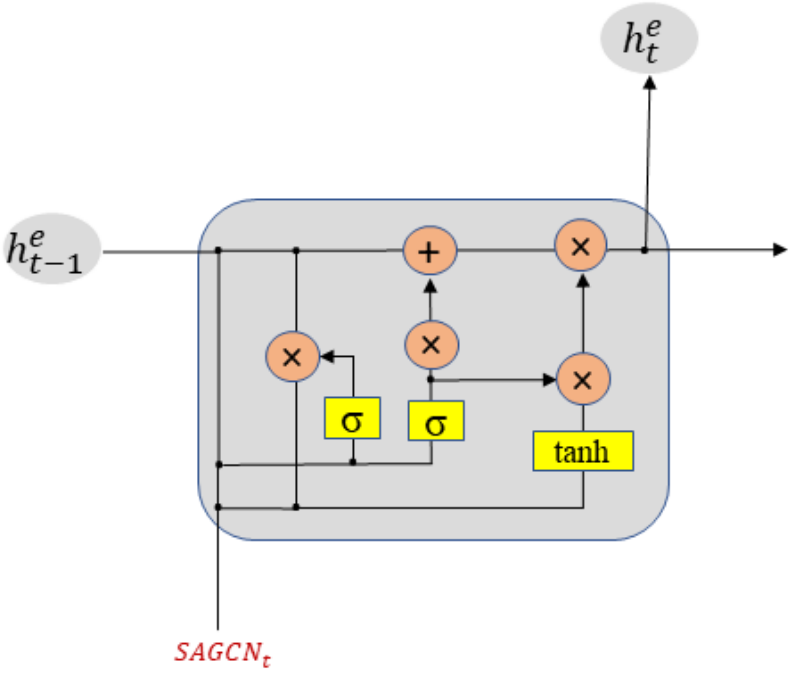


Figure 5.3: Gated Recurrent Unit

$$\mathbf{r_t} = \sigma(\mathbf{W_r} \times \mathbf{SAGCN_t} + \mathbf{U_r} \times \mathbf{h_{t-1}^e} + \mathrm{b_r}) \tag{5.12}$$

Then, the input $\mathbf{SAGCN_t}$, the reset gate $\mathbf{r_t}$ and the hidden state at the previous time interval $\mathbf{h_{t-1}^e}$ are used to activate the candidate hidden state $\tilde{\mathbf{h}}_\mathbf{t}^\mathbf{e}$ via Eq. (5.13).

$$\tilde{\mathbf{h}}_\mathbf{t}^\mathbf{e} = tanh(\mathbf{W_h} \times \mathbf{SAGCN_t} + \mathbf{U_h} \times (\mathbf{r_t} * \mathbf{h_{t-1}^e}) + \mathrm{b_h}) \tag{5.13}$$

where $\mathbf{W_z}$, $\mathbf{W_r}$ and $\mathbf{W_h}$ are the weights of the update gate, the reset gate and the candidate hidden state respectively while $\mathrm{b_z}$, $\mathrm{b_r}$ and $\mathrm{b_h}$ are the corresponding bias for each gate and state. Furthermore, $\mathbf{U_z}$, $\mathbf{U_r}$ and $\mathbf{U_h}$ are the weights of the hidden state at the previous time interval $\mathbf{h_{t-1}^e}$ in the update gate, the reset gate and the candidate hidden state, respectively. Finally, the current hidden state can be calculated using the update gate $\mathbf{z_t}$, the hidden state at the previous time interval $\mathbf{h_{t-1}^e}$ and the current candidate hidden state $\tilde{\mathbf{h}}_\mathbf{t}^\mathbf{e}$ using Eq. (5.14).

$$\mathbf{h_t^e} = (1 - \mathbf{z_t}) * \mathbf{h_{t-1}^e} + \mathbf{z_t} * \tilde{\mathbf{h}}_\mathbf{t}^\mathbf{e} \tag{5.14}$$

5.2.6 *Loss* Function

To train our SAGCN-SST model, the RMSprop optimiser (Tieleman & Hinton 2012) is used to minimise the error between the real and predicted traffic. It restricts the oscillations in the vertical direction. The learning rate can be adjusted to take larger steps in the horizontal direction for faster convergence compared to Gradient Descent Optimiser (Bottou 2010).

Mean Square Error (MSE) in Eq. (5.15) as loss function is adopted to train our model as it learns faster than Mean Absolute Error (MAE).

$$Loss = L(\mathrm{v_t}, \tilde{\mathrm{v}}_\mathrm{t}) = \frac{1}{N}\sum_{i=1}^{N}\left(v_t^i - \tilde{v}_t^i\right)^2 \tag{5.15}$$

where $L(.)$ is the MSE loss function. It calculates the residual error between the real traffic data v_t^i and the predicted traffic data $\tilde{v}_t^i$.

5.3 Data Description

To train and test our proposed framework, two real-world datasets from large-scale road networks are utilised: hereafter labelled as `LOOP-SEATTLE` (Cui et al. 2019) and `METR-LA` (Li et al. 2018). The detailed information about those two datasets can be found in Chapter 2.6.2.

5.4 Experiments

5.4.1 Parameter settings

In our SAGCN-SST model, there are several parameters that need to be set for achieving accurate predictions. The parameters mainly relate to the training process. Specifically, the parameters are the learning rate r, batch size B, observed time intervals T, targeted time intervals T' and the number of epochs. We follow (Zang et al. 2018, Cui et al. 2019) and set the learning rate r, batch size B and observed time intervals T as 10^{-3}, 40 and 10 respectively. As prior mentioned, since we are addressing multi-step traffic prediction

problem, we set the targeted time intervals T' as 1, 3, 6, 9 and 12, corresponding to 5, 15, 30, 45 and 60 mins as prediction horizons respectively (Li et al. 2018, Yu et al. 2018). To find the number of epochs, we use *stop early* strategy in which the training process will be stopped when the training loss continues to decrease in 10 epochs while the validation loss increases. This avoids the problem of over-fitting.

Finally, we follow the convention and use 70% of the data for training, 20% for validation and 10% for testing. All experiments are conducted on a GeForce GTX 1080 Ti GPU with 11 GB physical memory, and Pytorch is used to program this work.

5.4.2 Performance Evaluation of SAGCN-SST

To evaluate our proposed SAGCN-SST, we first conduct experiments with different number of GCN blocks (within the $m = [1..6]$ interval) for $T' = 1$ (i.e., prediction for 5 mins in advance) for the $1 - hop$ neighborhood (i.e., $k = 1$). Table 5.1 presents the prediction results achieved by our SAGCN-SST model for both `LOOP-SEATTLE` and `METRA-LA` datasets.

Table 5.1: Results of SAGCN-SST with $m = [1..6]$ GCN blocks for `LOOP-SEATTLE` and `METR-LA`

	`LOOP-SEATTLE` ($T' = 1$)			`METR-LA` ($T' = 1$)		
Layers	MAE	MAPE	RMSE	MAE	MAPE	RMSE
$m = 1$	0.8435	1.98	1.1175	0.9858	2.05	1.4051
$m = 2$	0.8186	1.91	1.0851	0.9672	2.01	1.3638
$m = 3$	0.8031	1.89	1.0686	0.9134	1.90	1.2894
$m = 4$	0.8181	1.92	1.0872	0.9499	1.98	1.3471
$m = 5$	0.8219	1.96	1.2140	0.9353	1.93	1.3209
$m = 6$	0.8102	1.89	1.0759	0.9340	1.93	1.3144

For `LOOP-SEATTLE`, our SAGCN-SST model achieves MAPE below 2% for m between 1 and 6. The lowest MAPE is achieved (i.e., 1.89%) when SAGCN-SST has 3 and 6 GCN blocks. In these two cases (i.e., $m = 3$ and $m = 6$), the errors in MAE and RMSE for $m = 3$ are lower. Furthermore, considering that having a deeper model with more GCN blocks costs more in terms of physical memory and GPU cycles, incurs longer running time, and may even result in over-fitting, we set $m = 3$ in the ensuing experiments for `LOOP-SEATTLE`. For `METR-LA`, the best result (i.e., MAPE = 1.90%) is achieved

when m is also set as 3. In addition, the average MAE and RMSE for both datasets are respectively below 1.00 and 1.50. Compared to results achieved from LOOP−SEATTLE, these three types of errors obtained from METR−LA are larger. This indicates that traffic pattern in METR−LA is more complex.

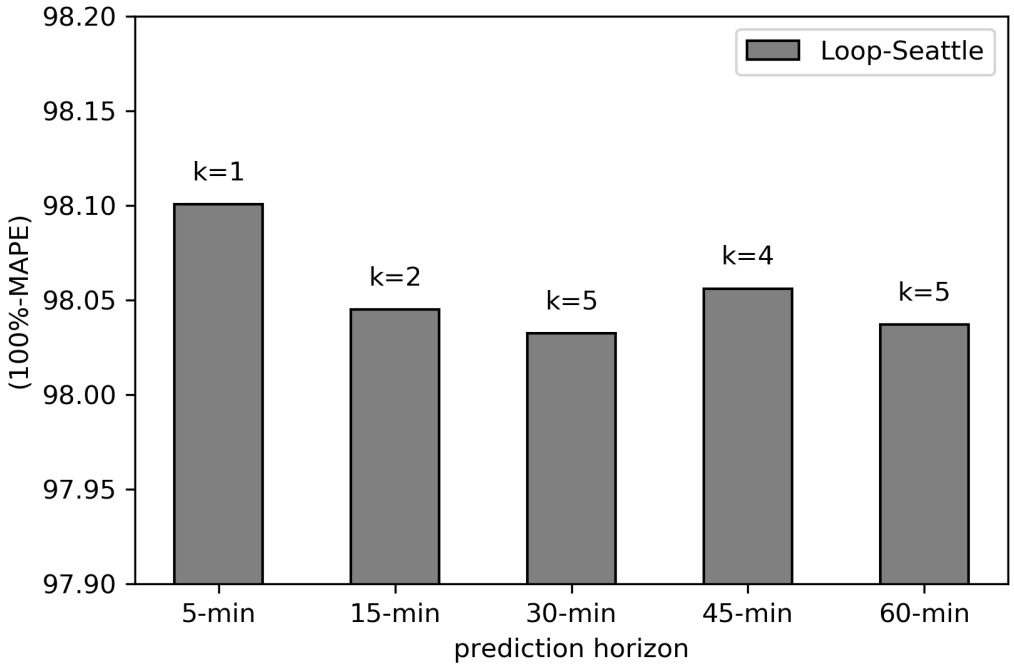
Next, we proceed with experiments for multi-step traffic speed prediction with prediction horizon, $T' = \{1, 3, 6, 9, 12\}$ with different neighbourhoods (i.e., $k = [1..5]$) on both datasets. The results are summarised in Table 5.2. For LOOP−SEATTLE, our SAGCN-SST consistently achieves MAPE less than 2% for the different $k - hop$ neighbourhoods across all prediction horizons. This indicates that longer prediction horizons (i.e., $T' = \{6, 9, 12\}$) do not affect SAGCN-SST's prediction performance. This is mainly due to our design which integrated the sequence-to-sequence architecture in our temporal block that effectively captures long-temporal dependency of the data. Figure 5.4 (a) presents the best prediction results achieved on LOOP−SEATTLE for different neighbourhoods (i.e., $k = \{1, 2, 3, 4, 5\}$) over different prediction horizons ($T' = \{1, 3, 6, 9, 12\}$). It is clear that short-term predictions mainly depend on adjacent neighbours while long-term predictions need to take into account the influence on a wider neighbourhood. Specifically, for $T' =$ 1 and 3, the best results are obtained when k is equal to 1 and 2, respectively. In contrast, for $T' = \{6, 9, 12\}$, the best performances are achieved when k is higher (i.e., when $k =$ 5, 4 and 5, respectively).

The observations are quite different for the METR−LA dataset. The best predictions are always obtained when $k = 1$ (see Figure 5.4(b)). This is due to the different nature of the traffic patterns in the two datasets. Specifically, we observe that congestion duration in LOOP−SEATTLE tends to be significantly longer than that recorded in METR−LA. We use Figure 5.5 and Figure 5.6 to exemplify this. The figures show the real (black solid line) and predicted (red dashed line) traffic speed from two randomly selected detectors retrieved from LOOP−SEATTLE (cf. Figure 5.5) and METR−LA (cf. Figure 5.6), respectively. From figures, we see that the traffic congestion duration is 120 mins ($= (150 - 126) \times 5$) on METR−LA while for LOOP−SEATTLE, the congestion lasts for 330 mins ($= (181 - 115) \times 5$). From these observations from the two datasets, we can see that higher k (i.e., considering wider neighbourhood) offers better prediction accuracy for traffic congestion that tends to last longer (i.e., long-term prediction) and vice versa.

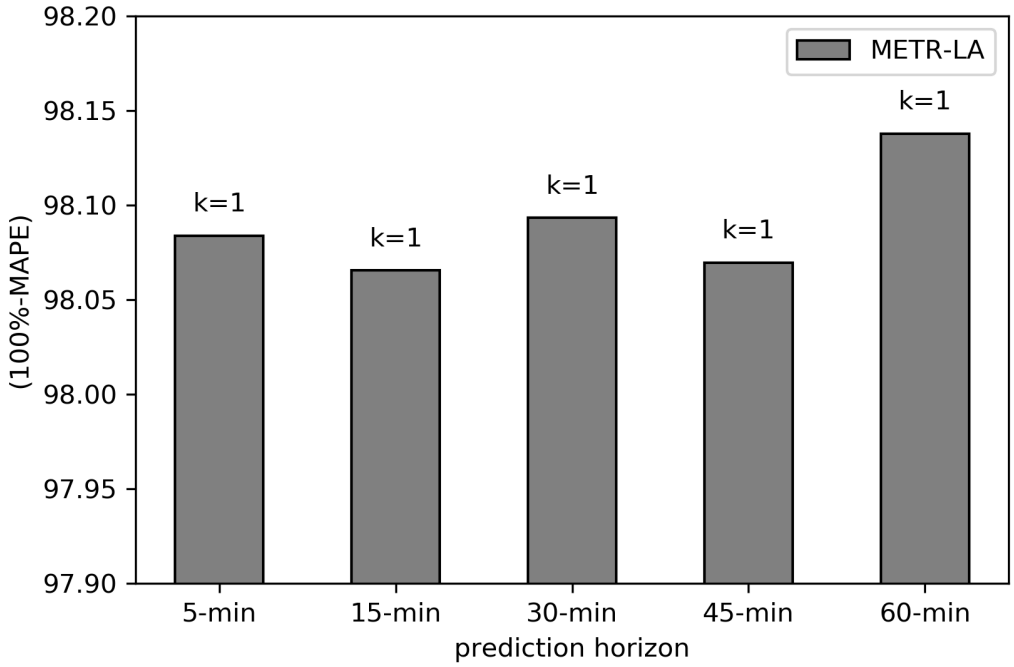
In Figure 5.5 and Figure 5.6, prediction horizon T' increases in the range $\{1, 3, 6, 9, 12\}$ from top to bottom. From the figure, it is clear that our model is able to accurately predict traffic speed across the entire duration including during peak hours (traffic speed is lower) and off-peak hours (traffic speed is higher). Specifically for more challenging tasks $T' = \{6, 9, 12\}$ compared to $T' = \{1, 3\}$, SAGCN-SST can not only accurately

Table 5.2: Results of SAGCN-SST on $k-hop$ neighbourhoods for both datasets

Model Name	LOOP-SEATTLE			METR-LA		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
(a) 5-min future prediction (T'=1)						
k=1	0.8156	1.90	1.0802	0.9256	1.92	1.3090
k=2	1.0562	2.55	1.4301	1.0494	2.18	1.5103
k=3	0.8320	1.96	1.1041	1.0091	2.11	1.4417
k=4	0.8021	1.90	1.0735	1.0706	2.23	1.5290
k=5	0.8277	1.94	1.1012	1.0375	2.14	1.4714
(b) 15-min future prediction (T'=3)						
k=1	0.8627	2.03	1.1528	0.9392	1.93	1.3260
k=2	0.8280	1.96	1.0982	1.0085	2.12	1.4313
k=3	0.8702	2.08	1.1692	1.0619	2.21	1.5153
k=4	0.8453	2.02	1.1300	1.0295	2.12	1.4467
k=5	0.8643	2.05	1.1540	1.0468	2.16	1.4894
(c) 30-min future prediction (T'=6)						
k=1	0.8500	2.02	1.1310	0.9242	1.91	1.3049
k=2	0.8767	2.08	1.1585	1.0471	2.19	1.5045
k=3	0.8745	2.07	1.1618	1.0023	2.08	1.4239
k=4	0.8651	2.04	1.1469	1.0488	2.17	1.4849
k=5	0.8332	1.97	1.1147	1.0577	2.21	1.5047
(d) 45-min future prediction (T'=9)						
k=1	0.8358	1.99	1.1197	0.9198	1.93	1.2976
k=2	0.8432	2.01	1.1287	1.0640	2.20	1.5159
k=3	0.8507	2.01	1.1332	1.0569	2.20	1.5159
k=4	0.8221	1.94	1.0921	1.0256	2.12	1.4603
k=5	0.8344	1.96	1.1082	1.0671	2.25	1.5244
(e) 60-min future prediction (T'=12)						
k=1	0.8613	2.03	1.1459	0.9033	1.86	1.2701
k=2	0.8577	2.03	1.1422	1.0399	2.19	1.4726
k=3	0.8382	2.00	1.1129	1.0510	2.19	1.4896
k=4	0.8271	1.97	1.1065	1.0671	2.25	1.5182
k=5	0.8266	1.96	1.0985	0.9890	2.04	1.3956



(a)



(b)

Figure 5.4: Relationship of $k-hop$ neighbourhoods and the prediction horizon with accuracy from SAGCN-SST model on `LOOP-SEATTLE` (a) and `METR-LA` (b). The x-axis represents the prediction horizon and the y-axis is the (100%-MAPE) which means the prediction accuracy.

follow the overall trends but also capture the details of rapid fluctuations.

Figure 5.7 and Figure 5.8 visualise the real (sub-figure (a)) and predicted (sub-figure (b)) traffic speed at a randomly selected time interval from our SAGCN-SST model on the road network of `LOOP-SEATTLE` and `METR-LA`, respectively. From the figures, it can be observed that the real traffic on the road network are closely predicted across the entire map and thus, further validating the capability of our SAGCN-SST model in computing accurate predictions on large-scale road networks. Furthermore, both Figure 5.7 and Figure 5.8 also show that very low or high traffic speed are recorded on several continuous detectors for both road networks. It indicates that traffic state recorded by a detector is influenced by its neighbours. This information should be considered in traffic prediction models. Our SAGCN-SST, that defines $k - hop$ neighbourhoods for each detector and analyses spatial features from its neighbourhood, captures exactly this information to achieve accurate predictions.

Let the residuals of traffic speed predictions be defined as $(v_t^i - \tilde{v}_t^i)$. Considering that the residual as an indicator on whether the results of a model are statistically correct, we show in Figure 5.9 and Figure 5.10 the residuals of traffic speed predictions by our proposed model on both datasets. For both datasets with 5 mins as prediction horizon (top row of the figure), we see that the residual distributions follow normal distributions with zero means. For longer prediction horizons (i.e., 30 and 60 mins), while the residual distributions still resemble that of a normal distribution, the means shift away from zero. This is because longer prediction horizons are less impacted by historical traffic data compared to short prediction horizons. The residual's normal distributions in Figure 5.9 and Figure 5.10 again suggests that our proposed model is capable of capturing dynamic spatial-temporal features and provide more accurate predictions.

5.4.3 Comparison Study

We compare our proposed model, SAGCN-SST, to well-known existing models in recent literature. The chosen representative models from the state-of-the-art adopt one of four different approaches to treat traffic prediction problem. These four approaches respectively consider traffic prediction as 1) a temporal, 2) a spatial, 3) a spatial-temporal and 4) a fixed spatial dynamic-temporal process. The seven models that are used for our comparison study based on the aforementioned approaches are the following:

1. Gate Recurrent Unit (GRU) (Chung et al. 2014): See Section 5.2.5 for details.

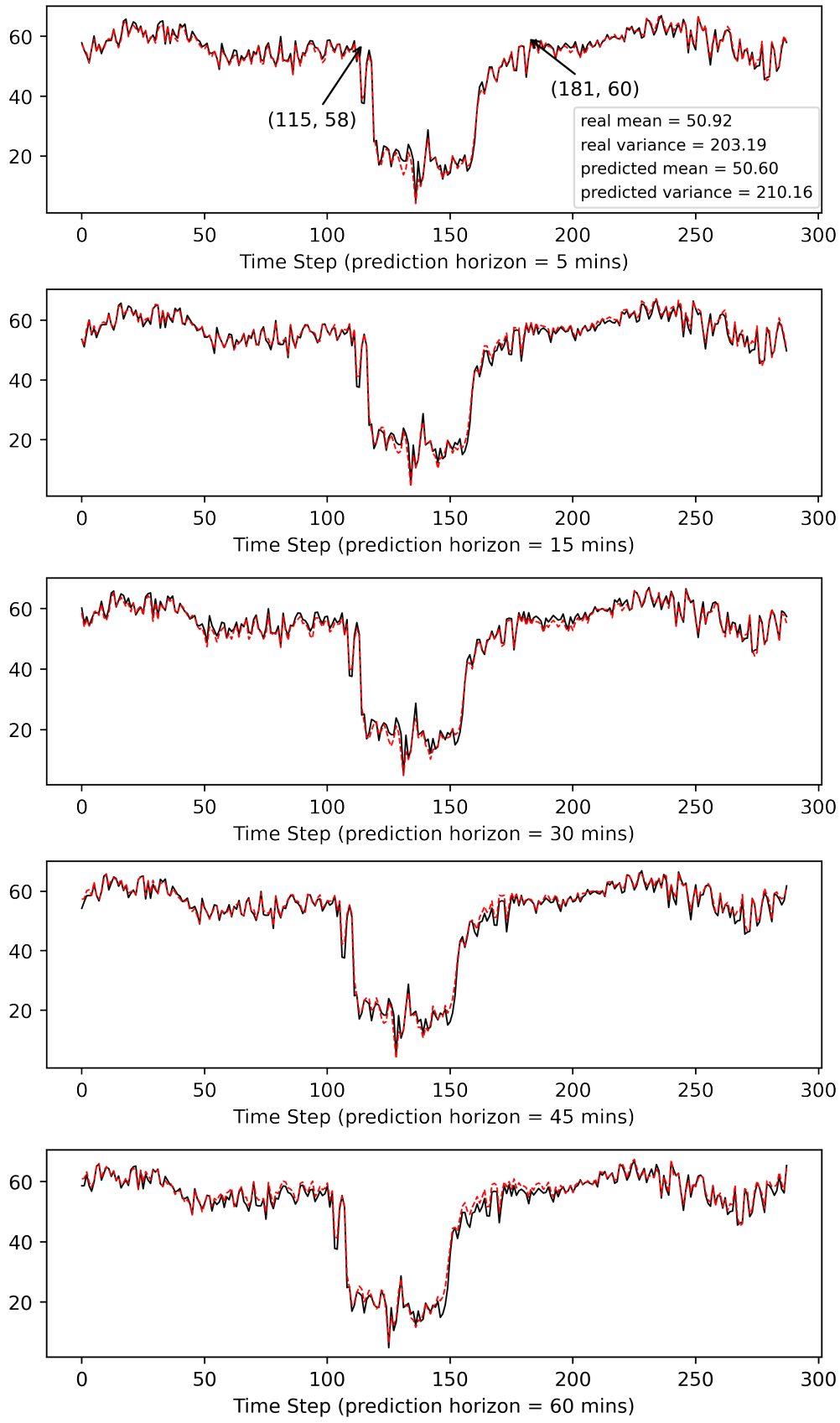


Figure 5.5: Real and predicted traffic speed $(miles/hour)$ in a day with 288 $(= \frac{24h*60mins}{5mins})$ time intervals from SAGCN-SST on LOOP-SEATTLE with a time interval = 5 mins.

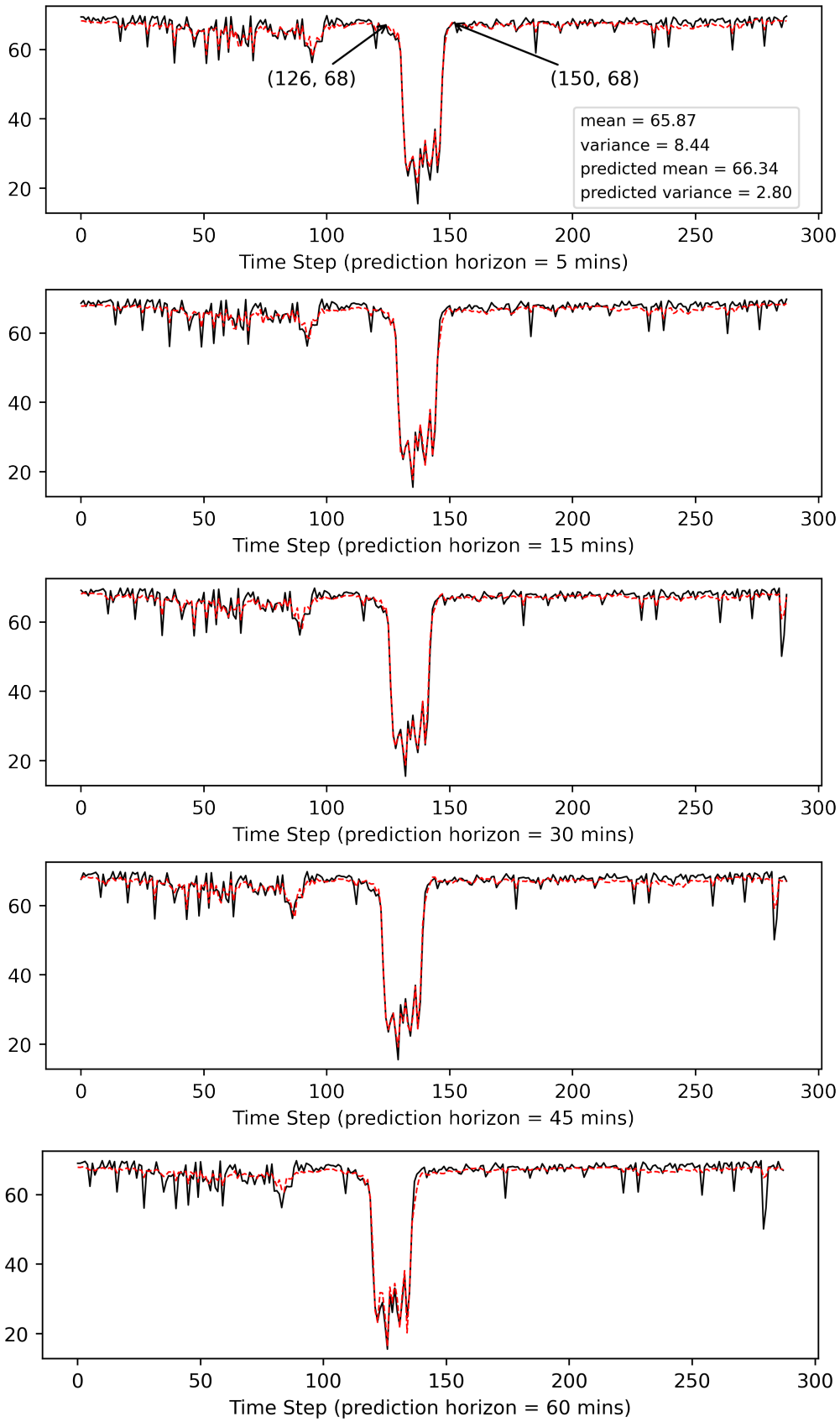
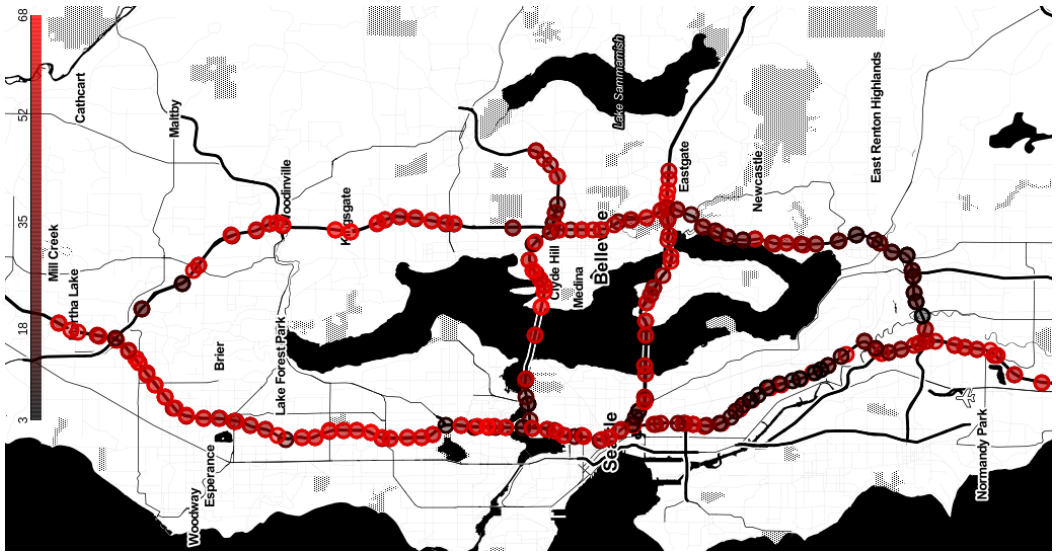
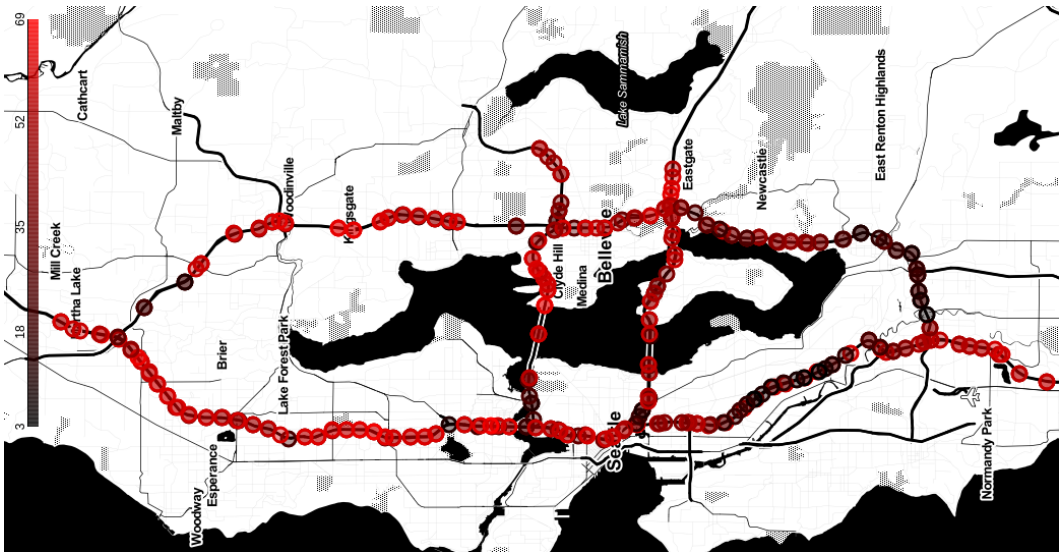


Figure 5.6: Real and predicted traffic speed $(miles/hour)$ in a day with 288 $(= \frac{24h*60mins}{5mins})$ time intervals from SAGCN-SST on METR-LA with a time interval = 5 mins.

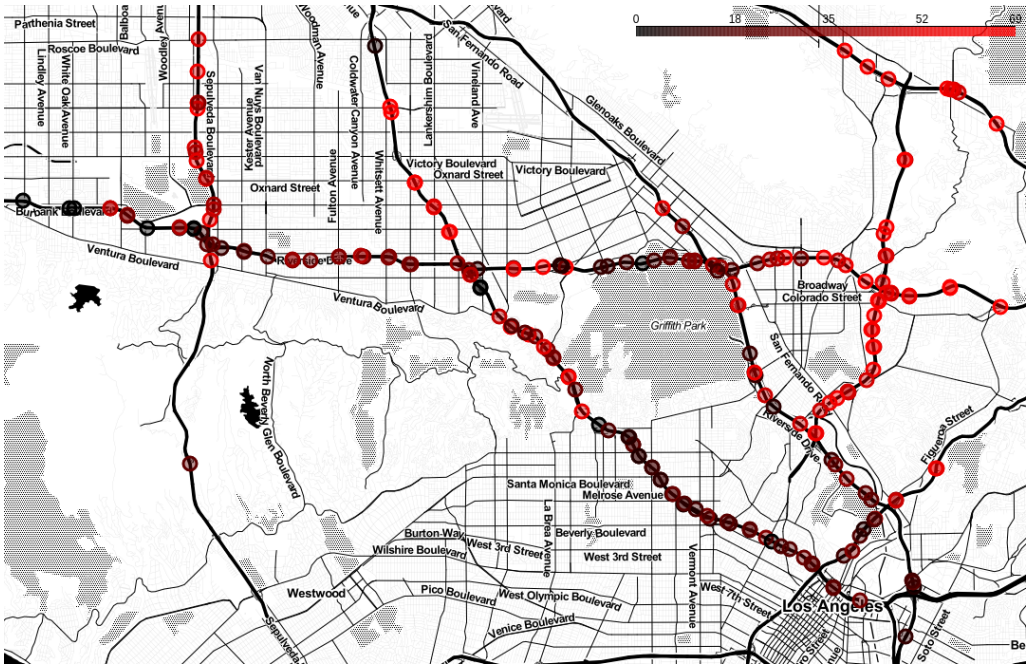


(a)

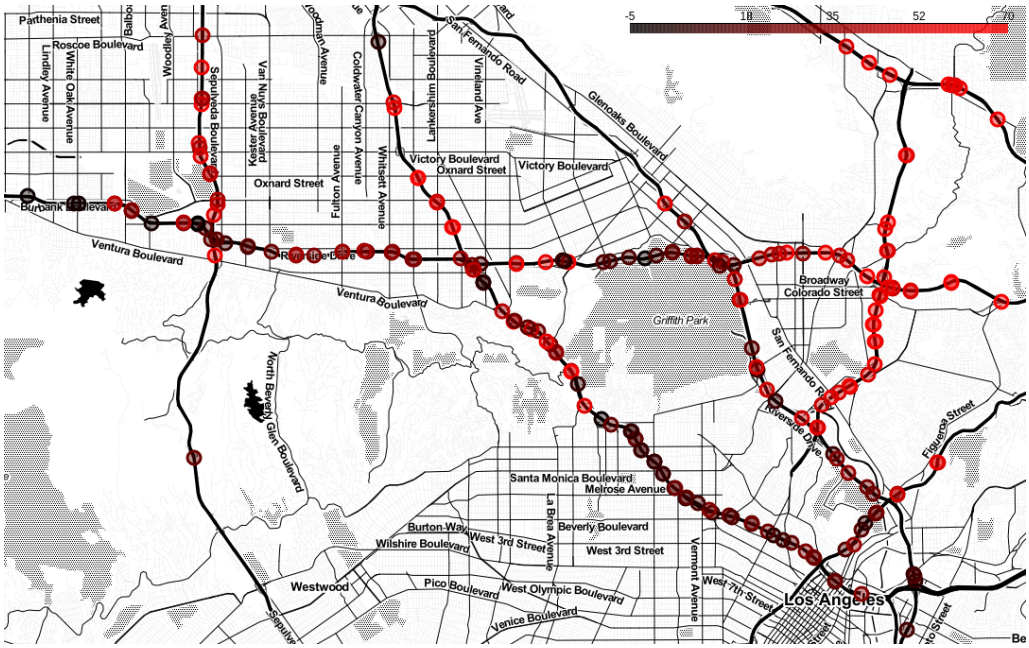


(b)

Figure 5.7: Visualisation of real (a) and predicted (b) traffic speed at a randomly selected time interval on the road network of `LOOP-SEATTLE`. Darker colour represent lower traffic speed.

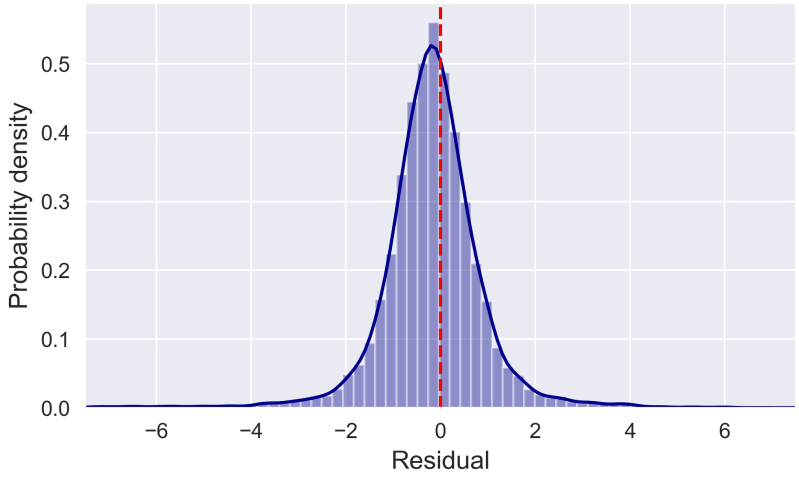


(a)

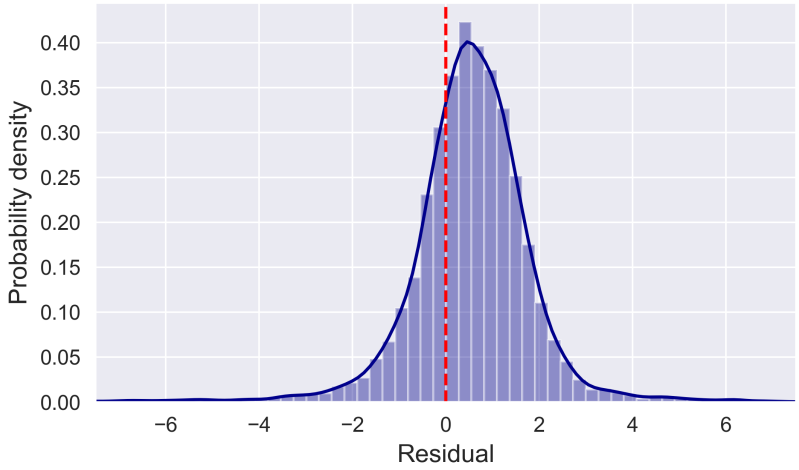


(b)

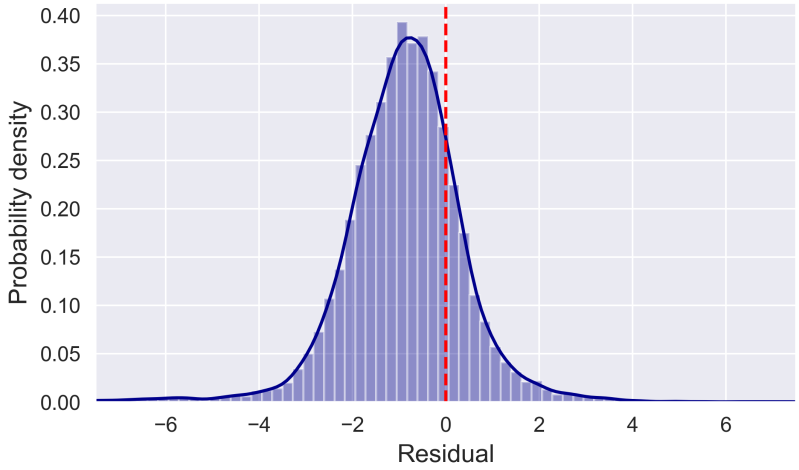
Figure 5.8: Visualisation of real (a) and predicted (b) traffic speed at a randomly selected time interval on the road network of METR-LA. Darker colour represent lower traffic speed.



prediction horizon = 5 mins

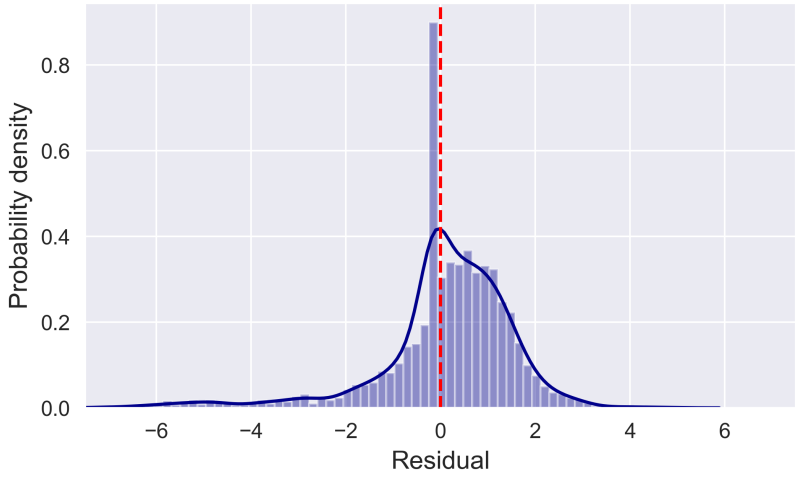


prediction horizon = 30 mins

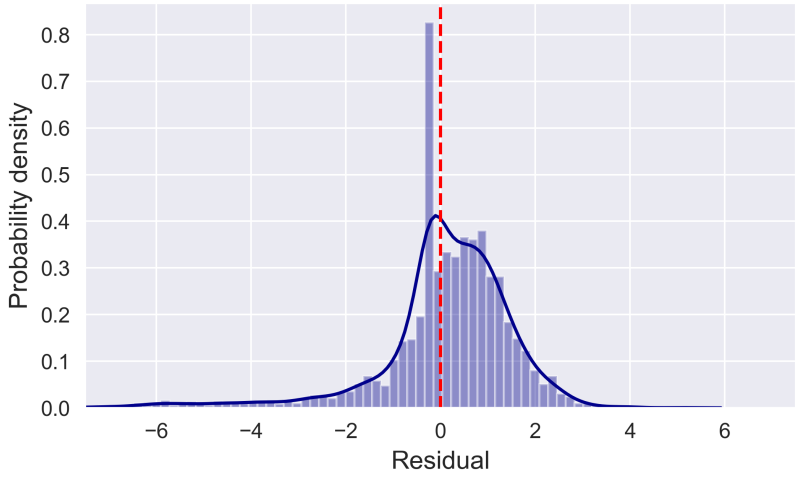


prediction horizon = 60 mins

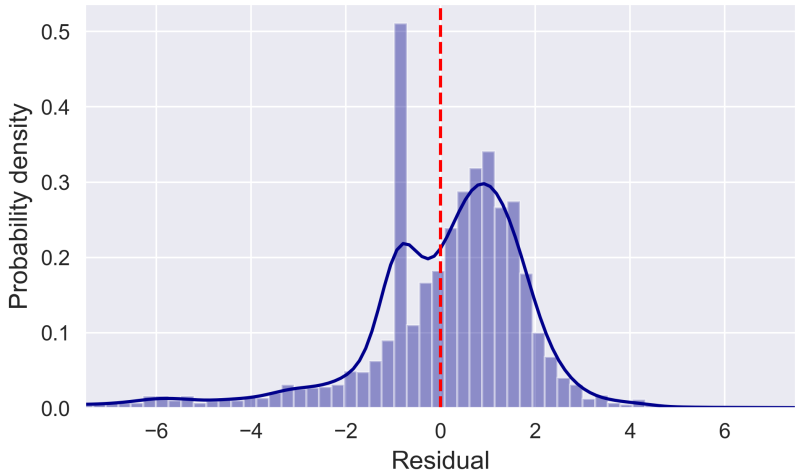
Figure 5.9: The prediction residuals of our proposed model on LOOP-SEATTLE. From top to bottom, the prediction horizons are 5, 30 and 60 mins, respectively. The x-axis represents the residuals (i.e., $v_t^i - \tilde{v}_t^i$) and the y-axis represents the probability density of the residuals.



prediction horizon = 5 mins



prediction horizon = 30 mins



prediction horizon = 60 mins

Figure 5.10: The prediction residuals of our proposed model on METR-LA. From top to bottom, the prediction horizons are 5, 30 and 60 mins, respectively. The x-axis represents the residuals (i.e., $v_t^i - \tilde{v}_t^i$) and the y-axis represents the probability density of the residuals.

2. Graph Convolutional Network (GCN) reported in (Zhang, Li, Lin, Wang & He 2019) consists of two graph convolutional layers, and each layer is followed by a $ReLU$ and a $dropout$ layer. Another work following this approach is Graph Convolutional Network with self-attention mechanism (namely SAGCN) which is treated as a sub-spatial block in our SAGCN-SST model.

3. CNN-LSTM (also known as SRCNs in (Yu et al. 2017)) consists of deep convolutional neural networks (DCNNs) and LSTMs. DCNNs are used to capture the spatial dependency of network-wide traffic and LSTMs are utilised to learn the temporal dynamics. T-GCN (Zhao, Song, Zhang, Liu, Wang, Lin, Deng & Li 2019) combines GCN and GRU. GCN captures the spatial dependency and GRU focuses on learning the temporal dependency. TGC-LSTM (Cui et al. 2019) consists of GCN and LSTM corresponding to capture spatial and temporal features, respectively. An L1-norm on the graph convolution weights and an L2-norm on the graph convolution features are added to the loss function for enhancing the interpretability of the model.

4. AGC-Seq2Seq-Attn model in (Zhang, Li, Lin, Wang & He 2019) includes two parts: the graph convolutional network and the sequence-to-sequence architecture consisting of an encoder and a decoder with the attention mechanism. Two GRUs are used to build the encoder and the decoder. The graph convolution operation is firstly utilised to capture the spatial characteristics based on the topology of the underlying road network, and then its output is treated as the input of the encoder that encodes the spatially-fused time series to a context vector. After that, the context vector is decoded to the target multi-step outputs in the decoder with the attention mechanism. Another work following this approach is the Seq2Seq-Attn model reported in (Zhang, Li, Lin, Wang & He 2019), and the main difference between Seq2Seq-Attn and AGCN-Seq2Seq-Attn (Zhang, Li, Lin, Wang & He 2019) is the graph convolution layer.

Table 5.3: Comparison of all models for both datasets

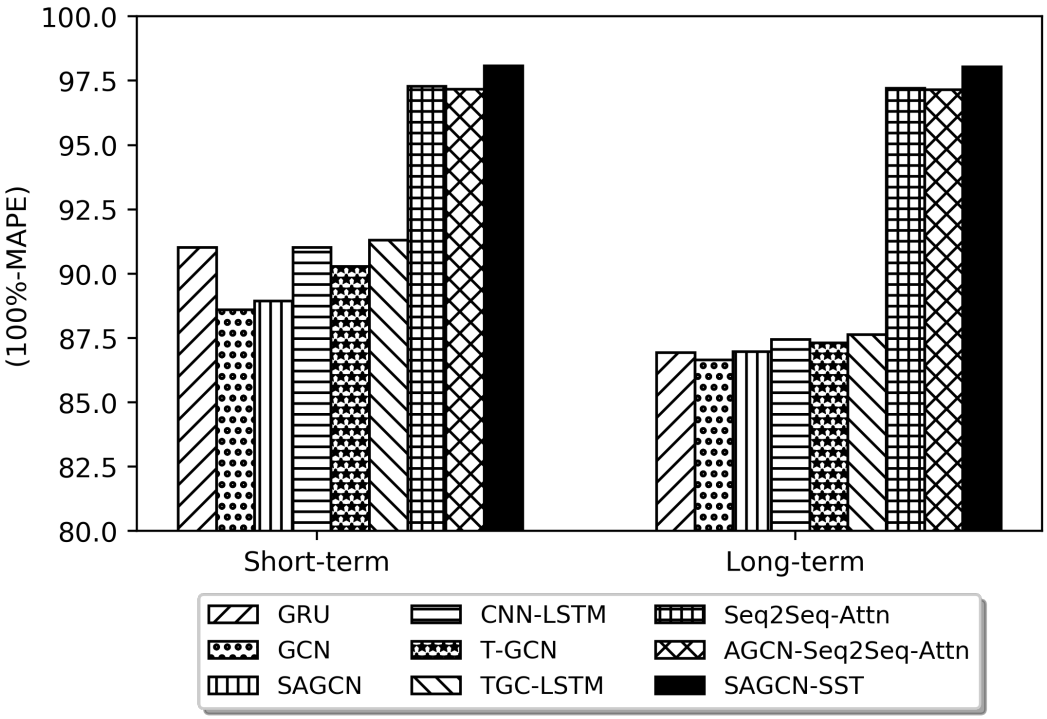
Model Name	LOOP-SEATTLE			METR-LA		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
(a) 5-min future prediction (T'=1)						
GRU	3.0796	8.10	4.5349	3.3181	7.87	5.2300
GCN	3.7696	11.00	5.8426	4.1113	10.00	6.4648

SAGCN	3.6297	10.38	5.4396	4.1258	9.86	6.3436
CNN-LSTM	3.0753	8.19	4.5690	3.2930	7.88	5.3107
T-GCN	3.3568	9.07	4.9371	3.8915	9.77	6.2140
TGC-LSTM	3.0007	7.90	4.4650	3.5857	8.31	5.5387
Seq2Seq-Attn	1.1724	2.75	1.5625	1.0344	2.10	1.4552
AGCN-Seq2Seq-Attn	1.2167	2.86	1.6378	1.2690	2.63	1.8230
SAGCN-SST	0.8156	1.90	1.0802	0.9256	1.92	1.3090
(b) 15-min future prediction (T'=3)						
GRU	3.5166	9.86	5.3336	3.8002	9.28	5.9989
GCN	3.9293	11.82	6.1877	4.4120	10.92	6.8530
SAGCN	3.8825	11.74	5.9793	4.3333	10.59	6.7672
CNN-LSTM	3.4752	9.77	5.3258	3.8441	9.42	6.1117
T-GCN	3.6482	10.36	5.5533	4.3111	10.97	6.7720
TGC-LSTM	3.4051	9.51	5.2335	3.7832	9.25	6.0438
Seq2Seq-Attn	1.1577	2.69	1.5352	1.0419	2.13	1.4614
AGCN-Seq2Seq-Attn	1.2134	2.82	1.6206	1.2925	2.67	1.8408
SAGCN-SST	0.8280	1.96	1.0982	0.9392	1.93	1.3260
(c) 30-min future prediction (T'=6)						
GRU	3.9625	11.73	6.0602	4.4496	10.92	6.8299
GCN	4.1249	12.62	6.5210	4.9551	12.24	7.4193
SAGCN	4.1371	12.63	6.4485	4.6797	11.63	7.1876
CNN-LSTM	3.8874	11.55	6.0237	4.3388	10.90	6.8113
T-GCN	3.9805	11.54	6.1095	4.6214	11.44	7.1553
TGC-LSTM	3.8570	11.17	5.9748	4.4771	11.03	6.9709
Seq2Seq-Attn	1.2281	2.87	1.6327	0.9741	1.97	1.5757
AGCN-Seq2Seq-Attn	1.2438	2.89	1.6425	1.2431	2.57	1.7826
SAGCN-SST	0.8332	1.97	1.1147	0.9242	1.91	1.3049
(d) 45-min future prediction (T'=9)						

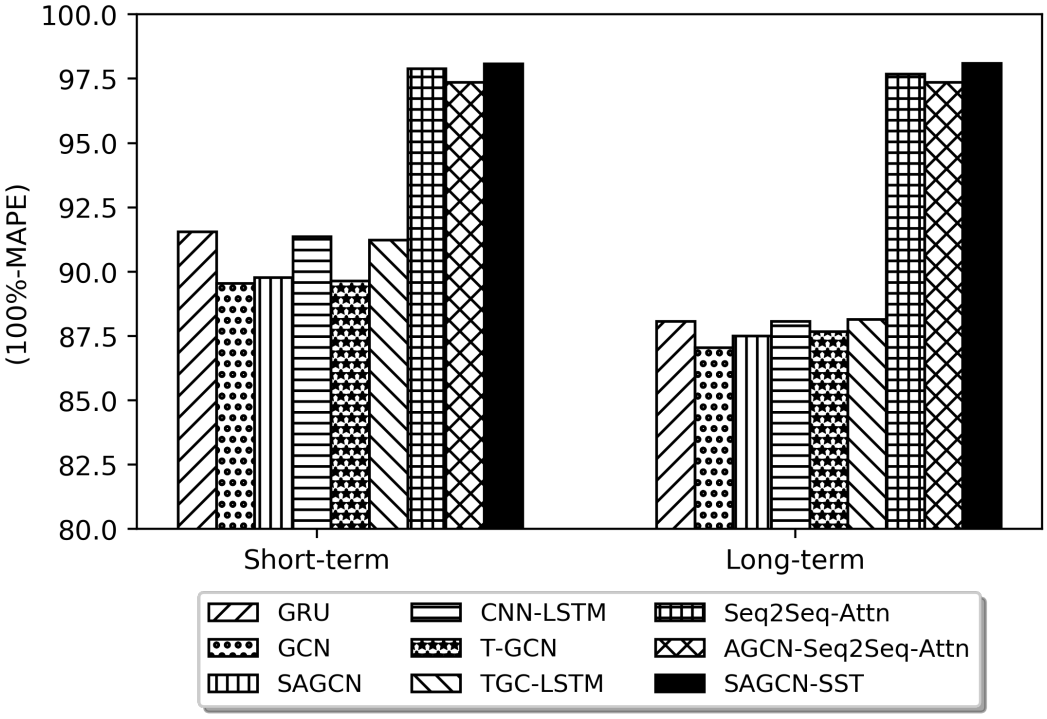
GRU	4.2900	13.04	6.5693	4.9017	12.03	7.4453
GCN	4.2786	13.00	6.7792	5.2008	12.94	7.7243
SAGCN	4.2671	12.80	6.6604	5.1772	12.75	7.7956
CNN-LSTM	4.0928	12.73	6.4350	4.7360	12.25	7.3165
T-GCN	4.2564	12.67	6.5664	4.9924	12.53	7.5092
TGC-LSTM	4.0914	12.51	6.4900	4.6058	11.84	7.2901
Seq2Seq-Attn	1.2129	2.83	1.6082	1.5080	3.19	2.1990
AGCN-Seq2Seq-Attn	1.2002	2.82	1.6132	1.2759	2.61	1.8303
SAGCN-SST	0.8221	1.94	1.0921	0.9198	1.93	1.2976
(e) 60-min future prediction (T'=12)						
GRU	4.5500	14.42	6.9905	5.1421	12.86	7.6679
GCN	4.4920	14.44	7.1989	5.4567	13.69	8.1325
SAGCN	4.4873	13.67	7.0132	5.2514	13.11	8.0610
CNN-LSTM	4.3054	13.38	6.7549	5.0391	12.67	7.7585
T-GCN	4.5414	13.90	7.0176	5.1833	13.02	7.7762
TGC-LSTM	4.3459	13.44	6.9132	5.0078	12.73	7.7247
Seq2Seq-Attn	1.1607	2.70	1.5469	0.9079	1.82	1.2839
AGCN-Seq2Seq-Attn	1.2172	2.85	1.6270	1.3181	2.73	1.8674
SAGCN-SST	0.8266	1.96	1.0985	0.9033	1.86	1.2701

Table 5.3 presents achieved results of all chosen models for $T' = \{1, 3, 6, 9, 12\}$ on both datasets while Figure 5.11 shows the corresponding accuracy for $T' = \{1, 3\}$ and $T' = \{6, 9, 12\}$. From Table 5.3 and Figure 5.11, the following attributes can be observed with regards to the different approaches to resolve the traffic prediction problem.

1. The three types of errors achieved by GRU increase when the prediction horizon is longer, as shown in Table 5.3. For example, from $T' = 1$ to $T' = 12$, the MAE, MAPE and RMSE of GRU on `LOOP-SEATTLE` increase from 3.0796, 8.10%



(a)



(b)

Figure 5.11: Comparison of prediction accuracy (100%-MAPE) on short- and long-term prediction tasks for all models on `LOOP-SEATTLE` (a) and `METR-LA` (b).

and 4.5349 to 4.5500, 14.42% and 6.9905 respectively. The same applies to the METR-LA dataset. This indicates that GRU, which mainly works for the temporal feature extraction, can offer higher prediction accuracy for short-term prediction while the performance deteriorates when the prediction horizon is longer.

2. From Table 5.3, GCN and SAGCN are the two worst performing models. This is due to the fact that GCN and SAGCN are generally used to capture spatial features. They are unable to capture temporal features. Compared to GRU that presents a large increase of MAE, MAPE and RMSE from $T' = 1$ to $T' = 12$ on both datasets, GCN and SAGCN models only present a small increase. For example, on LOOP-SEATTLE, the MAPEs of GCN and SAGCN increase by 3.44% and 2.29%, respectively, while the MAPE of GRU grows by 6.32%. The reason is that the spatial feature starts to play an increasingly more important role when the prediction horizon is longer. In addition, on both datasets, SAGCN performs slightly better than GCN for the same prediction horizon. This is because the self-attention mechanism in SAGCN is able to derive the different contributions of neighbouring nodes via distribution of different weights to each of them.

3. Spatial-temporal models (including CNN-LSTM, T-GCN and TGC-LSTM) achieve better performances on both datasets compared to spatial models (i.e., GCN and SAGCN). For example, for $T' = 1$, the average MAE, MAPE and RMSE of spatial-temporal models on LOOP-SEATTLE are 3.1443, 8.39% and 4.6570, respectively, while the average MAE, MAPE and RMSE achieved by spatial models are 3.6997, 10.69% and 5.6411, respectively. This is due to the fact that CNN-LSTM, T-GCN and TGC-LSTM are able to capture both spatial and temporal features for their final prediction while GCN and SAGCN only rely on extracting spatial features. This phenomenon is more obvious for short-term prediction. For instance, for $T' = 1$, the average MAE, MAPE and RMSE of CNN-LSTM, T-GCN and TGC-LSTM decrease by 0.5554, 2.30% and 0.9841, respectively, compared to the average MAE (3.6997), MAPE (10.69%) and RMSE (5.6411) of GCN and SAGCN. For $T' = 12$, the three types of errors only decrease by 0.0921, 0.48% and 0.2109, respectively, compared to GCN and SAGCN with 4.4897, 14.06% and 7.1061 of these errors. This is because the spatial feature starts to play an increasingly more important role when the prediction horizon is longer. Similar observations can also be found in METR-LA.

4. Seq2Seq-Attn and AGCN-Seq2Seq-Attn perform better than CNN-LSTM, T-GCN and TGC-LSTM. Their MAE, MAPE and RMSE over all different prediction horizons on two datasets are less than 1.5080, 3.19% and 2.1990, respectively. The

accuracy for short-term prediction by Seq2Seq-Attn and AGCN-Seq2Seq-Attn are similar for long-term prediction. Between Seq2Seq-Attn and AGCN-Seq2Seq-Attn, the results achieved by Seq2Seq-Attn are slightly better. This observation does not agree with the results reported in (Zhang, Li, Lin, Wang & He 2019) where the reversed is observed. The main reason for this phenomenon is that AGCN-Seq2Seq-Attn is a more complex model that requires large datasets and higher number of features for achieving better results. In (Zhang, Li, Lin, Wang & He 2019), AGCN-Seq2Seq-Attn is tested utilising several features including maximum, minimum and median of traffic speed as opposed to our experiments here where despite similar size datasets, only use the original traffic speed as the sole feature. Therefore, Seq2Seq-Attn performs slightly better than AGCN-Seq2Seq-Attn in our work.

5. Our SAGCN-SST model achieves the best results on both datasets over all different prediction horizons with an average MAPE less than 2% (i.e., prediction accuracy $> 98\%$). In addition, the MAE and RMSE are 0.8156 and 1.0802, respectively. The closest rivals are Seq2Seq-Attn and AGCN-Seq2Seq-Attn. These models consider the dynamics on temporal feature extraction by taking attention mechanism in the decoder of the sequence-to-sequence architecture. On the other hand, our SAGCN-SST considers the dynamic process on spatial feature extraction by adopting self-attention mechanism on the graph convolutional layer that can more effectively capture the dynamic spatial dependency between the targeted node and their neighbours in different neighbourhoods. The experimental results in this section also indicate that adopting self-attention mechanism on the graph convolutional layer is more efficient than including it in the decoder of the sequence-to-sequence architecture for traffic prediction. For example, comparing with less than 2% of MAPE achieved by SAGCN-SST, Seq2Seq-Attn and AGCN-Seq2Seq-Attn only obtain less than 3% of MAPE.

   In addition, our SAGCN-SST can obtain accurate predictions with small number of features. Between AGCN-Seq2Seq-Attn and SAGCN-SST, our SAGCN-SST model obtains the higher prediction accuracy ($> 98\%$) compared to the 97% accuracy achieved by the AGCN-Seq2Seq-Attn when the experiments are conducted on the same datasets. The main reason is that sub-spatial blocks in SAGCN-SST are paralleled, rather than stacked. This avoids increasing the depth of our model so as to improves scalability and avoid over-fitting at the early stage of training.

Overall, models can achieve more accurate predictions when the traffic prediction problem is treated as a dynamic spatial-temporal process as opposed to considering the prob-

lem as 1) a temporal, 2) a spatial or 3) a spatial-temporal and 4) a fixed spatial dynamic-temporal process. This can be clearly observed in Figure 5.11 where SAGCN-SST, Seq2Seq-Attn and AGCN-Seq2Seq-Attn perform significantly better than the other six models. From the results, we also see that our SAGCN-SST, that is able to capture dynamic spatial features, achieves the best results for both short- and long-term predictions. For $T' = \{1, 3\}$, the next best batch of models are GRU, CNN-LSTM, T-GCN and TGC-LSTM, followed by GCN and SAGCN. This is because GRU, CNN-LSTM, T-GCN and TGC-LSTM are able to capture temporal features, which play a more important role for short-term prediction. For $T' = \{6, 9, 12\}$, CNN-LSTM, T-GCN and TGC-LSTM still forms the group of models offering the next best results, followed by GRU that has performance similar to GCN and SAGCN, but the difference between all models is smaller than for $T' = \{1, 3\}$. This can be attributed to the increased importance of spatial features in longer prediction horizons.

5.5 Chapter Summary

In this Chapter, we present a novel deep learning model, namely SAGCN-SST, for addressing multi-step traffic speed prediction problem on large-scale road networks. We claim that the influence of different neighbouring road segments towards the future traffic state of a specific road segment of interest are unique and should be considered into the prediction process.

Considering traffic speed prediction as a dynamic spatial-temporal process, our model takes advantage of graph convolutional networks, the graph convolutional layer with self-attention mechanism and the sequence-to-sequence architecture, respectively for the fixed spatial, dynamic spatial and long-temporal feature extractions, to make our predictions. We examine our proposed model, SAGCN-SST, on two real-world large-scale road networks: `LOOP-SEATTLE` and `METR-LA`. Traffic congestion frequently occurs for a short duration in `METR-LA` and for longer duration in `LOOP-SEATTLE`.

We compare our SAGCN-SST against well-known models in recent literature including: 1) one model treating traffic prediction as a temporal process (i.e., GRU), 2) two models treating traffic prediction as a spatial process (i.e., GCN and SAGCN), 3) three models treating treating traffic prediction as a spatial-temporal process (i.e., CNN-LSTM, T-GCN and TGC-LSTM), and 4) two models treating traffic prediction as fixed spatial dynamic temporal process (i.e., Seq2Seq-Attn and AGCN-Seq2Seq-Attn). Our SAGCN-SST model achieves the highest accuracy among all competing models for both short- and

long-term predictions. The average MAE, MAPE and RMSE on both datasets with frequent traffic congestion and accidents are less than 1.0, 2.0% and 1.4, respectively, which translate to prediction accuracy being higher than 98.0%. This results show our SAGCN-SST model is not only accurate but also robust, recording similar accuracy when making predictions for different prediction horizons including the more challenging long-term prediction.

Chapter 6

Virtual and Dynamic Spatial-Temporal Feature-based Multiple-Steps Traffic Speed Prediction on Large-Scale Road Networks

6.1 Introduction

The most recent works including our previous approach presented in Chapter 5, that commonly use GCN to extract spatial dependencies of traffic on road networks, usually consider actual physical road connections between road segments. This requires topological data of the road network (conventionally represented by an adjacency matrix). Since the adjacency matrix only contains information regarding connections between adjacent neighbours, $k - hop$ matrix built based on the adjacency matrix is sometimes used to extract connectivity information within a local neighbourhood. These do not comprehensively encode the complex spatial dependencies hidden within the road network. There are different types of road with different features (e.g., speed limit, number of lanes, existence of traffic lights or roundabouts). For example, in United Kingdom, urban roads are categorised into three types: general roads, roads with traffic lights and roads with roundabouts. Generally, traffic state at a road segment near a traffic light (or roundabout) has correlation with the traffic states on other road segments with traffic lights (or roundabouts). Sensors deployed near these road features may not be directly connected but could have correlation among them. Therefore, the adjacency matrix or $k - hop$ matrix are insufficient to fully describe the complex spatial dependencies in the road network.

To account for these problems, we propose a novel deep learning model named, Virtual Dynamic Graph Convolution Network and Transformer with Gate and Attention mechanisms (VDGCNTGA) for addressing traffic prediction on large-scale road networks. The main contributions of this chapter are as follows:

- Our VDGCNTGA model makes predictions using a *virtual* dynamic road network updated after each batch based on the Self-Attention mechanism. This idea is inspired by the application of GCN on solving classification problems (Mondal et al. 2020). Instead of purely relying on the actual physical road topology (as that represented by adjacency matrix or $k - hop$ matrix), we consider a certain number of randomly selected samples of traffic data for each node once. We then treat the historical traffic data in those samples as the node features to analyse the relations of each node to others on the network for generating and updating the virtual dynamic graph. Not only can this method capture dynamic and hidden correlations among road segments across the network, it also allows the model to learn a more generalised graph because the relationships of nodes are learned from randomly selected samples of historical traffic data (cf. Section 6.2.2).

- We exploit the attention mechanism of the transformer technology (Vaswani et al. 2017) to capture dynamic and hidden spatial and temporal features from historical traffic data. We further employ graph convolution neural network to correct the learned dynamic spatially-fused features from transformers on the updated dynamic graph (cf. Section 6.2.3).

- We compare our VDGCNTGA model against seven well-known existing models using two real-world large-scale road traffic datasets. In addition, we conduct ablation experiments to further gain insights into the characteristics of our model by systematically evaluating it with removal of individual constituent module.

The rest of this chapter is organised as follows. First, we define our traffic prediction problem and detail the design rationale and the architecture of our virtual dynamic graph convolution network and transformer with gate and attention mechanisms (VDGCNTGA) framework in Section 6.2. We then evaluates our proposed model and compares it with seven well-known existing models in recent literature on two real-world large-scale road network datasets, and also conduct ablation experiments to understand and gain insights of our model in Section 6.4. Finally, we summary our work in Section 6.5.

6.2 Methodology

6.2.1 Problem Formulation

Considering a road network with a set of N geographically distributed sensors, let x_t^i denote the traffic speed measured at node i at t^{th} time interval. The traffic speed data is written as $x_t = \{x_t^1, x_t^2, \ldots, x_t^i, \ldots, x_t^{N-1}, x_t^N\}; x_t \in \mathbb{R}^N$. Typical time interval, m, could be 5, 15, 30, 45 and 60 minutes (Bickel et al. 2007). The datasets used for our experiments are with $m = 5$ minutes. We then define X_T and X_D as the traffic speed data collected from N sensors for T previous time intervals and the same time interval with the targeted time interval in D previous days, respectively. They are formulated as follows:

$$\begin{aligned} X_T &= \{x_1, x_2, \ldots, x_t, \ldots, x_T\}; \\ X_T &\in \mathbb{R}^{T \times N}, T = 1, 2, 3, \ldots \end{aligned} \tag{6.1}$$

$$\begin{aligned} X_D = \{&X_{(T+T')-\frac{24\times60}{m}\times D}, X_{(T+T')-\frac{24\times60}{m}\times(D-1)}, \ldots, \\ &X_{(T+T')-\frac{24\times60}{m}\times2}, X_{(T+T')-\frac{24\times60}{m}\times1}\}; \\ X_D &\in \mathbb{R}^{D\times T'\times N}, D = 1, 2, 3, \ldots, T' = 1, 2, 3 \ldots \end{aligned} \tag{6.2}$$

In a similar manner, future traffic data is denoted as $X_{T+T'} = \{x_{T+1}, x_{T+2}, \ldots, x_{T+t'}, \ldots, x_{T+T'}\} \in \mathbb{R}^{T'\times N}$ where T' is the prediction horizon. In this chapter, we consider multi-interval predictions where $T' = \{1, 2, 3, \ldots, 12\}$ corresponding to $\{5, 10, 15, \ldots, 60\}$ minutes. These are common values used in the literature (e.g., (Li et al. 2018, Cui et al. 2019)).

Conventionally (e.g., (Cui et al. 2020, Zhang, Li, Lin, Wang & He 2019)), the road graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes representing road segments or sensor locations with $|\mathcal{V}| = N$ and $\mathcal{E}$ is the set of edges representing connectivity between road segments. $\mathcal{G}$ can be represented by $A \in \mathbb{R}^{N\times N}$, the $N \times N$ symmetric adjacency matrix, with its element $A_{i,j} = 1$ if there exists a link between node i and j, otherwise $A_{i,j} = 0$. Since future traffic state of a node is influenced by its own current state, $\mathcal{G}$ can then be written as $A_0 = (A + I_N) \in \mathbb{R}^{N\times N}$ where I_N is the $N \times N$ identity matrix. However, in our work, instead of purely using physical road connectivity, we advocate the use of virtual dynamic road graph (cf. Section 6.2). The virtual dynamic road graph

generated by the Self-Attention Block in our model is represented by $A_{vd} \in \mathbb{R}^{N \times N}$. Considering that traffic speed has both short- and long-term temporal patterns (Shi et al. 2020), the timestamps of traffic data X_T, including the time interval of a day and the day of a week, are defined as external features (i.e., $TT' = \{t_T, t_D\}; TT' \in \mathbb{R}^{(T+T') \times 2}$, $t_T = \{t_1, \ldots, t_t, \ldots, t_T, t_{T+1}, \ldots, t_{T+t'}, \ldots, t_{T+T'}\} \in \mathbb{R}^{T+T'}; t_t = 1, 2, \ldots, \frac{24 \times 60}{m}$ and $t_D = \{t_{d_1}, \ldots, t_{d_t}, \ldots, t_{d_T}, t_{d_{T+1}}, \ldots, t_{d_{T+t'}}, \ldots, t_{d_{T+T'}}\} \in \mathbb{R}^{T+T'}; t_{dt} = 1, 2, 3, 4, 5, 6, 7$, respectively) and are used to embed external temporal features. The timestamps of traffic data X_D are defined as $TDT' \in \mathbb{R}^{D \times (T+T') \times 2}$ in a similar manner. Based on traffic speed data and the road graph information, the traffic prediction problem considered here can be formulated as following.

$$\tilde{X}_{T+T'} = \boldsymbol{F}\Big(X_T; X_D; TT'; TDT'; \mathcal{G}(\mathcal{V}, \mathcal{E}, A_0)\Big) \tag{6.3}$$

where the objective is to learn the mapping function $F(.)$ and then use the learned $F(.)$ to compute the traffic speed in the next T' time intervals based on traffic speed data in T previous time intervals and in the same time interval with the targeted time interval from D previous days, their timestamps and the virtual road graph information.

6.2.2 VDGCNTGA Workflow

Figure 6.1 presents the workflow of our VDGCNTGA model for network-wide traffic prediction given a road map. It predicts traffic in the next T' time intervals using historical traffic data. It consists of two main phases (i.e., training and testing phases) with each phase taking slightly different inputs.

In the training phase, VDGCNTGA takes five inputs:

1. historical traffic data for T time intervals, $X_T \in \mathbb{R}^{B \times T \times N}$ (Note that X_T is formatted to $\mathbb{R}^{B \times T \times N}$ for the training process and the same operation applies to other inputs. B is the batch size.),

2. historical traffic data in the same time interval with the targeted time interval from past D days, $X_D \in \mathbb{R}^{B \times D \times T' \times N}$,

3. the timestamps of time intervals in a day and days in a week, $TT' \in \mathbb{R}^{B \times (T+T') \times 2}$ and $TDT' \in \mathbb{R}^{B \times (T+T') \times 2D}$,

4. pre-embedded physical spatial matrix based on the original road graph, $\mathcal{G}(\mathcal{V}, \mathcal{E}, A_0)$,

5. virtual dynamic spatial matrix, $A_{vd} \in \mathbb{R}^{N \times N}$.

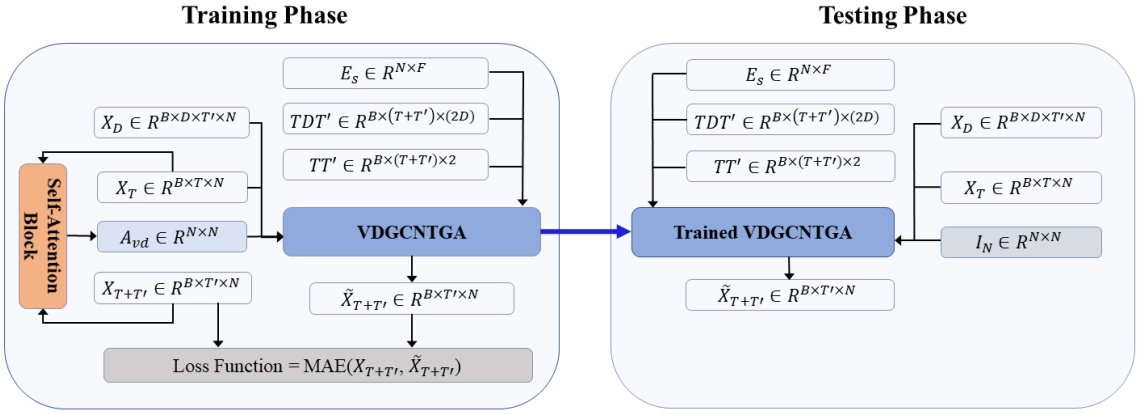


Figure 6.1: The workflow of the VDGCNTGA model for multi-interval traffic predictions.

Particularly, the virtual dynamic spatial matrix, A_{vd}, is updated after each batch in the training phase by a Self-Attention-Block. This enables VDGCNTGA to learn the dynamic and hidden spatial dependencies of road segments across the network. This Self-Attention-Block maps and learns the relationships between historical traffic data X_T and $X_{T+T'}$ via the following:

$$A_{vd} = Softmax\left(Reshape(X_T)\left(Reshape(X_{T+T'})\right)^{\mathsf{T}}\right) \tag{6.4}$$

where $Reshape(.)$ is used to reshape $\mathbb{R}^{B \times T(orT') \times N}$ into $\mathbb{R}^{N \times BT(orBT')}$ so as to obtain the virtual dynamic spatial matrix $A_{vd} \in \mathbb{R}^{N \times N}$. This operation enables each node to build the relationships with others through BT features from B samples. It results in the learned spatial matrix that is (1) dynamic as it is updated after each batch in the process and (2) general as the learned spatial matrix is not specific to any time interval due to fact that we use randomly selected B samples rather than particular fixed time intervals.

After the training phase, the spatial dependencies of road segments on the network would have been learned and encoded in the weight and bias matrices of A_{vd}. When the trained VDGCNTGA model is tested, the virtual dynamic spatial matrix, A_{vd}, is replaced by the identity matrix, $I_N \in \mathbb{R}^{N \times N}$. This is to restore the learned spatial matrix so as to avoid information leakage and thus ensuring model predictions are made only based on learned features and historical traffic data.

6.2.3 VDGCNTGA Model Architecture

Since VDGCNTGA uses the same internal structure to extract features from X_T and X_D, we will only detail in the following the internal structure for X_T as depicted in Figure 6.2. It consists of three main blocks, i.e., two Spatial and Temporal Transformer blocks (STTras-Blocks), each consisting of a Spatial Transformer (STm), a Temporal Transformer (TTm) and a Spatial-Temporal Fusion (STF) module, and a Dynamic Graph Convolution Network Block (DGCN-Block), connected between the two STTras-Blocks. STm and DGCN-Block are considered as memory-less parts while TTm is considered as a memory-based part. The STTras-Blocks are used for spatial and temporal feature analysis. The DGCN-Block is for dynamic spatial feature analysis as well as for correcting the learned spatially-fused feature from the first STTras-Block. In addition, VDGCNTGA also includes two additional embedding modules, namely Spatial Embedding (SEm) for spatial matrix (E_s) embedding based on the physical road network, A_0, and Temporal Embedding (TEm) for temporal matrix (T_T and $T_{T'}$) embedding based on the timestamps of traffic data, TT'. The Self-Attention block is included only in the training phase for updating the virtual dynamic spatial matrix A_{vd} which takes place after each batch.

The embedded spatial matrix, E_s, is generated by SEm using the physical road graph, A_0, while the embedded temporal matrices, T_T and $T_{T'}$, are generated by TEm using the timestamps of the historical and targeted traffic data TT'. X_T and E_s are sent to STm in the first STTras-Block for spatial feature learning while X_T and T_T are fed to its TTm for historical temporal feature learning. Both learned spatial and temporal features are fused in STF based on the gate mechanism in GRU (Fu et al. 2016). Then the output with the virtual dynamic spatial matrix, A_{vd}, are sent to the DGCN-Block for further dynamic spatial feature analysis and learned spatially-fused feature correction. The output of DGCN-Block, A_{vd}, E_s and $T_{T'}$, are further fed into the second STTras-Block for learning dynamic spatial and temporal features by aligning the learned features from DGCN-Block with virtual dynamic spatial matrix A_{vd} in STm and with the targeted temporal embedding matrix $T_{T'}$ in TTm. Finally, the fused dynamic spatial-temporal features from STF are sent to a Fully-Connected Layer for the final prediction. We detail the operations of each VDGCNTGA block and module next.

SEm

This module embeds the physical road network into a spatial matrix E_s using the $node2vec$ algorithm (Grover & Leskovec 2016). It maps nodes within the road network to a low-

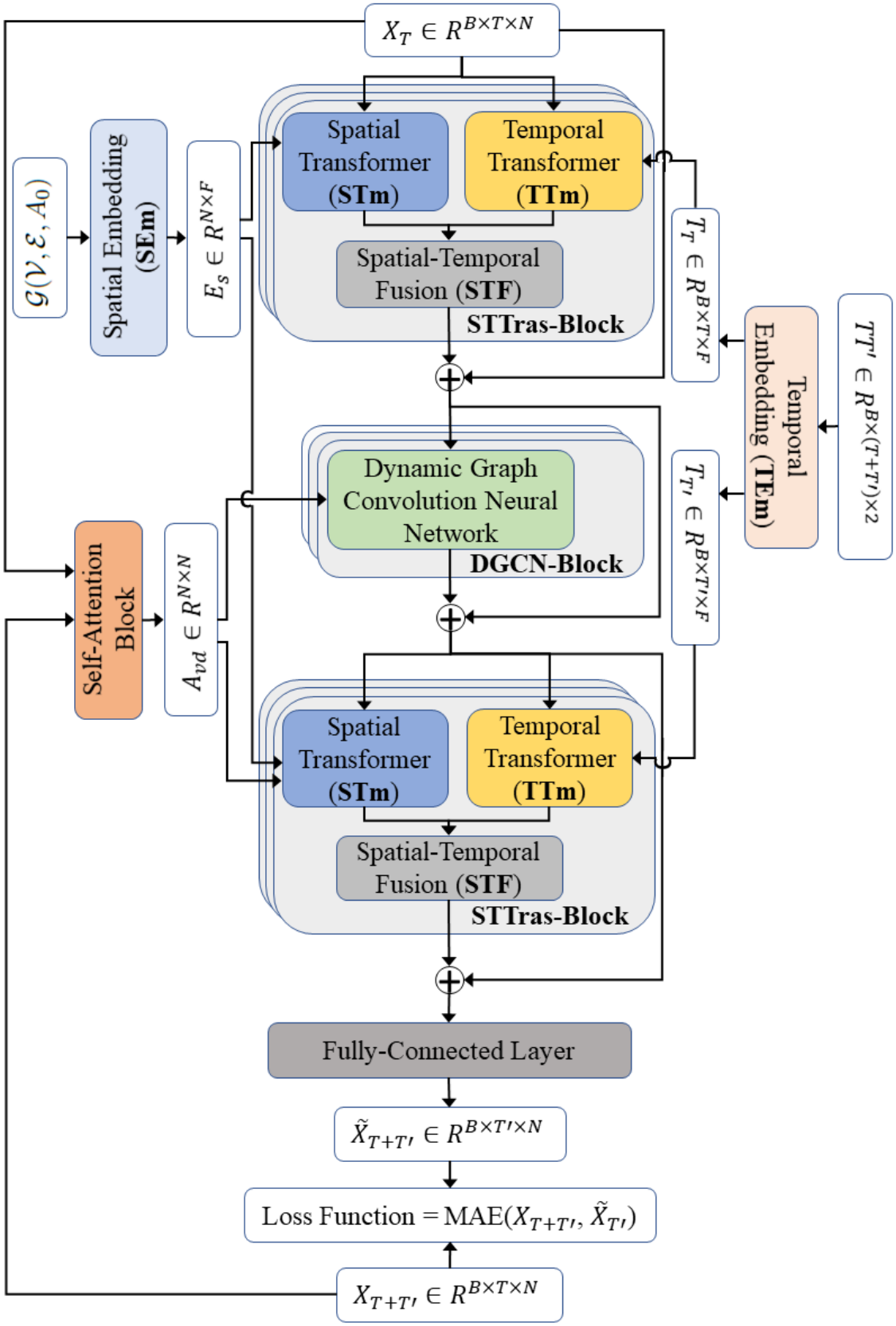


Figure 6.2: Internal structure of the VDGCNTGA model.

dimensional space while keeping the relationships of nodes to their neighbours. In our work, the physical road network, $A_0 \in \mathbb{R}^{N \times N}$, is learned and mapped as $E_s \in \mathbb{R}^{N \times F}; F < N$ by a biased random walk procedure, which provides a flexible neighbourhood size to each node for learning richer representations.

TEm

This module aims to embed the timestamps of historical and targeted traffic data as external temporal features. It consists of an encoding layer and a fully-connected layer. The encoding layer applies the One-Hot Encoding technique (Karthiga et al. 2021) to embed $TT' \in \mathbb{R}^{B \times (T+T') \times 2}$ as $TT'_{em} \in \mathbb{R}^{B \times (T+T') \times 295}$ where 295 is the sum of 288 (i.e., the time-interval timestamps in a day) and 7 (i.e., the number of daily timestamps in a week). Therefore, one value in the third dimension used to present the time-interval timestamp from TT' is embedded into a vector with the length of 288 while the other one for the daily timestamp is embedded into a vector with the length of 7. Both embedded vectors are concatenated into one. After that, TT'_{em} is sent to a fully-connected layer before being split into $T_T \in \mathbb{R}^{B \times T \times F}$ as historical-external temporal features and $T_{T'} \in \mathbb{R}^{B \times T' \times F}$ as targeted-external temporal features where F is the number of embedded features in the Fully-connected layer.

STTras-Block

This block aims to analyse dynamic spatial and temporal features. Its internal structure is shown in Figure 6.3. It consists of three modules: STm, TTm and STF.

STm (dark blue box at the upper left corner in Figure 6.3) is responsible for dynamic spatial feature extraction. It consists of two main types of layers: Fully-Connected Layer and Spatial Attention Layer. First, the input X_T passes a Fully-Connected layer for embedding more features as $X_T^{STm;fc1} \in \mathbb{R}^{B \times T \times N \times F}$. Then the embedded spatial matrix E_s is joined into $X_T^{STm;fc1}$ by the function, $(X_T^{STm;Es} = X_T^{STm;fc1} + E_s; X_T^{STm;Es} \in \mathbb{R}^{B \times T \times N \times F})$ so as to enable the traffic input to carry spatial dependencies of road segments. To obtain more information, $X_T^{STm;fc1}$ and $X_T^{STm;Es}$ are concatenated as $X_T^{STm;cl} \in \mathbb{R}^{B \times T \times N \times 2F}$ for generating (1) Queries, $Q^S \in \mathbb{R}^{B \times T \times N \times (h \times d_q^s)}$, (2) Keys, $K^S \in \mathbb{R}^{B \times T \times N \times (h \times d_k^s)}$ and (3) Values, $V^S \in \mathbb{R}^{B \times T \times N \times (h \times d_v^s)}$ through three Fully-Connected layers with ReLU activation functions using the following equations:

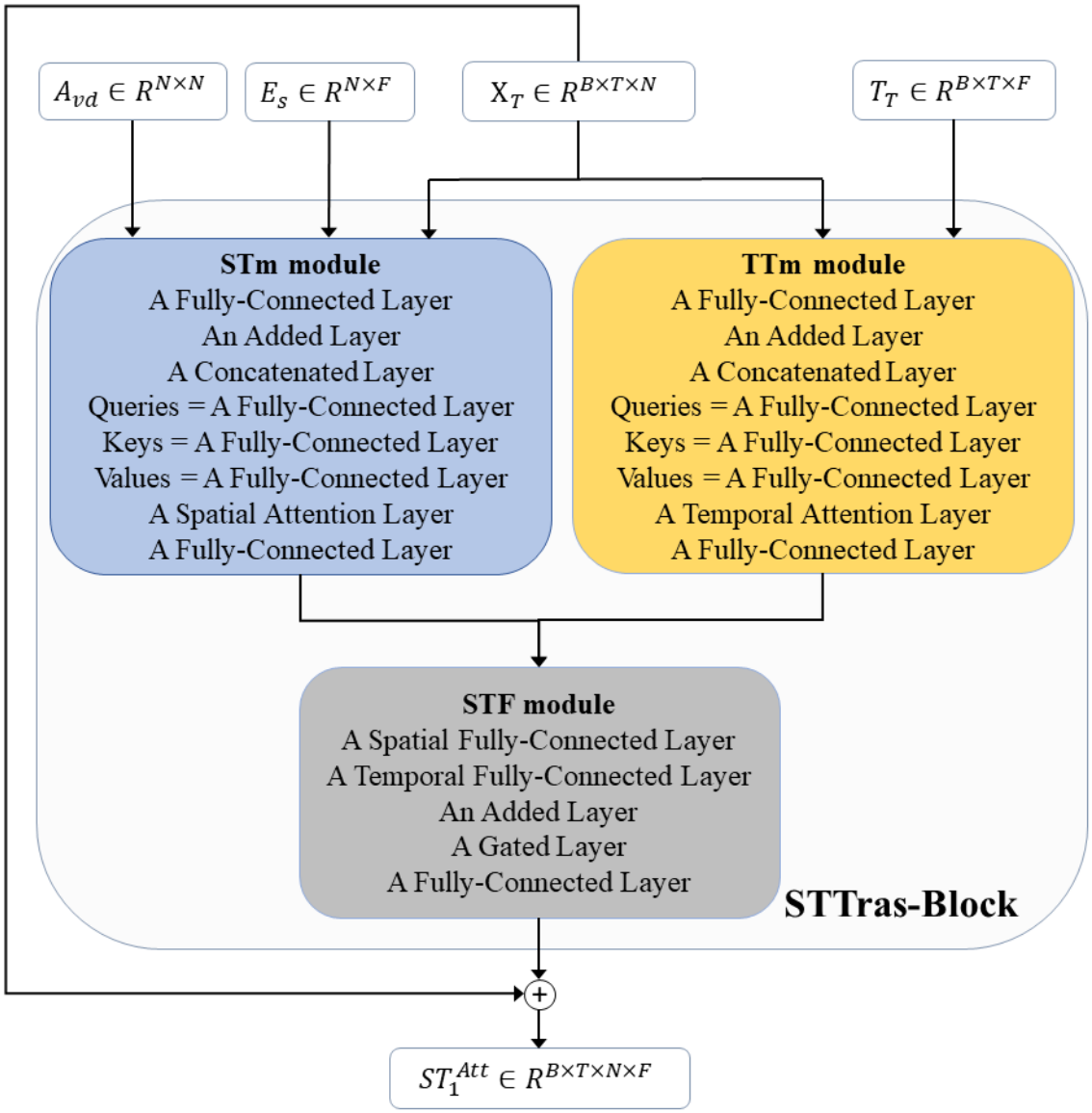


Figure 6.3: The STTras-Block with its constituent modules including (1) STm (the left dark blue box), (2) TTm (the right yellow box) and (3) STF (the bottom grey box)

$$
\begin{aligned}
Q^S &= ReLU(w_q^s X_T^{STm;cl}) \\
K^S &= ReLU(w_k^s X_T^{STm;cl}) \\
V^S &= ReLU(w_v^s X_T^{STm;cl})
\end{aligned}
\tag{6.5}
$$

where w_q^s, w_k^s and w_v^s are learnable weight matrices and d_q^s, d_k^s and d_v^s are embedded spatial features of each sensor or road segment for Q^S, K^S and V^S, respectively. h is the number of heads.

After obtaining the three high dimensional spatially-fused features (Q^S, K^S and V^S), dynamic-spatial dependencies $S^S \in \mathbb{R}^{B\times T\times N\times F}$ are calculated by a Spatial Attention layer, given by

$$S^S = Softmax\left(\frac{A_{vd}\left(Q^S(K^S)^\intercal\right)}{\sqrt{d_k^s}}\right)V^S \tag{6.6}$$

where $Q^S(K^S)^\intercal \in \mathbb{R}^{B\times T\times N\times N}$ represents the relationship of each road segment to others in the network and A_{vd} is used to correct this spatial relationship. $\intercal$ stands for matrix transpose operation. Note that the correction of the spatial relationship should be done before multiplying a scaled dot-product attention $\frac{1}{\sqrt{d_k^s}}$. Otherwise, the dot-products grow larger again in magnitude, which could push the Softmax function into regions where it has extremely small gradients (Vaswani et al. 2017). In addition, A_{vd} is set as "None" in the first STTras-Block to learn the initial physical-spatial dependencies. In order to explore interactions among latent features, a Fully-Connected layer with ReLU activation function is used for computing dynamic spatial features $S^S_{output} \in \mathbb{R}^{B\times T\times N\times F}$ with the residual connection, which is formulated as follow:

$$S^S_{output} = ReLU(w_o^s S^S) + X_T^{STm;fc1} \tag{6.7}$$

where w_o^s is the learnable weight matrix and S^S_{output} is the output of STm.

TTm (yellow box at the upper right corner in Figure 6.3) is responsible for dynamic temporal feature extraction. Similar to STm, it also consists of two main types of layers: Fully-Connected Layer and Temporal Attention Layer. Existing works (e.g., (Cui et al. 2019, Zhang, Li, Lin, Wang & He 2019, Zhao, Song, Zhang, Liu, Wang, Lin, Deng & Li 2019)) either used GRU or LSTM to capture temporal features due to their abilities in analyzing long-term dependency using recurrent units to deliver temporal features from the current time interval to the next time interval. However, the transformer-based models use the attention mechanism to distribute different weights to traffic data from previous time intervals so as to contribute to traffic data in the future time intervals.

The inputs required by TTm are the historical traffic data, X_T, and the historical-external temporal features, T_T. Note that T_T is replaced by the future-external temporal features, $T_{T'}$, in the second STTras-Block. This design aims to learn the historical-external temporal features first and then align with the future-external temporal features, $T_{T'}$, to improve the prediction performance. The first Fully-Connected layer is used to embed more fea-

tures into X_T as $X_T^{TTm;fc1} \in \mathbb{R}^{B\times T\times N\times F}$, and then an Added layer enables the embedded features to be enhanced by the historical-external temporal features through the function of $X_T^{TTm;T_T} = X_T^{TTm;fc1} + unsqueeze(T_T); X_T^{TTm;T_T} \in \mathbb{R}^{B\times T\times N\times F}$. The function $unsqueeze(.)$ is used to expand an additional dimension to match the format of $X_T^{TTm;fc1}$. To correct the embedded features and obtain more information, $X_T^{TTm;fc1}$ and $X_T^{TTm;T_T}$ are concentrated as $X_T^{TTm;cl} \in \mathbb{R}^{B\times T\times N\times 2F}$ before being sent to three Fully-Connected layers for generating (1) Queries $Q^T \in \mathbb{R}^{B\times N\times T\times (h\times d_q^t)}$, (2) Keys $K^T \in \mathbb{R}^{B\times N\times T\times (h\times d_k^t)}$ and (3) Values $V^T \in \mathbb{R}^{B\times N\times T\times (h\times d_v^t)}$ by Eq. (6.8).

$$\begin{aligned} Q^T &= ReLU(w_q^t X_T^{TTm;cl}) \\ K^T &= ReLU(w_k^t X_T^{TTm;cl}) \\ V^T &= ReLU(w_v^t X_T^{TTm;cl}) \end{aligned} \tag{6.8}$$

where w_q^t, w_k^t, w_v^t are learnable weight matrices for Q^T, K^T and V^T, respectively while d_q^t, d_k^t and d_v^t are the corresponding embedded temporal features of each time interval. After achieving the three high dimensional temporally-fused features (Q^T, K^T and V^T), dynamic-temporal dependencies $S^T \in \mathbb{R}^{B\times N\times T\times F}$ are calculated by a Temporal Attention layer as given in Eq. (6.9).

$$S^T = Softmax\left(\frac{Q^T(K^T)^\intercal}{\sqrt{d_k^t}}\right)V^T \tag{6.9}$$

where $Q^T(K^T)^\intercal \in \mathbb{R}^{B\times N\times T\times T}$ represents the relationship of each time interval to all others. We follow (Xu et al. 2020) to set $d_q^t = d_k^t = d_v^t = 8$. Furthermore, a Fully-Connected layer with ReLU activation function is used to generate dynamic temporal features $S_{output}^T \in \mathbb{R}^{B\times N\times T\times F}$ with the residual connection via Eq. (6.10).

$$S_{output}^T = ReLU(w_o^t S^T) + (X_T^{TTm;fc1})^\intercal \tag{6.10}$$

where w_o^t is the learnable weight matrix and S_{output}^T is the output of TTm.

STF fuses dynamic-spatial and dynamic-temporal features (S_{output}^S and S_{output}^T) into $S^{ST} \in \mathbb{R}^{B\times T\times N\times F}$ based on the gate mechanism of GRU (Fu et al. 2016). First, the dynamic-spatial features S_{output}^S are added to the dynamic-temporal features S_{output}^T after passing a Fully-Connected layer with $Tanh$ activation function (as given in Eq. (6.11)).

Then, the gate mechanism (cf. Eq. (6.12)) is used for calculating the output of STF, $S_G^{ST} \in \mathbb{R}^{B\times T\times N\times F}$.

$$S^{ST} = Tanh\Big(w_{ss}S^S_{output} + w_{st}(S^T_{output})^{\intercal}\Big) \tag{6.11}$$

$$S^{ST}_G = S^{ST} \times S^S_{output} + (1 - S^{ST}) \times S^T_{output} \tag{6.12}$$

where w_{ss} and w_{st} are learnable weight matrices of S^S_{output} and S^T_{output} respectively.

DGCN

This block aims to correct the learned spatial relationships of sensors or road segments by conducting the convolutional operation on the virtual dynamic spatial matrix A_{vd} and then joining the corrected spatial relationships into S^{ST}_G. It consists of a GCN layer with ReLU activation function and a Batch Normalization layer. The inputs of this block are the updated virtual dynamic spatial matrix A_{vd} and S^{ST}_G. The output of this block, $GCN_d \in \mathbb{R}^{B\times T\times N\times F}$, representing the corrected spatially-fused features, is computed via Eq. (6.13).

$$GCN_d = BatchNorm\Bigg(ReLU\Bigg((w_{dgcn} * A_{vd} + b_{dgcn})S^{ST}_G\Bigg)\Bigg) + S^{ST}_G \tag{6.13}$$

where $w_{dgcn} \in \mathbb{R}^{N\times N}$ is the parameter matrix while $b_{dgcn} \in \mathbb{R}^N$ is the related bias. $BatchNorm(.)$ is the batch normalization and $*$ is the Hadamard product operator.

6.3 Data Description

To evaluate our VDGCNTGA model, two real-world datasets from large-scale road networks, labelled as `PEMS-BAY` and `METR-LA` (Li et al. 2018), are used. The detailed information about those two datasets can be found in Chapter 2.6.2.

6.4 Experiments

6.4.1 Parameter Study

For VDGCNTGA to obtain accurate predictions, there are three types of parameters needed to be set and learned: (1) parameters for the input preparation, (2) hyper-parameters for our VDGCNTGA model and (3) parameters for training. We set and learn those parameters as follows:

(1) For the input preparation, the parameters include historical time intervals T, targeted time intervals T', previous days D and batch size B. We follow (Li et al. 2018, Guo et al. 2019) and set historical time intervals T, targeted time intervals T', and previous days D as 12, 12 and 1, respectively. For setting batch size B, we experimentally find the best value as it refers to the number of features (=$B \times T$) used to update the virtual dynamic spatial matrix A_{vd} and directly affects the performance of our VDGCNTGA. If B is too large, it would bring too many features when updating the virtual dynamic spatial matrix. On the other hand, if it is too small, it would not maximise the generalisation of the learned virtual dynamic spatial matrix. We show this in Figure 6.4 which shows the relationships of batch size and MAE for both `PEMS-BAY` (blue line with dot marker using left y-axis) and `METR-LA` (black line with star marker using right y-axis). For `METR-LA`, the lowest average MAE is achieved when B is equal to 17 while, for `PEMS-BAY`, B is 23. The reason of requiring larger B for `PEMS-BAY` is that `PEMS-BAY` contains 325 sensors and needs more features (=$B \times T$) to update the virtual dynamic spatial matrix A_{vd}, compared to 207 sensors in `METR-LA`.

(2) For the proposed model, hyper-parameters include the number of multi-heads h and the number of embedding features $\{d_q^t, d_k^t, d_v^t, d_q^s, d_k^s, d_v^s\}$. We follow (Zheng et al. 2020) to set both the number of embedding features and the number of multi-heads to 8. Therefore, the number of the embedded features F from the first Fully-Connected layer in both STm and TTm is equal to 64 ($= 8 \times 8$). In addition, the total number of parameters for `PEMS-BAY` and `METR-LA` are 298,849 and 235,955, respectively. Again, more parameters are required for `PEMS-BAY` due to the higher number of sensors, resulting in more parameters in DGCN-Block.

(3) For the training phase, parameters include learning rate, r and the number of epochs, e. Setting the learning rate to a small value will typically make a training algorithm converge slowly and conversely, using a large value for learning rate may make the

algorithm to diverge. Using experimental methods to find the best learning rate is usually time consuming. In our work, we use the Cyclical Learning Rates (CLR) method (Smith 2017) to optimize the learning rate. Based on this, the optimized learning rate is $1.12e^{-03}$. In addition, we use *stop early* strategy to find the number of epochs. Specifically, the *stop early* strategy will stop the training process when the training loss continues to decrease in 10 consecutive epochs while the validation loss increases. This avoids the problem of over-fitting. Figure 6.5 presents the relationships of training loss, validation loss and the number of epochs. The x-axis represents the number of epochs and the y-axis gives the MAE loss (left y-axis for training loss and right y-axis for validation loss). Due to the *stop early* strategy, the training process stops at epoch 13 for `PEMS-BAY` and at epoch 12 for `METR-LA` where the validation losses are the lowest.

Considering that $Adam$ (Kingma & Ba 2014) has been shown to be very efficient for optimising parameters of deep learning models, we have used $Adam$ as the Optimiser in this work. Finally, we follow the convention (e.g., (Zheng et al. 2020)) and use 70% of the dataset for training, 10% for validation and 20% for testing. All experiments are conducted on a machine equipped with a GeForce RTX 2080 Ti GPU card with 11 GB Memory and 1545MHz Boost Clock and PyTorch is used to implement this work.

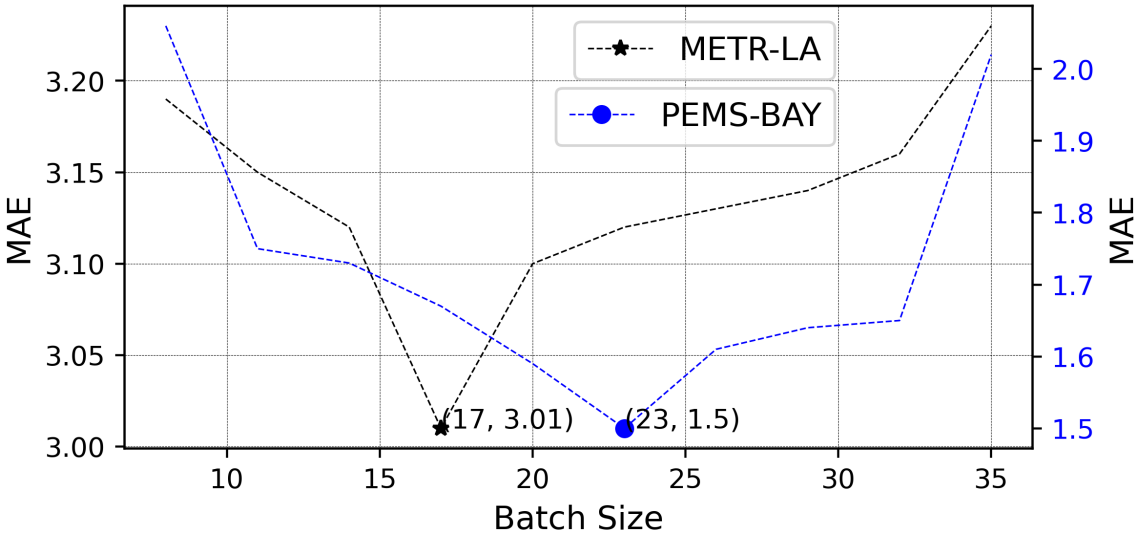
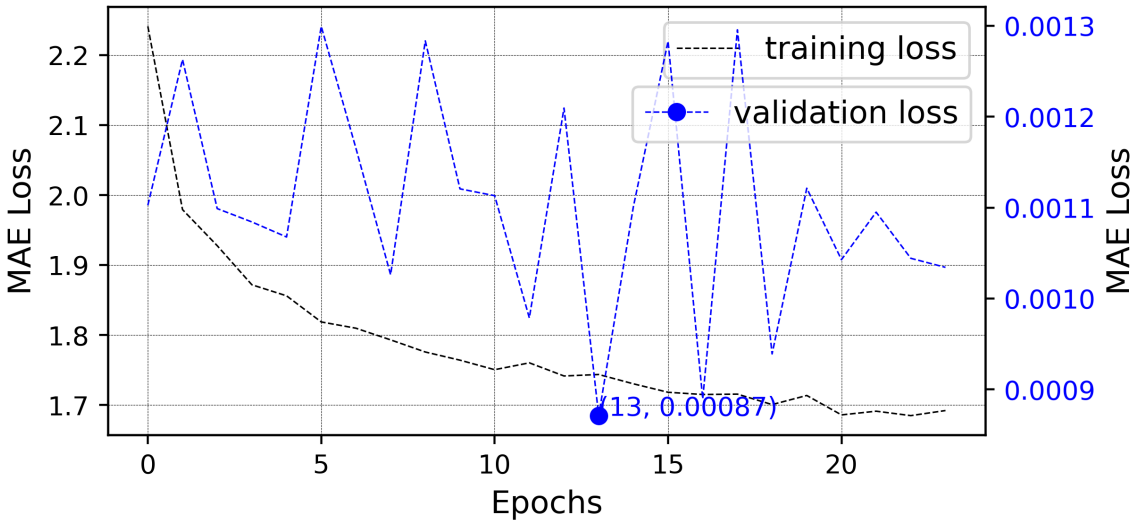


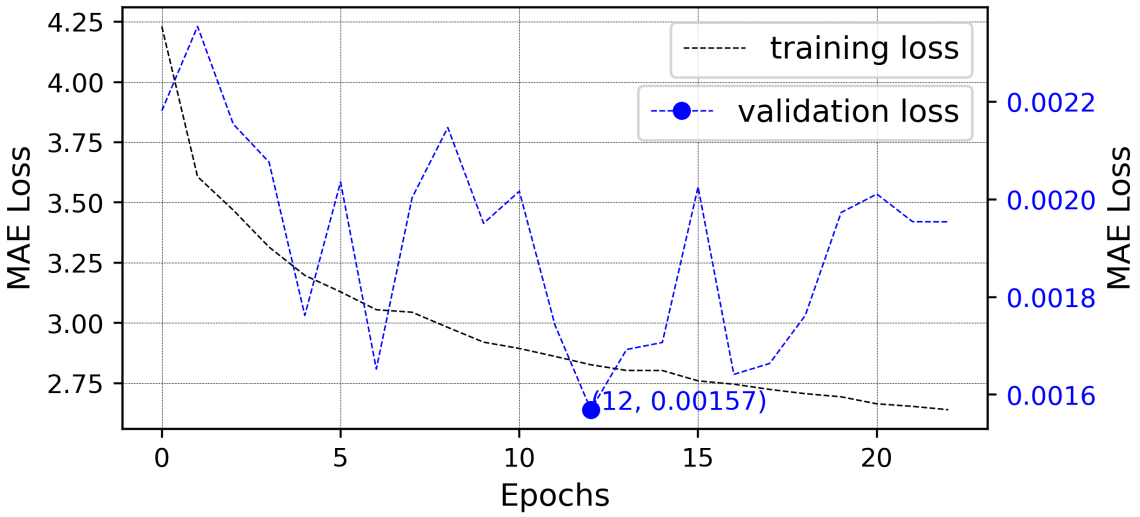
Figure 6.4: The relationship of batch size and MAE for both datasets.

6.4.2 Analysis of the VDGCNTGA Model

To gain insights into the performance and behaviour of VDGCNTGA, we conducted the following evaluations: (1) Comparison against ground truth data, (2) parameter visualisa-



(a)



(b)

Figure 6.5: The relationship of training and validation MAE losses with the number of epochs.

tion and (3) ablation experiment.

Comparison against Ground Truth Data

Figure 6.6 and Figure 6.7 show the real and predicted traffic states by VDGCNTGA in a day from two randomly selected sensors in `PEMS-BAY` and `METR-LA`, respectively. The prediction horizons are 5-min, 15-min, 30-min, 45-min and 60-min from top to bottom, respectively. The locations of these two selected sensors are shown in Figure 6.8 and Figure 6.9 with blue pointers (pointer 400030 for `PEMS-BAY` and pointer 773869

for METR−LA). In Figure 6.7, we indicated serious missing data in the METR−LA dataset with shaded area (e.g., traffic speed continues to be Nan between 2012-03-02 10:20:00 and 2012-03-02 11:10:00.) and, even so, VDGCNTGA can still predict traffic states and follow the trend over time. Besides, both real and predicted traffic speed figures also show that the predictions of our VDGCNTGA are more accurate on PEMS−BAY compared to METR−LA. From 5-min to 60-min prediction, the predictions are increasingly less accurate but overall, still follow the trend of changes over time.

Figure 6.8 and Figure 6.9 compare the real (top sub-figures) and predicted (bottom sub-figures) traffic speed by our VDGCNTGA model on PEMS−BAY and METR−LA, respectively. From the figures, we see close agreements between the real and the predicted traffic speed for both sets of sensors across the two road networks. This suggests that VDGCNTGA is capable of offering accurate predictions on large-scale road networks. Besides, both map figures also show that the very low or high traffic speed happens on several continuous sensors for both road networks. It indicates that traffic states at one location are influenced by its neighbours. Meanwhile, similar traffic speed at a time interval can be observed at sensors which are far away from each other. It indicates that traffic states may exist hidden network-wide spatial dependencies. Our model takes such dependency into consideration by generating a virtual dynamic road graph that describes the hidden and dynamic connections between road segments with respect to traffic states.

Parameter Visualization

In this section, we visualise and compare the learned spatial weights computed by our VDGCNTGA model against the physical road network to reveal the hidden spatial dependency. Figure 6.10 and Figure 6.11 show the adjacency matrix A_0 of the physical road network and the learned spatial weights, w_{dgcn} (cf. Eq. (6.13)), from the DGCN-Block of VDGCNTGA model for historical traffic in previous time intervals, X_T and in the same time interval with the targeted time interval from previous days, X_D on PEMS−BAY and METR−LA, respectively. For clarity, only the first 100 sensors are shown. Both x- and y-axis present the sensor ID. The colour describes the spatial dependency relationship (darker = more relevant). By comparing the sub-figure (b) against the sub-figure (c) in both Figure 6.10 and Figure 6.11, we see that the targeted traffic data is more dependent on T previous time intervals than on the same time interval from D previous days (i.e., darker shades in sub-figure (b) from both Figure 6.10 and Figure 6.11). In addition, the principal diagonals of the spatial weight matrices have significantly darker shade, indicating that traffic states are strongly related to its own historical traffic states. This is more

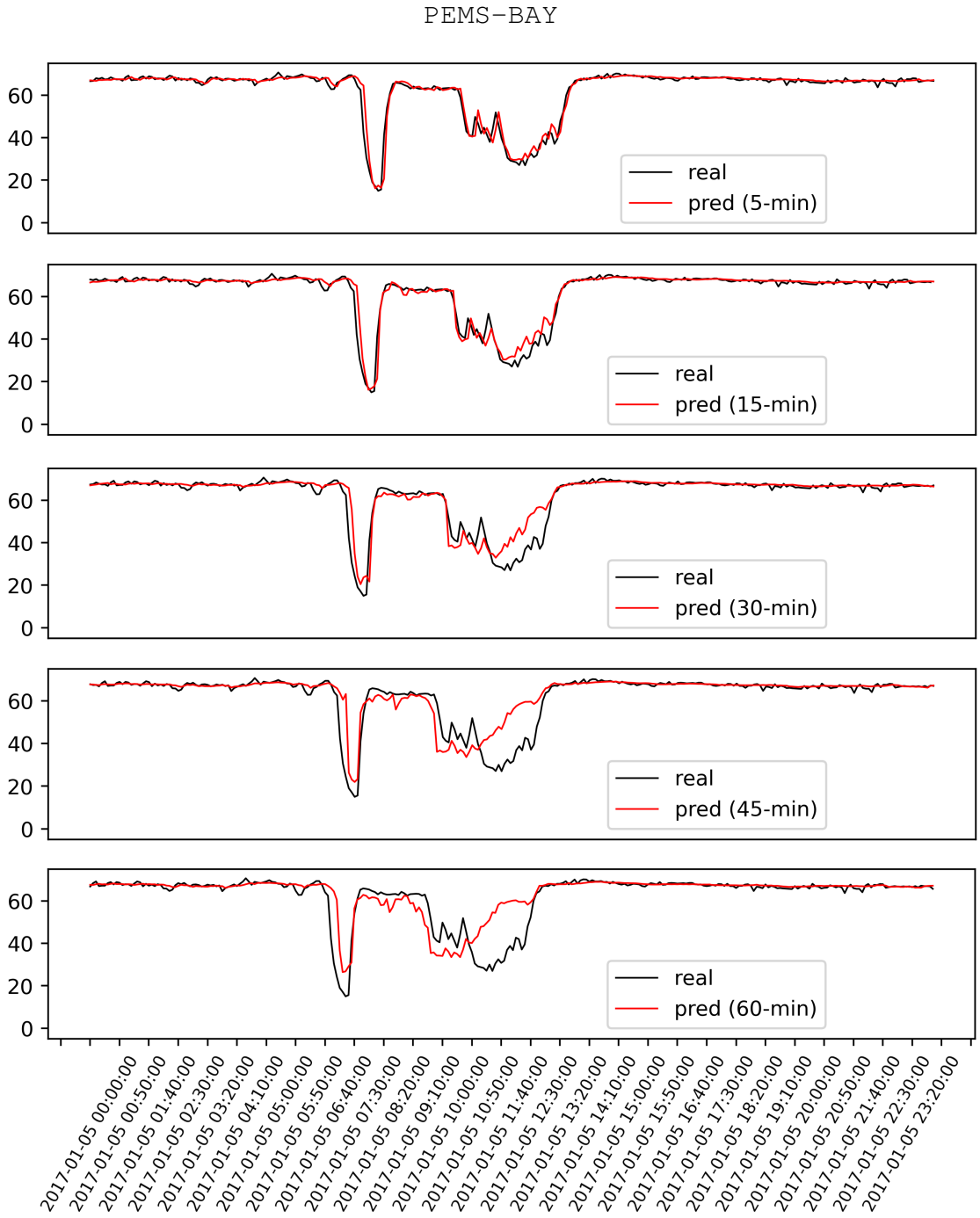


Figure 6.6: Real (black line) and predicted (red line) traffic speed $(miles/h)$ in a week with 288 ($= \frac{1days*24hours*60mins}{5mins}$) time intervals by VDGCNTGA on PEMS-BAY with a time interval = 5 mins. The x-axis represents the time and the y-axis is traffic speed. The prediction horizons are 5-min, 15-min, 30-min, 45-min and 60-min from top to bottom, respectively.

obvious in sub-figure (c) from both Figure 6.10 and Figure 6.11 since historical traffic states from neighbours in previous days have less impacts on traffic state in the targeted

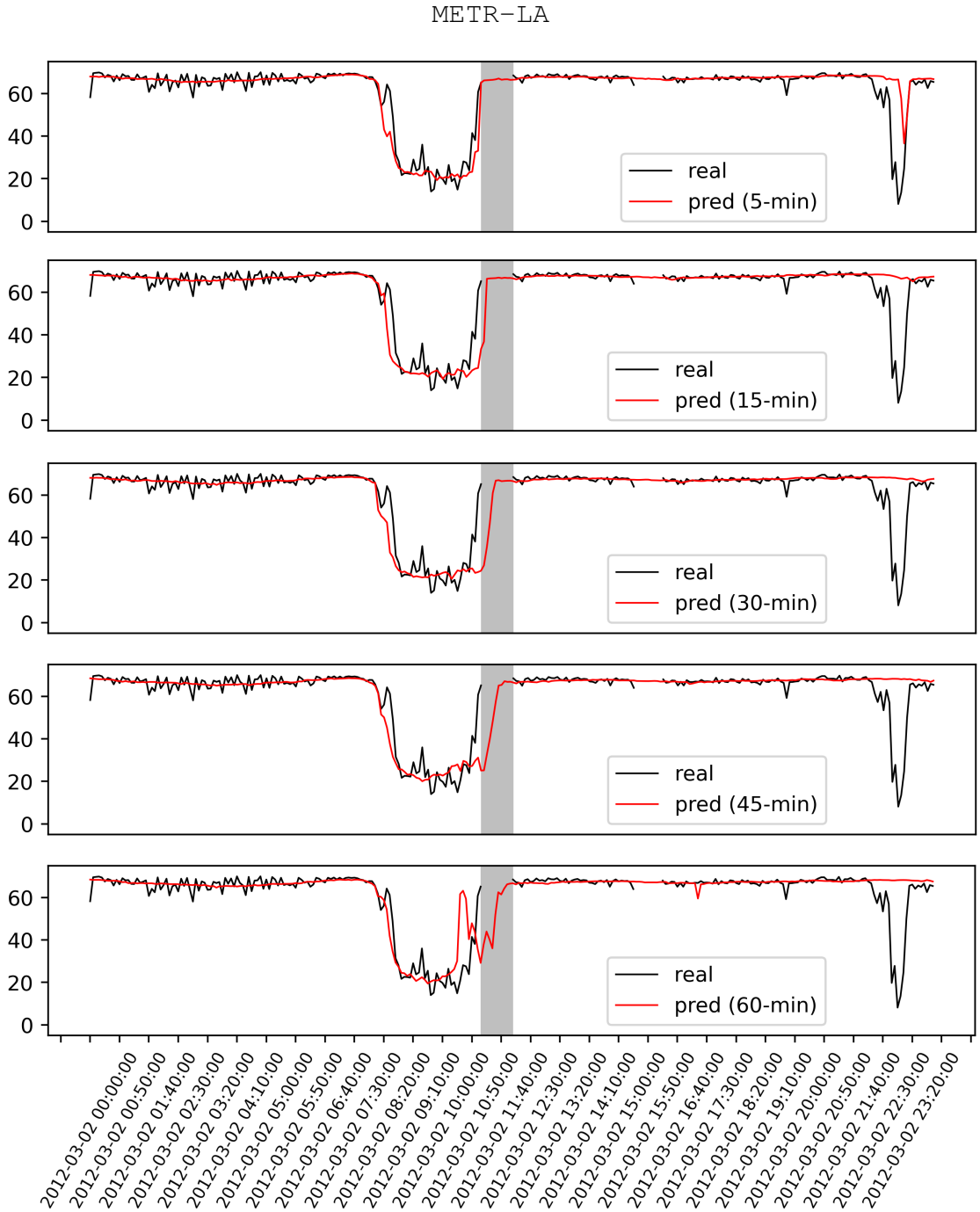
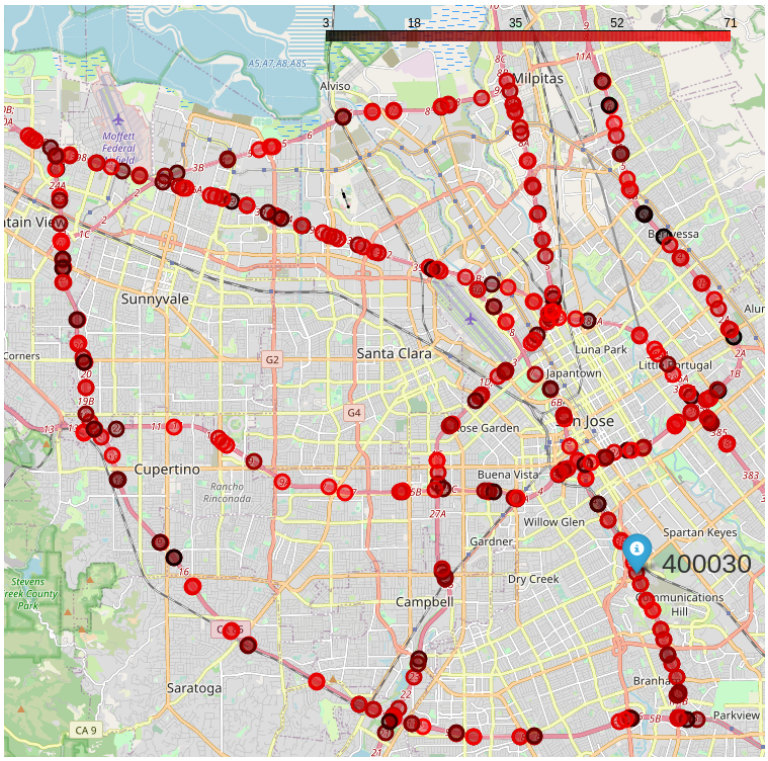
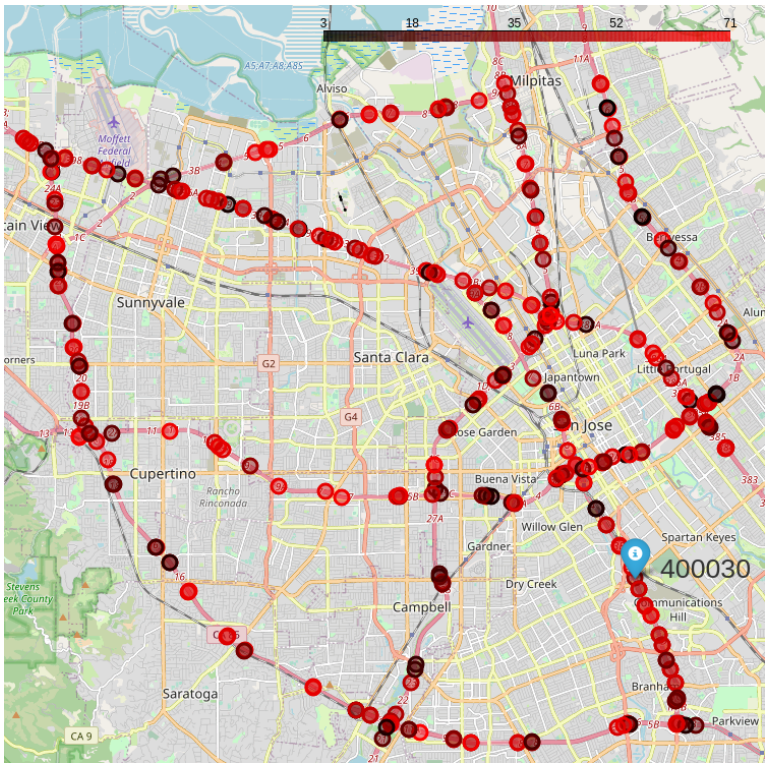


Figure 6.7: Real (black line) and predicted (red line) traffic speed $(miles/h)$ in a week with 288 $(= \frac{1days*24hours*60mins}{5mins})$ time intervals by VDGCNTGA on METR-LA with a time interval = 5 mins. The x-axis represents the time and the y-axis is traffic speed. The prediction horizons are 5-min, 15-min, 30-min, 45-min and 60-min from top to bottom, respectively.

sensor. Between the two road networks, sub-figure (b) for PEMS-BAY is darker than the sub-figure (b) for METR-LA. It indicates that the spatial dependencies of sensors in

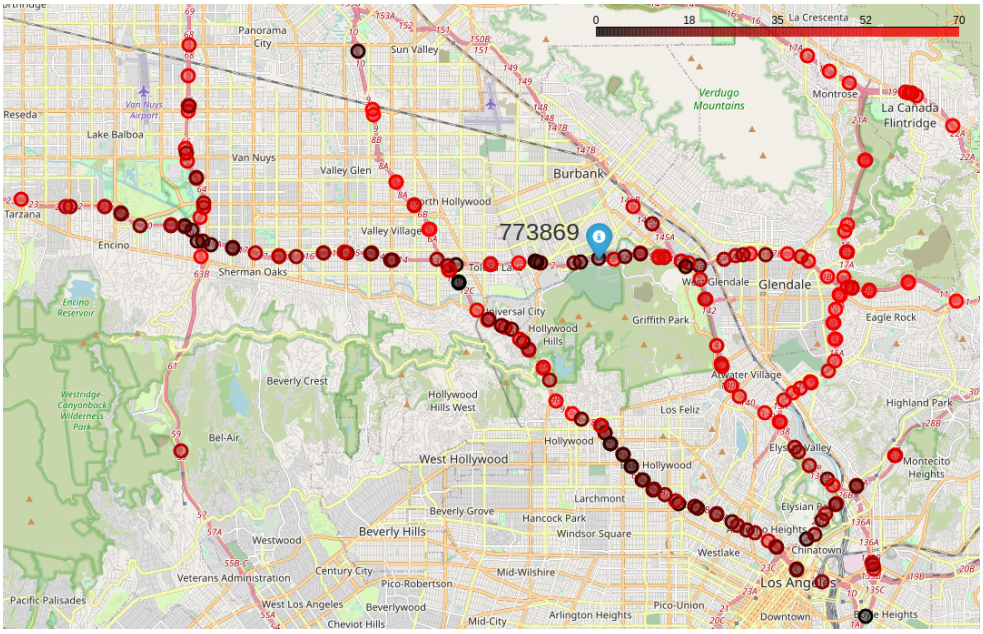


(a) Recorded real traffic speed

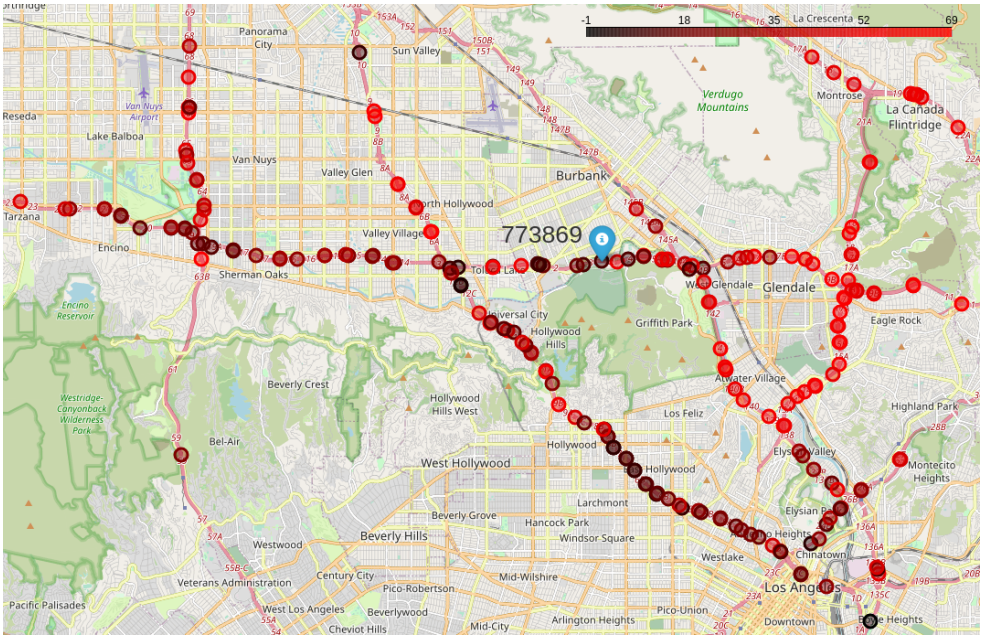


(b) Predicted traffic speed

Figure 6.8: Visualisation of real (top sub-figures) and predicted (bottom sub-figures) traffic speed at a time interval on the road network of `PEMS-BAY`. The lower traffic speed, the darker colour.



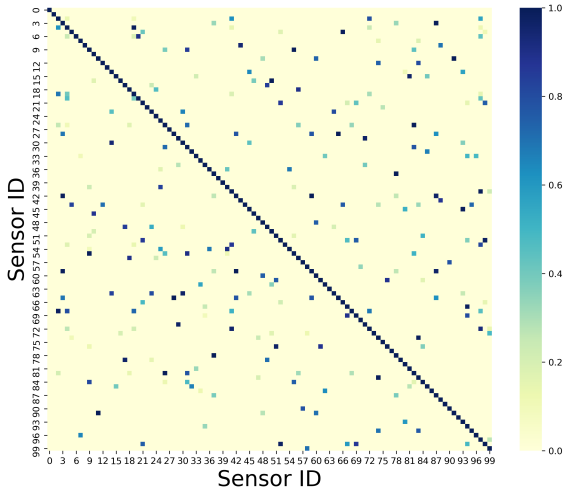
(a) Recorded real traffic speed



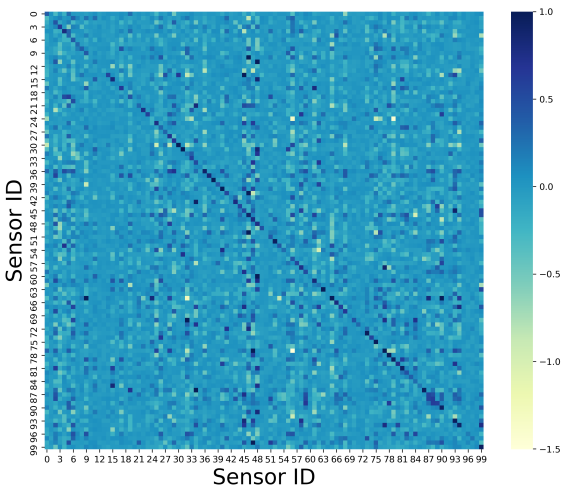
(b) Predicted traffic speed

Figure 6.9: Visualisation of real (top sub-figures) and predicted (bottom sub-figures) traffic speed at a time interval on the road network of METR-LA. The lower the traffic speed, the darker the colour.

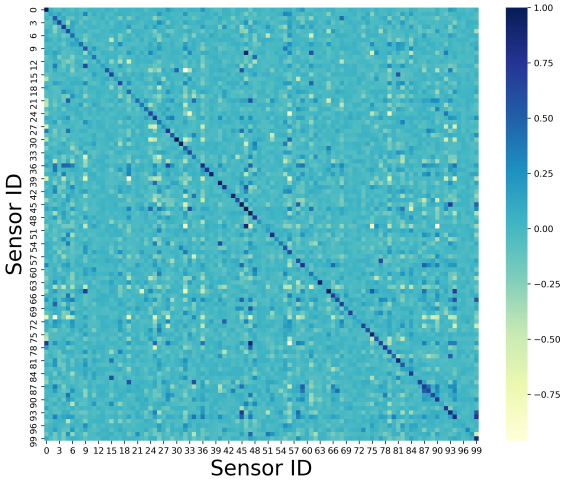
PEMS-BAY are more significant than METR-LA. In addition, from spatial weight matrices, some sensors, which are far away from the targeted sensor, still generate important impacts on the targeted sensor. It shows that traffic states of non-adjacent sensors affect each other.



(a) Adjacency matrix A_0



(b) Learned dynamic spatial weights for X_T



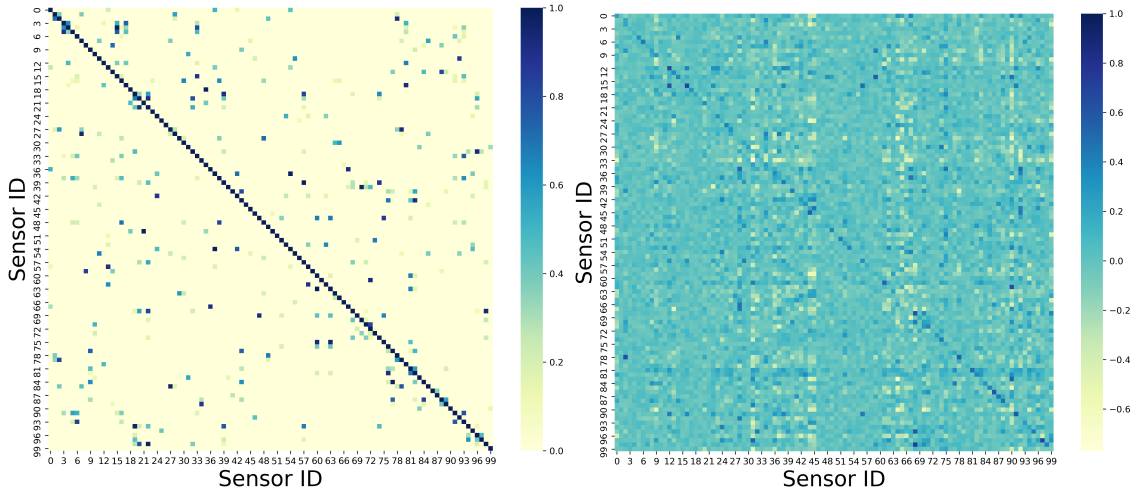
(c) Learned dynamic spatial weights for X_D

Figure 6.10: The adjacency matrix A_0 and learned dynamic spatial weight matrices for X_T and X_D are shown on {(a), (b), (c)} for PEMS−BAY. These matrices show the spatial dependencies of the first 100 sensors on both datasets. The colour indicates the spatial dependency relationship, the darker the more relevant.

Ablation Experiment

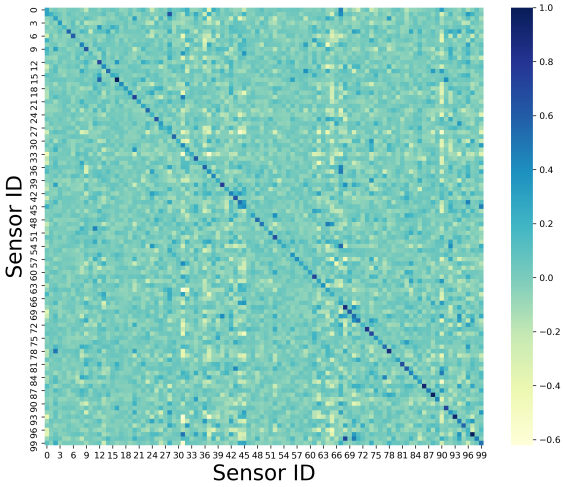
We further analyse and study our VDGCNTGA model via ablation experiments to study the contribution and importance of each block towards to the final prediction. For this, we create three variants that are built by removing one module from the proposed VDGC-NTGA as follows.

- **GCNTG**: This variant uses the adjacency matrix A_0 instead of the virtual dynamic spatial matrix A_{vd} generated by the Self-Attention Block in the training phase.



(a) Adjacency matrix A_0

(b) Learned dynamic spatial weights for X_T



(c) Learned dynamic spatial weights for X_D

Figure 6.11: The adjacency matrix A_0 and learned dynamic spatial weight matrices for X_T and X_D are shown on {(a), (b), (c)} for METR-LA. These matrices show the spatial dependencies of the first 100 sensors on both datasets. The colour indicates the spatial dependency relationship, the darker the more relevant.

Comparing the results between GCNTG and VDGCNTGA allows us to understand the contribution of the virtual dynamic graph proposed in this work to the final prediction.

- **VTGA**: This variant is developed by removing the DGCN-Block of VDGCNTGA and aimed at investigating the role played by the DGCN-Block in the full-fledged VDGCNTGA model.

- **VDGCNTGA***: For this variant, we remove the second STTras-Block in VDGCNTGA to understand the contribution of correcting learned features from DGCN-

Block by aligning them with the virtual dynamic spatial matrix A_{vd} in STm and the targeted temporal embedding features $T_{T'}$ in TTm.

TABLE 6.1 presents the results of our ablation experiments on the two real-world datasets for $T' = 6$ (i.e., 30-min prediction horizon). Overall, VDGCNTGA achieves the best performance in both networks. From the results, we observe the following:

- The virtual dynamic spatial matrix A_{vd}, that is generated by the Self-Attention Block to learn the dynamic spatial relationships of road segments, plays an important role on improving the prediction accuracy. This conclusion can be derived from the better result achieved by our VDGCNTGA over GCNTG. This observation further supports our design of VDGCNTGA in using virtual dynamic graph rather than relying only on the physical road connectivity.
- The DGCN-Block conducts convolution operation on virtual dynamic graph A_{vd}. It can further and efficiently learn dynamic spatial-temporal features and also join the spatial relations of historical traffic and future traffic into the learned features from the first STTras-Block. The contribution of the DGCN-block can be seen when comparing the prediction accuracy between the full VDGCNTGA and VTGA whereby VTGA performed 0.13% (`PEMS-BAY`) and 0.04% (`METR-LA`) worse without the DGCN-Block.
- The removal of the second STTras-Block results in the largest deterioration of the prediction accuracy. This can be observed by the difference in accuracy achieved by the three variants compared to the proposed VDGCNTGA whereby VGCNTGA* obtained the largest decrease in accuracy. This is due to the fact that VGCNTGA* loses (1) the functions of the virtual dynamic spatial matrix A_{vd} in STm to correct the learned spatially-fused features and (2) the targeted temporal embedding features $T_{T'}$ in TTm to build the external relationships of historical and future traffic data. Specifically, for the virtual dynamic spatial matrix, A_{vd}, by removing the second STTras-Block, the model can no longer correct the spatial features in the learned spatially-fused features from DGCN-Block.

6.4.3 Comparison Study

We now proceed to conduct a comparison study pitting our VDGCNTGA model against the following seven well-known baseline models.

Table 6.1: Comparison of results from different modules

Model Name	PEMS-BAY			METR-LA		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
(a) 30-min future prediction (T'=6)						
GCNTG	1.7085	3.76	4.6548	3.1933	8.96	6.5808
VTGA	1.6407	3.48	3.8876	**3.0853**	8.66	6.3775
VDGCNTGA*	1.7298	3.91	4.0023	3.1729	9.08	6.5595
VDGCNTGA	**1.5648**	**3.35**	**3.5141**	3.0971	**8.62**	**6.3267**

- GRU (Fu et al. 2016) is a variant of RNN such as LSTM. However, it has less gates than LSTM so as to allow faster computing while still retaining competitive performance against LSTM.

- CNN-LSTM (Cao et al. 2017) integrates CNN and LSTM modules for single-service traffic prediction and interactive network traffic prediction. CNN and LSTM analyze spatial and temporal dependencies, respectively.

- T-GCN (Zhao, Song, Zhang, Liu, Wang, Lin, Deng & Li 2019) uses GCN to learn complex topological structures in the space domain and GRU to learn dynamic changes of traffic data in the time domain.

- STGCN (Yu et al. 2018) composes of two spatial-temporal convolutional blocks (ST-Conv) and a fully-connected layer. Each ST-Conv block consists of two temporal gated convolution layers and a spatial graph convolution layer in the middle to mine spatial and temporal dependencies.

- ASTGCN (Guo et al. 2019) uses a spatial-temporal attention mechanism to analyze dynamic spatial and temporal features, a GCN for the spatial pattern analysis and a CNN for temporal feature analysis.

- DCRNN (Li et al. 2018) models traffic as a diffusion process on a weighted road graph and uses diffusion convolution neural network to learn spatial dependencies and recurrent neural network to learn temporal dependencies.

- GMAN (Zheng et al. 2020) utilizes the transform attention mechanism to model spatial and temporal dependencies in an encoder-decoder architecture. It builds a spatial-temporal embedding to model the graph structure and time information and embeds it into multi-attention mechanisms.

Figure 6.12 presents the prediction accuracy (100%-MAPE) of all models for both PEMS−BAY (Figure 6.12 (a)) and METR−LA (Figure 6.12 (b)). Overall, the prediction accuracy of all models is higher for PEMS−BAY. Our VDGCNTGA achieves the highest prediction accuracy on average – 96.77% prediction accuracy on average for PEMS−BAY and 91.68% for METR−LA, followed by GMAN (96.60% and 91.47% respectively) and DCRNN (96.05% and 91.39% respectively). At the other end of the spectrum, T-GCN is almost always the worst on both datasets and the average prediction accuracy is 93.71% for PEMS−BAY and 83.86% for METR−LA. The one probable reason is that only $1 - hop$ neighbour matrix is used in the GCN module of the T-GCN. As such, spatial features considered by T-GCN are restricted to relations of each sensor with its adjacent neighbours. In addition, the differences of performances among all models become larger over longer prediction horizons. This indicates that all the models including the simpler models (e.g. GRU) compute better predictions for shorter prediction horizons. For longer prediction horizons, more complex models with the abilities to analyse dynamic spatial and temporal dependencies are needed to obtain high prediction accuracy.

TABLE 6.2 compares the MAE, MAPE and RMSE across both datasets for the seven baseline models and our VDGCNTGA. All models perform better for PEMS−BAY than for METR−LA. For instance, their MAPEs are less than 8.00% across all prediction horizons for PEMS−BAY as opposed to under 20.00% for METR−LA. One reason for this is due to the missing data in the METR−LA dataset. From the results, we see that our VDGCNTGA mostly obtains the best results for all prediction horizons across both datasets with the exception for 5-min and 15-min prediction horizons where DCRNN narrowly outperforms our model on both datasets. Again, all types of errors achieved by all models increase when the prediction horizons become longer.

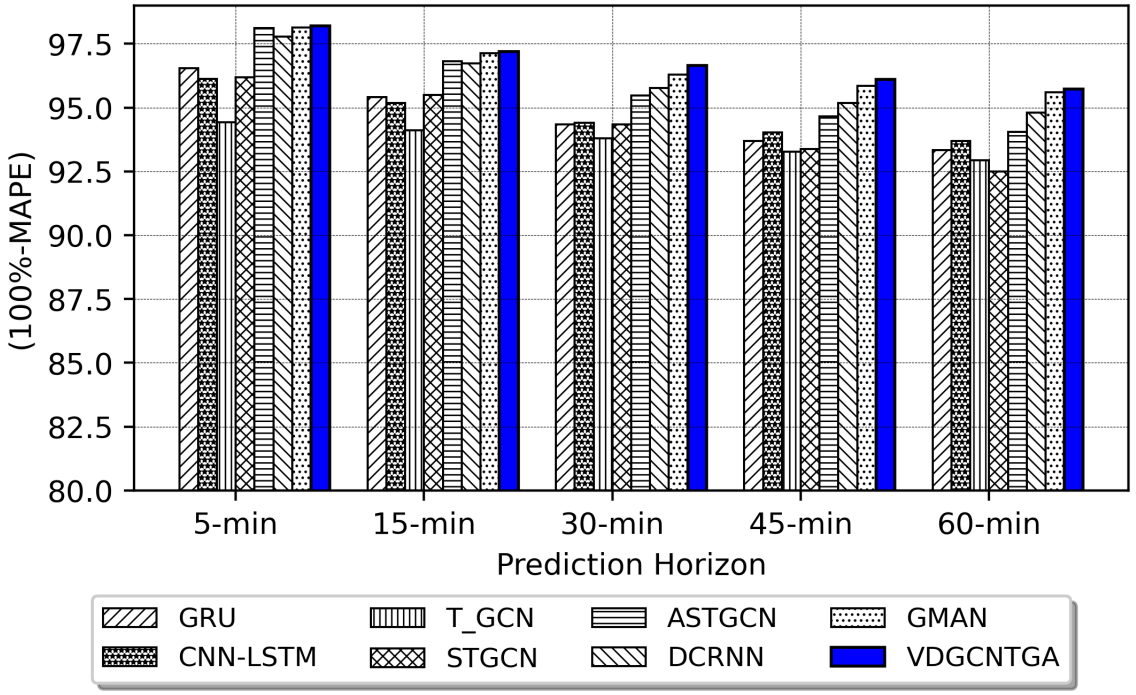
Table 6.2: Results achieved by all models for both datasets

Model Name	PEMS−BAY			METR−LA		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
(a) 5-min future prediction (T'=1)						
GRU	1.6913	3.45	2.8974	3.3369	7.83	5.8201
CNN-LSTM	1.8765	3.88	3.1958	3.3888	7.80	6.0269
T-GCN	2.4687	5.58	4.5619	4.9543	12.44	8.1131
STGCN	1.8384	3.81	3.1105	3.6787	8.22	6.5665

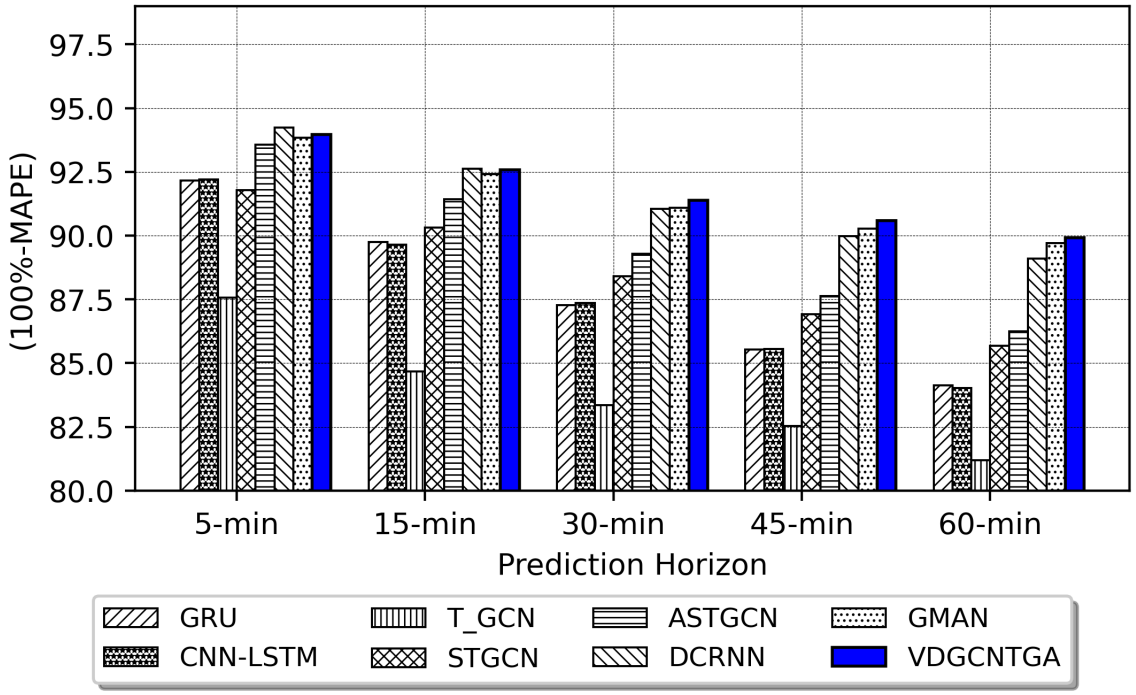
ASTGCN	0.9347	1.88	1.7710	2.4857	6.43	4.4114
DCRNN	**0.9170**	2.22	**1.6999**	**2.3325**	**5.77**	**4.0161**
GMAN	0.9493	1.86	1.7579	2.4445	6.16	4.4177
VDGCNTGA	0.9329	**1.81**	1.7064	2.3828	6.04	4.3205
(b) 15-min future prediction (T'=3)						
GRU	2.1303	4.58	3.7916	4.1811	10.25	7.6012
CNN-LSTM	2.1933	4.81	3.9610	4.3127	10.35	7.8845
T-GCN	2.5660	5.88	4.7532	6.1823	15.32	9.7222
STGCN	2.1103	4.51	3.7309	4.1964	9.69	7.9137
ASTGCN	1.4671	3.19	3.0985	3.0786	8.58	6.0340
DCRNN	**1.3434**	3.26	**2.8615**	**2.7775**	**7.38**	**5.3535**
GMAN	1.3533	2.86	2.9219	2.8269	7.58	5.6075
VDGCNTGA	1.3599	**2.80**	2.9192	2.7796	7.42	5.4790
(c) 30-min future prediction (T'=6)						
GRU	2.5033	5.66	4.5481	5.0948	12.72	9.1599
CNN-LSTM	2.4535	5.59	4.5984	5.1480	12.65	9.3875
T-GCN	2.6843	6.20	5.0002	6.8121	16.65	10.6894
STGCN	2.4834	5.66	4.6108	4.9270	11.59	9.2209
ASTGCN	1.9311	4.53	4.3212	3.6345	10.71	7.4193
DCRNN	1.6805	4.23	3.8619	3.1839	8.95	6.4789
GMAN	1.6422	3.71	3.7884	3.1499	8.90	6.5428
VDGCNTGA	**1.5648**	**3.35**	**3.5141**	**3.0971**	**8.62**	**6.3267**
(d) 45-min future prediction (T'=9)						
GRU	2.7262	6.31	4.9696	5.6871	14.46	10.0219
CNN-LSTM	2.5918	5.98	4.8798	5.8488	14.45	10.2587
T-GCN	2.8726	6.72	5.3505	7.1436	17.46	11.1565
STGCN	2.7979	6.63	5.2562	5.5858	13.09	10.1572
ASTGCN	2.2188	5.34	5.0119	4.0540	12.36	8.3200

DCRNN	1.8706	4.82	4.3794	3.4493	10.02	7.1828
GMAN	1.7885	4.14	4.1667	3.3485	9.72	7.0905
VDGCNTGA	**1.7584**	**3.90**	**4.0445**	**3.3014**	**9.42**	**6.8548**
(e) 60-min future prediction (T'=12)						
GRU	2.8705	6.66	5.2104	6.2478	15.88	10.6667
CNN-LSTM	2.6922	6.31	5.0876	6.4159	15.98	10.8960
T-GCN	2.9401	7.06	5.5091	7.7986	18.81	11.9359
STGCN	3.0762	7.50	5.7680	6.1128	14.32	10.8037
ASTGCN	2.4518	5.96	5.5218	4.4138	13.75	9.0173
DCRNN	1.9998	5.20	4.6791	3.6755	10.91	7.7280
GMAN	1.8857	4.41	4.3726	3.4815	10.30	7.4331
VDGCNTGA	**1.8769**	**4.27**	**4.3248**	**3.4577**	**10.08**	**7.2180**

From TABLE 6.2, on PEMS−BAY, VDGCNTGA, GMAN, DCRNN and ASTGCN are always the top four best performers across all prediction horizons. Among them, VDGCNTGA always achieves better results as the prediction horizon is increased from 15-min to 60-min. This owes to the virtual dynamic road graph A_{vd} comprehensively mining the hidden spatial dependencies of road segments as well as the spatial- and temporal-transformers in VDGCNTGA. For the remaining models, (GRU, CNN-LSTM, T-GCN and STGCN), GRU performs better than others and achieves 3.45% MAPE for 5-min prediction horizon mainly due to its effective function on long-term dependency analysis. T-GCN is the worst performing model with 2.4687, 5.58% and 4.5619 recorded for MAE, MAPE and RMSE respectively. For 30-min, 45-min and 60-min prediction horizons, CNN-LSTM is better than GRU, T-GCN and STGCN. This is likely due to the spatial features extracted by CNN module in CNN-LSTM where it becomes more important for longer prediction horizons. In addition, the 1D CNN layers in the CNN module enable traffic information from all sensors in the road network to contribute to the targeted sensor, as opposed to T-GCN and STGCN that only consider traffic information from several neighbours. Note that the spatial features extracted by 1D CNN in CNN-LSTM and GCN in T-GCN and STGCN can be considered as global and local spatial features, respectively. To some degree, the global spatial features offer better performance than local spatial features but with high computation cost.



(a) PEMS−BAY



(b) METR−LA

Figure 6.12: The prediction accuracy (100% - MAPE) of all models for different prediction horizons.

For METR−LA, the rank of all considered models is similar to PEMS−BAY but the accuracies are worse due to the missing data issue. Similar to PEMS−BAY, the top four models

are still VDGCNTGA, GMAN, DCRNN and ASTGCN. For 5-min and 15-min prediction horizons, DCRNN obtains 5.77% and 7.38% of MAPE followed by our VDGCNTGA with 6.04% and 7.42%. For other prediction horizons, VDGCNTGA is consistently the best. This is because VDGCNTGA not only uses more features from previous time intervals and the same time interval with the targeted interval in previous days to analyse spatial and temporal dependencies, it also generates a dynamic road graph (A_{vd}) to fully mine the hidden and non-uniform spatial relations among sensors or road segments in the network. These can efficiently take into account the sudden changes caused by the missing data and incidents which are considered as big challenges for others. For the other four models (i.e., GRU, CNN-LSTM, T-GCN and STGCN), T-GCN remains the worst performing model with its MAPE increases from 12.44% for 5-min prediction horizon to 18.81% for 60-min prediction horizon. STGCN is almost always the best among these four for all prediction horizons.

6.4.4 Prediction Accuracy *vs* Computation Time

We have thus far shown the performance of the different models in terms of prediction accuracy. In this section, we turn our attention to the computation time. For this purpose, we run a set of experiments comparing our VDGCNTGA with STGCN, ASTGCN, DCRNN and GMAN. To ensure the fairness, for all models, we set the batch size as 18 in the training phase and run them on the same machine equipped with a GeForce RTX 2080 Ti GPU card with 11 GB Memory and 1545MHz Boost Clock. We obtain the average computation time of seven runs. We present the training time of four baselines and our VDGCNTGA for both datasets in TABLE 6.3. From the table, we observe that the training time for both STGCN and ASTGCN are much shorter than the other models. However, the prediction accuracy of these two models are much lower than the other three, especially when the prediction horizons are large. Among top three models in terms of accuracy (i.e., DCRNN, GMAN and our VDGCNTGA), VDGCNTGA appears to be the most efficient compared to the other two. The longest training time of DCRNN is caused by the sequence learning in RNN while, for GMAN, the longer training time is due to the setting of the number of ST-Attention Block as $L = 3$ (Zheng et al. 2020) that incur high computation cost. Our experiment results suggest that our proposed model, VDGCNTGA, offers the best tradeoff between accuracy and computation time.

Table 6.3: Computation time of four baselines and our proposed model when the batch size is set as 18.

	Training Time (seconds/epoch)	
Model Name	PEMS-BAY	METR-LA
STGCN	196.72	104.09
ASTGCN	119.07	67.64
DCRNN	662.13	384.56
GMAN	834.71	472.59
VDGCNTGA	516.63	187.95

6.5 Chapter Summary

In this chapter, we present a novel deep learning model, VDGCNTGA, for addressing the traffic prediction problem on large-scale road networks. Considering the complex and dynamic spatial dependencies of traffic at road segments hidden within the road networks, exploring these hidden and dynamic spatial dependencies is important for achieving high prediction accuracy. Instead of purely relying on the use of the adjacency matrix and other neighbourhood matrices that describe the physical connectivity between road segments, we developed an algorithm in the training phase of VDGCNTGA to generate a virtual dynamic road graph that comprehensively mine the hidden and dynamic spatial dependency of the road network.

We designed a framework for our VDGCNTGA based on GCN and the attention mechanism-based Transformers to analyse temporal and spatial dependencies with correction. We trained and tested our VDGCNTGA on two large-scale real-world road networks: PEMS-BAY and METR-LA. We compared it against seven well-known models in literature including: GRU, CNN-LSTM, T-GCN, STGCN, ASTGCN, DCRNN and GMAN. To further gain insights into the characteristic behavior of our model, we conducted ablation experiments and compared the full-fledged VDGCNTGA against three of its variants that are built with a removal of one module from the full model.

The experimental results indicate that our proposed model, VDGCNTGA, obtains the best performance (average accuracy $\approx 96.77\%$ for PEMS-BAY and $\approx 91.68\%$ for METR-LA) and for almost all prediction horizons, only being closely challenged at the shortest prediction horizon (i.e., at 5-min and 15-min horizons). The MAEs, MAPEs and RMSEs on PEMS-BAY are less than 1.8800, 4.30% and 4.3300, and less than 3.4600, 10.10% and 7.2200 on METR-LA, respectively. These results indicate that our VDGCNTGA can effi-

ciently improve the prediction accuracy on large-scale road networks, even on the dataset suffering from missing data such as that in METR-LA.

Chapter 7

Long-term Traffic Speed Prediction on Large-Scale Road Networks

7.1 Introduction

This chapter focuses on solving the long-term traffic prediction problem on urban transport networks and aims to predict traffic speed in the whole urban road network. Compared to short-term traffic prediction, long-term traffic prediction offer different challenges since the target prediction time has weaker relation to previous traffic states. We designed a novel deep learning framework, based on the Sequence-to-Sequence architecture with an embedded module, named GCNT-Seq2Seq, for long-term traffic speed prediction with high prediction accuracy. The embedded module uses Graph Convolution Neural Network, a memory-less approach, for the local-spatial dependency analysis by conducting convolution operation on the $k - hop$ neighbourhood matrix, while utilises Transformer as a memory-less approach for the global-spatial dependency analysis by implementing the attention mechanism that assigns individual weights to neighbour detectors for contributing to the targeted detector. The sequence-to-sequence architecture considered as a memory-based approach is built to analyse temporal dependencies of the spatially-fused time series from the embedded module. To evaluate our proposed model and compare it against existing well-known ones, we use real traffic speed dataset with missing data and frequent traffic incidents to train and test the models. The experimental results indicate that our proposed framework achieves the most accuracy prediction, even obtaining more than 80% accuracy for predicting traffic two hours in advance.

The rest of this chapter is organised as follows. We elaborate our approach for long-term

traffic prediction on the large-scale road network, including the problem definition, the architecture of our proposed model and its embedded module, in Section 7.2. Furthermore, real-world traffic data collected from a large road network is described in Section 7.3 while the performances of our model compared against several existing models are shown in Section 7.4. Finally, Section 7.5 summarises our research work for long-term traffic prediction on the large-scale road network.

7.2 Methodology

7.2.1 Problem Definition

The traffic prediction problem in this Chapter considers large-scale road network with N detectors. The road graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes with $|\mathcal{V}| = N$. $\mathcal{E}$ is the set of edges representing physical connectivity between detectors. Generally, $\mathcal{G}$ can be represented by $A \in \mathbb{R}^{N \times N}$, the $N \times N$ symmetric adjacency matrix, with its element $A_{i,j} = 1$ if there exists a link between node i and j and 0 otherwise. Considering that the future traffic states of a detector are influenced by its own historical states, the road graph, $\mathcal{G}$, is represented by $\tilde{A} = (A + I_N) \in \mathbb{R}^{N \times N}$ where I_N is the $N \times N$ identity matrix. $\tilde{A}$ only describes the connectivity of neighbours one hop away from each node (i.e., $1 - hop$ neighbourhood). Due to considering that traffic speed propagates downstream while traffic congestion propagates upstream and the dataset used in this work includes traffic congestion, we introduce the notion of $k - hop$ neighbourhood to represent the set of nodes that are reachable within k hops from the targeted node and define the $k - hop$ neighbourhood matrix as $\tilde{A}^k \in \mathbb{R}^{N \times N}$.

The traffic speed data from the road network with N detectors is described as $v_t = \{v_t^1, v_t^2, \ldots, v_t^i, \ldots, v_t^{N-1}, v_t^N\}; x_t \in \mathbb{R}^N, (i = 1, 2, 3, \ldots, N)$, and v_t^i denotes the traffic speed measured at detector i at t^{th} time interval. Typically, a time interval can represent 5, 15, 30, 45 and 60 minutes (Bickel et al. 2007). In this chapter, the dataset used for evaluating the proposed model considers 5 minutes as the time interval. Then $V \in \mathbb{R}^{T \times N}$ (cf. Eq. (7.1)) represents traffic speed data collected from N detectors in the network for T previous time intervals. Conversely, the traffic speed for the future is written as $V' = \{v_{t+1}, v_{t+2}, \ldots, v_{t+T'}\} \in \mathbb{R}^{T' \times N}$ where T' is the prediction horizon. Generally, traffic prediction problems can be categorised into short- ($T' < 30$ minutes) and long-term ($T' \geq 30$ minutes). Since we aim to solve the long-term traffic prediction problem on large-scale road network, this chapter covers timescales $T' = \{6, 12, 18, 24\}$ corre-

sponding to $\{30, 60, 90, 120\}$ minutes.

$$\begin{aligned} V = \{v_{t-T+1}, v_{t-T+2}, \ldots, v_{t-1}, v_t\}; \\ V \in \mathbb{R}^{T \times N}, T = 1, 2, 3, \ldots \end{aligned} \tag{7.1}$$

Based on traffic speed data and the road graph described above, the traffic prediction problem for the proposed approach in this chapter can be formulated as Eq. (7.2).

$$\tilde{V}' = \boldsymbol{F}\Big(V; \mathcal{G}(\mathcal{V}, \mathcal{E}, \tilde{A}^k)\Big) \tag{7.2}$$

where the objective is to learn the mapping function $F(.)$ and compute the traffic speed data in the next T' time intervals based on the historical traffic speed data in T previous time intervals and the road graph $\mathcal{G}$. For fitting the training phase, both input V and targeted V' need to be formatted to $V \in \mathbb{R}^{B \times T \times N}$ and $V' \in \mathbb{R}^{B \times T' \times N}$ where B is the batch size.

7.2.2 The GCNT-Seq2Seq Model

Figure 7.1 presents the framework of the proposed deep learning model, GCNT-Seq2Seq, that is designed to comprehensively analyse the spatial and temporal dependencies of traffic speed data. GCNT-Seq2Seq is developed under the sequence-to-sequence architecture consisting of an encoder and a decoder for the long-term dependency analysis. The encoder is used to learn the historical information and encodes it to a context vector. The decoder is utilised to decode the context vector for the final prediction. The modules of the sequence-to-sequence architecture is built based on Graph Convolution Neural Networks (GCN) and the Transformer for analysing spatial dependencies from original traffic data and the road graph data. The following details the GCNT-Seq2Seq model on the spatial dependency analysis and the temporal dependency analysis in detail, respectively.

GCNT as the embedded module of the sequence-to-sequence architecture is used to analyse the spatial dependencies. Figure 7.2 shows the GCNT module that consists of L_g graph convolutional neural (GCN) layers and L_s transformer layers in parallel. Considering the ${l_g}^{th}$ GCN layer at the t^{th} time interval as an example, the convolution operation is conducted on the $k - hop$ neighbourhood matrix $\tilde{A}^k$ joined to the output of the $(l_g - 1)^{th}$ GCN layer for obtaining spatial features using Eq. (7.3).

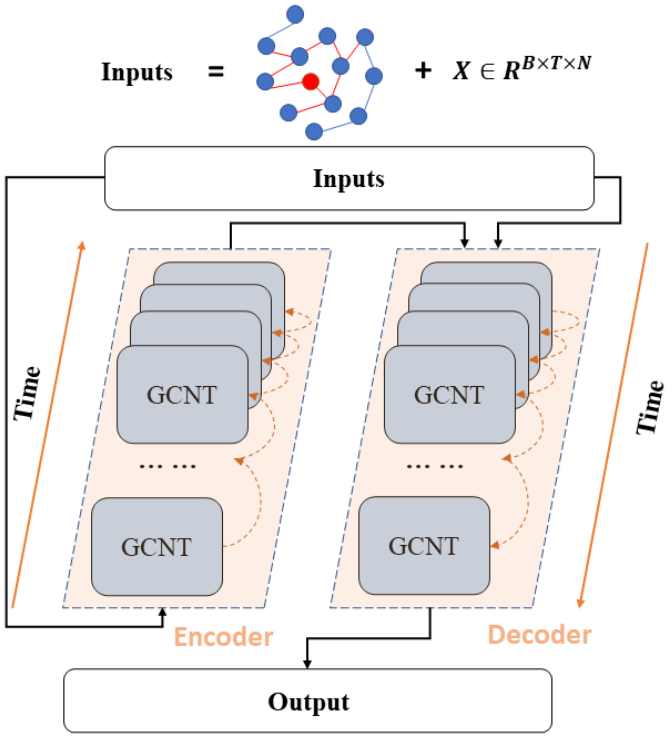


Figure 7.1: The framework of GCNT-Seq2Seq.

$$GCN_{l_g;t} = ReLU\left((W_{l_g;t} * \tilde{A}^k)GCN_{(l_g-1);t}\right) + GCN_{(l_g-1);t} \tag{7.3}$$

where $GCN_{(l_g-1);t} \in \mathbb{R}^{B\times N}$ is the output of the $(l_g-1)^{th}$ GCN layer and is treated as the input of the ${l_g}^{th}$ GCN layer. $GCN_{l_g;t} \in \mathbb{R}^{B\times N}$ is the output of the ${l_g}^{th}$ GCN layer and the input of the 1^{st} GCN layer is $v_t \in \mathbb{R}^{B\times N}$. $W_{l_g;t} \in \mathbb{R}^{N\times N}$ is the weight matrix of $\tilde{A}^k$ and $ReLU$ is the activation function of GCN layers. Due to $\tilde{A}^k$ describing connections of detectors in the $k-hop$ neighbourhood, obtained spatial features from the GCN layers can be considered as local spatial features.

Meanwhile, transformer layers analyse spatial dependencies by the attention mechanism to assign individual weights to neighbours of the targeted detector so as to contribute to the targeted detector. Notes that this spatial dependency analysis can provide spatial features, that may be missed by GCN layers, for the proposed model to achieve high prediction accuracy. Each transformer layer consists of three fully-connected layers, an attention layer and a linear layer. Considering the ${l_s}^{th}$ transformer layer at the t^{th} time interval as an example, three fully-connected layers are used to generate the multi-head inputs of queries $Q_{l_s;t} \in \mathbb{R}^{B\times(H\times d_q)\times N}$, keys $K_{l_s;t} \in \mathbb{R}^{B\times(H\times d_k)\times N}$ and values $V_{l_s;t} \in \mathbb{R}^{B\times(H\times d_v)\times N}$ by

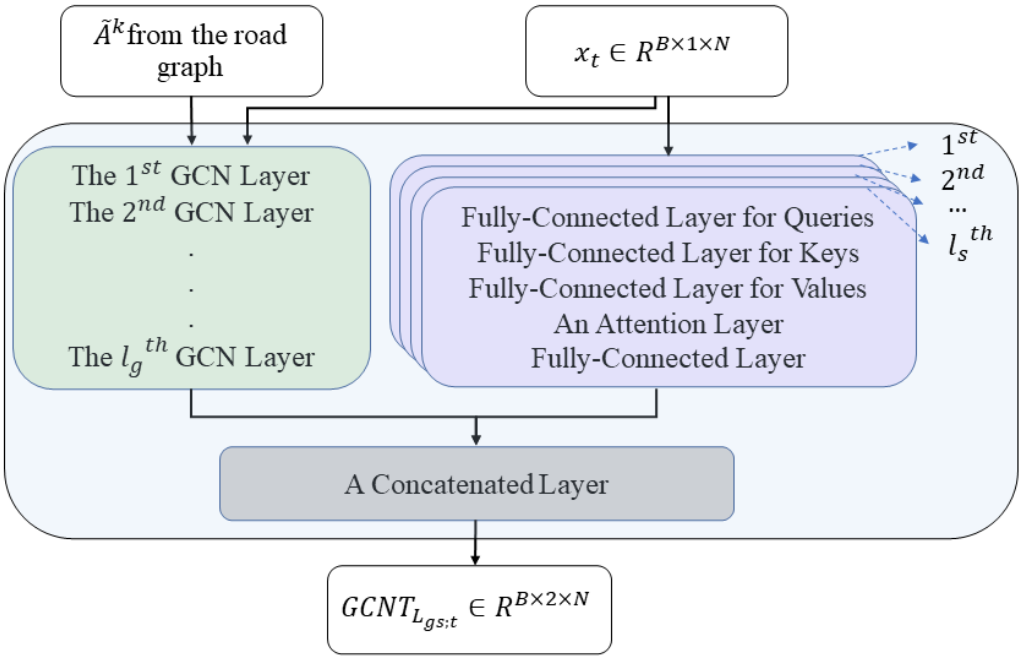


Figure 7.2: The module of GCNT

Eq. (7.4), in which H is the number of multi-heads and d_q, d_k and d_v are the number of embedded features for $Q_{l_s;t}$, $K_{l_s;t}$ and $V_{l_s;t}$, respectively.

$$\begin{aligned} Q_{l_s;t} &= W^q_{l_s;t} S_{(l_s-1);t}{}^{\intercal} \\ K_{l_s;t} &= W^k_{l_s;t} S_{(l_s-1);t}{}^{\intercal} \\ V_{l_s;t} &= W^v_{l_s;t} S_{(l_s-1);t}{}^{\intercal} \end{aligned} \tag{7.4}$$

The spatial weight matrix is generated by multiplying the transposition of $K_{l_s;t}$ by $Q_{l_s;t}$ and then utilised to obtain spatial features using Eq. (7.5).

$$S_{l_s;t} = ReLU\left(W^s_{l_s;t}\left(softmax(\frac{Q_{l_s;t}K_{l_s;t}{}^{\intercal}}{\sqrt{d_k}})V_{l_s;t}\right) + b^s_{l_s;t}\right) + S_{(l_s-1);t} \tag{7.5}$$

where $S_{(l_s-1);t}{}^{\intercal} \in \mathbb{R}^{B\times N}$ is the transposition of the output of the $(l_s-1)^{th}$ transformer layer as the input of the $l_s{}^{th}$ transformer layer. S_{l_s} is the output of the $l_s{}^{th}$ transformer layer and the input of the 1^{st} transformer layer is v_t. $W^q_{l_s;t}$, $W^k_{l_s;t}$ and $W^v_{l_s;t}$ are weight matrices of queries, keys and values, respectively. $W^s_{l_s;t}$ is the weight matrix of the linear layer and $b^s_{l_s;t}$ is the related bias. $ReLU$ is an activation function. Due to the spatial weight matrix

$\frac{Q_{ls;t}K_{ls;t}{}^{\intercal}}{\sqrt{d_k}} \in \mathbb{R}^{B\times H\times N\times N}$ enabling all other detectors to have individual weights for the targeted detector, spatial features obtained here are considered as global spatial features.

Finally, the output of the last GCN layer, $GCN_{L_g;t}$, and the output of the last transformer layer, $S_{L_s;t}$, are concatenated, and then pass to a linear layer to generate local-global spatial features $GCNT_{L_{gs};t}$ using Eq. (7.6).

$$GCNT_{L_{gs};t} = W_{L_{gs};t}concat(GCN_{L_g;t}, S_{L_s;t}) + b_{L_{gs};t} \tag{7.6}$$

where $W_{L_{gs};t}$ and $b_{L_{gs};t}$ are the weight matrix and the bias, respectively.

In addition, the residual connection network (He et al. 2016) is used in each GCN and transformer layer to ensure stable training and also supplement the important information hidden in negative values that are neglected by the $ReLU$ activation function.

Seq2Seq consists of LSTMs as the encoder and the decoder, which is used to embed our GCNT module for extracting and delivering temporal features from spatially-fused features. Taking the t^{th} time interval as an example, the encoder takes the $GCNT_{L_{gs};t}$ as the input and encodes the spatially-fused features using Eq. (7.7).

$$\begin{aligned} f_t &= \sigma_g(w_f \cdot GCNT_{L_{gs};t} + u_f \cdot h_{t-1} + b_f) \\ i_t &= \sigma_g(w_i \cdot GCNT_{L_{gs};t} + u_i \cdot h_{t-1} + b_i) \\ o_t &= \sigma_g(w_o \cdot GCNT_{L_{gs};t} + u_o \cdot h_{t-1} + b_o) \\ c_t &= f_t * c_{t-1} + i_t * \sigma_c(w_c \cdot GCNT_{L_{gs};t} + u_c \cdot h_{t-1} + b_c) \\ h_t &= o_t * \sigma_h \times (c_t) \end{aligned} \tag{7.7}$$

where w_f, w_i, w_o and w_c are the weights of the forget gate f_t, the input gate i_t, the output gate o_t and the cell state c_t respectively while b_f, b_i, b_o and b_c are the corresponding biases for each gate and the cell state. Furthermore, u_f, u_i, u_o and u_c are the weights of the last hidden state h_{t-1}. σ_g denotes a sigmoid function (= $\frac{1}{1+e^{-x}}$) in three gates and the operator $*$ denotes Hadamard product. σ_c and σ_h are hyperbolic tangent function ($tanh(x)$) for the cell state and the final output. h_t is considered as the output of the encoder and carries historical information. In the decoder, the output of the encoder h_t is treated as the initialised hidden state and $GCNT_{L_{gs};t}$ is still considered as the input. The output of the decoder is the final prediction.

7.3 Data Description

Dataset used for evaluating the proposed model and comparing our model against well-known existing models is collected from the real-world road network, named `METR-LA` (Li et al. 2018). The detailed information about this dataset can be found in Section 2.6.2.

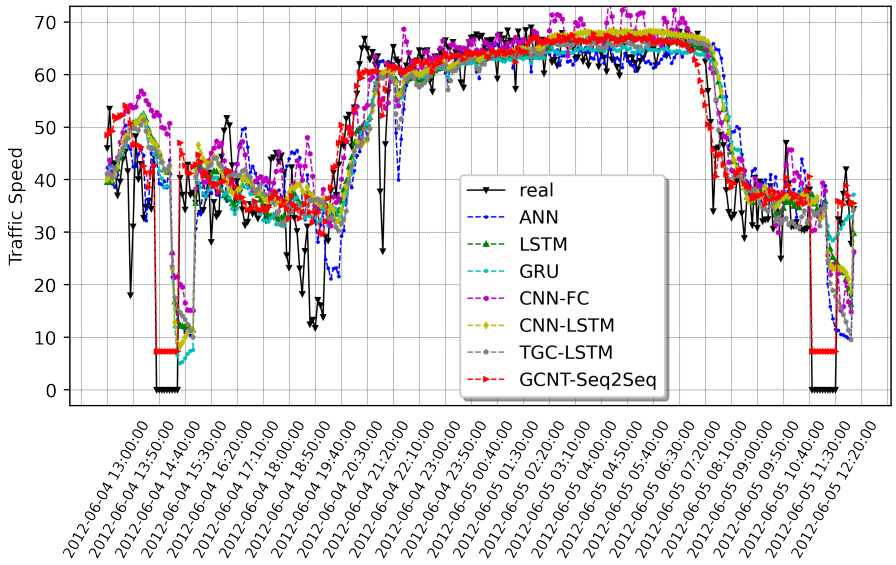
7.4 Experiments

7.4.1 Parameter Settings

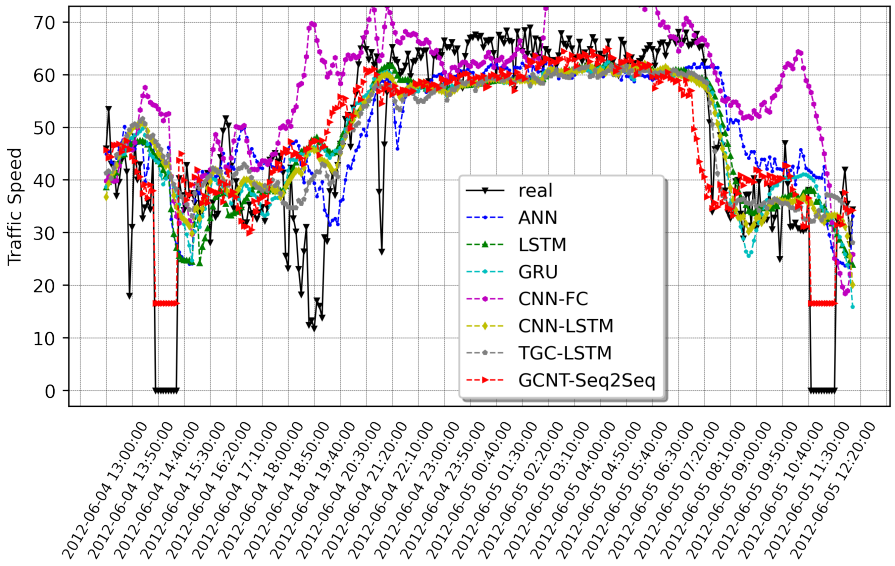
To optimise the GCNT-Seq2Seq model and obtain high prediction accuracy, there are several hybrid-parameters that need to be tuned including historical time intervals T, targeted time intervals T', the number of multi-heads H, the number of embedded features $\{d_q, d_k, d_v\}$, the learning rate r, batch size B and the number of epochs. Based on the well-known existing works (Zheng et al. 2020), the historical time intervals T, targeted time intervals T', the number of multi-heads H, the number of embedded features $\{d_q, d_k, d_v\}$, and batch size B are set as 12, 24, 8, {8, 8, 8} and 32, respectively. For the learning rate setting, generally, a too small learning rate will make a training algorithm converge slowly while a too large learning rate will make the algorithm diverge. Therefore, finding an optimal learning rate is very important to improve the performance of the algorithm. However, the experimental method to find the best learning rate is time-consuming. In this work, we use Cyclical Learning Rates (CLR) (Smith 2017) to optimise the learning rate. Using this method, we have set the learning rate to $1.02e^{-03}$. In addition, *stop early* strategy is used to optimise the number of epochs and avoid the problem of over-fitting. The training process will be stopped when the training loss continues to decrease in 10 epochs while the validation loss increases. Finally, we follow the convention and use 70% of the dataset for training, 10% for validation and 20% for testing.

7.4.2 Results and Discussion

To evaluate the proposed model, six baseline models are used in our comparison experiments. They include 1) one linear feature-based model, ANN (Csikós et al. 2015); 2) two temporal feature-based models, LSTM (Kang et al. 2017) and GRU (Fu et al. 2016); 3) one spatial feature-based model, CNN-FC (Ma et al. 2017); and 4) two spatial and



(a)



(b)

Figure 7.3: Real (black color) and predicted (other colours) traffic speed $(miles/h)$ in a day with 288 $(= \frac{1days*24hours*60minutes}{5minutes})$ time intervals from all models on METR-LA with a time interval = 5 minutes. The x-axis represents the time and the y-axis is traffic speed (miles/hour). The prediction horizons are 30 minutes in (a) and 120 minutes in (b), respectively. The red lines in (a) and (b) represent predicted traffic speed from our GCNT-Seq2Seq.

Table 7.1: Experimental results from all models on METR-LA

Model Name	METR-LA		
	MAE(T'=6/12/18/24)	MAPE(T'=6/12/18/24)	RMSE(T'=6/12/18/24)
ANN	5.0232 / 6.7519 / 8.1165 / 9.1839	13.12% / 18.20% / 21.68% / 24.42%	9.8455 / 11.8983 / 13.1064 / 13.8129
LSTM	5.2023 / 6.6489 / 7.6383 / 8.2251	12.74% / 16.44% / 18.68% / 21.21%	9.4332 / 11.1926 / 12.2611 / 12.8663
GRU	5.0715 / 6.2108 / 7.0753 / 7.7387	12.67% / 15.64% / 18.17% / 20.03%	**9.1238** / **10.6058** / 11.7414 / 12.4671
CNN-FC	6.4447 / 8.0232 / 8.4805 / 9.4339	14.77% / 18.40% / 19.91% / 21.88%	9.7793 / 11.5983 / 12.0401 / 13.0291
CNN-LSTM	5.2192 / 6.4425 / 7.5352 / 8.3240	12.70% / 16.03% / 18.79% / 20.96%	9.3857 / 10.9365 / 12.1356 / 12.8291
TGC-LSTM	5.2822 / 6.6194 / 7.6074 / 8.2736	12.62% / 15.99% / 18.64% / 20.54%	9.5018 / 11.1085 / 12.1906 / 12.8627
GCNT-Seq2Seq	**4.9956** / **6.1774** / **6.9358** / **7.5813**	**12.10%** / **15.38%** / **17.44%** / **19.41%**	9.2731 / 10.8090 / **11.6876** / **12.3108**

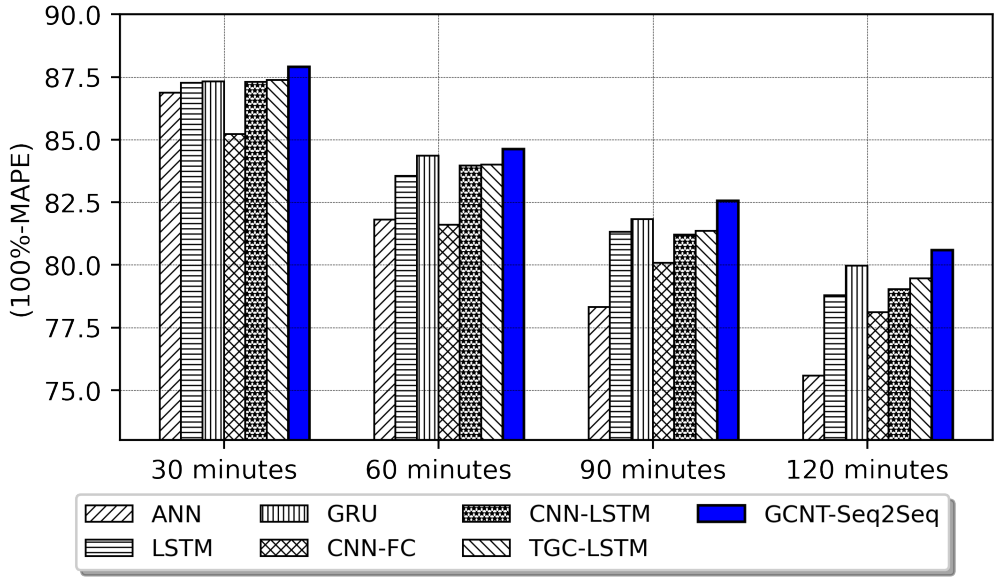


Figure 7.4: The prediction accuracy (100%-MAPE) from all comparison models.

temporal feature-based models, CNN-LSTM (Liu et al. 2017) and TGC-LSTM (Cui et al. 2019).

Figure 7.3 presents the real (black line) and predicted (other colours) traffic speed in a day by the GCNT-Seq2Seq model and other six baseline models. Figure 7.3 (a) and (b) consider 30 minutes and 120 minutes as prediction horizons, respectively. Overall, GCNT-SeqSeq can efficiently predict the trend of traffic speed changes, even for longer prediction horizons. Besides, GCNT-Seq2Seq also captured the sudden changes caused by missing data or traffic incidents.

TABLE 7.1 presents experimental results from all models considered for different prediction horizons. $T' = \{6, 12, 18, 24\}$ are corresponding to $\{30, 60, 90, 120\}$ minutes as prediction horizons. It is observed that our proposed model, GCNT-Seq2Seq, achieves the best performance among all considered models. Its MAEs, MAPEs and RMSEs for all prediction horizons are almost always the lowest. Hence, the prediction accuracy of GCNT-Seq2Seq is the highest, which can be also observed in Figure 6.12 that shows the prediction accuracy (100%-MAPE) of all models. From Figure 7.4, the differences of prediction accuracy between GCNT-Seq2Seq and other models become more obvious for longer prediction horizons. The advantage of our model is higher for longer prediction horizons. Even when for 120 minutes as prediction horizon, the prediction accuracy of GCNT-Seq2Seq is still more than 80% and it is the only one achieving over 80% accuracy among all models.

Furthermore, from TABLE 7.1, the linear feature-based model, ANN, that relies on the

linear relationship of traffic speed data in time domain for prediction, almost always obtains the worst results except for 30 minutes prediction horizon where the spatial feature-based model, CNN-FC, is the worst one. The probable reason is that linear relationships for longer prediction horizons are not obvious while the spatial dependencies among road network become more important. Therefore, CNN-FC model that mainly focuses on spatial feature extraction by the convolution kernels performs better than ANN for longer prediction horizons. Between two temporal feature-based models, LSTM and GRU, GRU always outperforms LSTM for all different prediction horizons. This conclusion is in agreement with the results in (Chung et al. 2014). Both CNN-LSTM and TGC-LSTM achieve similar results, likely because they are able to analyse both spatial and temporal features for final prediction.

7.4.3 Prediction Accuracy *vs* Computation Time

We have shown the performance of all competed models by prediction accuracy above. In this section, we turn our attention to the computation time. For this purpose, we run a set of experiments comparing our GCNT-Seq2Seq with other six competed existing models. To ensure the fairness, for all models, we set the batch size as 32 in the training phase and run them on the same machine equipped with a GeForce RTX 2080 Ti GPU card with 11 GB Memory and 1545MHz Boost Clock. We obtain the average computation time of ten runs. We present the training time of six baselines and our GCNT-Seq2Seq in TABLE 7.2. From the table, we observe that the training time for both ANN is much shorter than the other six models. However, its prediction accuracy are lower than five of the other six for 30 minutes prediction and much lower than all other models for large prediction horizons (i.e., 90 and 120 minutes predictions). Our GCNT-Seq2Seq model costs much more time to run due to the large number of parameters of Transformer and GCN modules, compared to all competed models. However, its prediction accuracy is highest among all, especially for large prediction horizons. Considering that this model is used for long-term prediction to support efficient traffic strategy making that does not have the demand of quick responding, the accuracy is more importance. If both the quick responding and accuracy were requested, GRU could be best choice.

Table 7.2: Computation time of six baselines and our proposed model when the batch size is set as 32.

	Training Time (seconds/epoch)
Model Name	METR-LA
ANN	2.19s
LSTM	9.16s
GRU	8.23s
CNN-FC	16.93s
CNN-LSTM	11.71s
TGC-LSTM	14.46s
GCNT-Seq2Seq	51.59s

7.5 Chapter Summary

In this chapter, the problem of predicting long-term traffic on large-scale road network is addressed and a novel deep learning framework, named GCNT-Seq2Seq, is developed. The proposed framework takes the advantages of Graph Convolution Network (GCN) and Transformer on the different spatial dependency analysis and the advantage of the Sequence-to-Sequence (Seq2Seq) architecture on the temporal dependency analysis. GCN is used to extract the local spatial features by operating convolution on the $k - hop$ neighbourhood matrix and Transformer is utilised to capture the global spatial features by assigning individual weights to neighbours of the targeted detector so as to contribute to the targeted detector. The concatenation of local and global spatial features is embedded into the Seq2Seq architecture for temporal feature extraction and final prediction. The proposed framework is compared against existing well-known models using real world dataset with missing data and frequent traffic incidents. Among all the models being compared, our proposed model shows the best performance and can achieve more than 80% accuracy even for predicting two hours traffic status in advance.

Chapter 8

Conclusions

In this chapter, we summarise the research work and our contributions in this thesis by a logic map that shows our idea and logic regarding this research work and achieve the corresponding aim in Section 8.1. Furthermore, we also provide future research directions in Section 8.2.

8.1 Summary

In this thesis, we are motivated by many traffic problems around the world such as serious traffic congestion, long travel time and frequent traffic accidents. We aim to build novel deep learning models for traffic prediction on urban transport networks which can be used to support and optimise the performance of ITSs so as to reduce traffic congestion, accidents and long travel time. We summarise our research in this context in Figure 8.1.

After reviewing the literature, we found two common ways to formulate traffic prediction problems which we adopt in our work. However, while the problems are general, there exists a broad range of solutions adopting different methodologies including statistical models, machine learning models and deep learning models. In our work, we have adopted the deep learning approach.

In our first work, we targeted short-term traffic flow prediction problem for linear roadways. This problem, in comparison with the rest of the our work, presented the least challenge and offers a suitable stepping stone towards the more complex version of the problem. For this, we designed an Ensemble Model (EM) that can analyse and extract

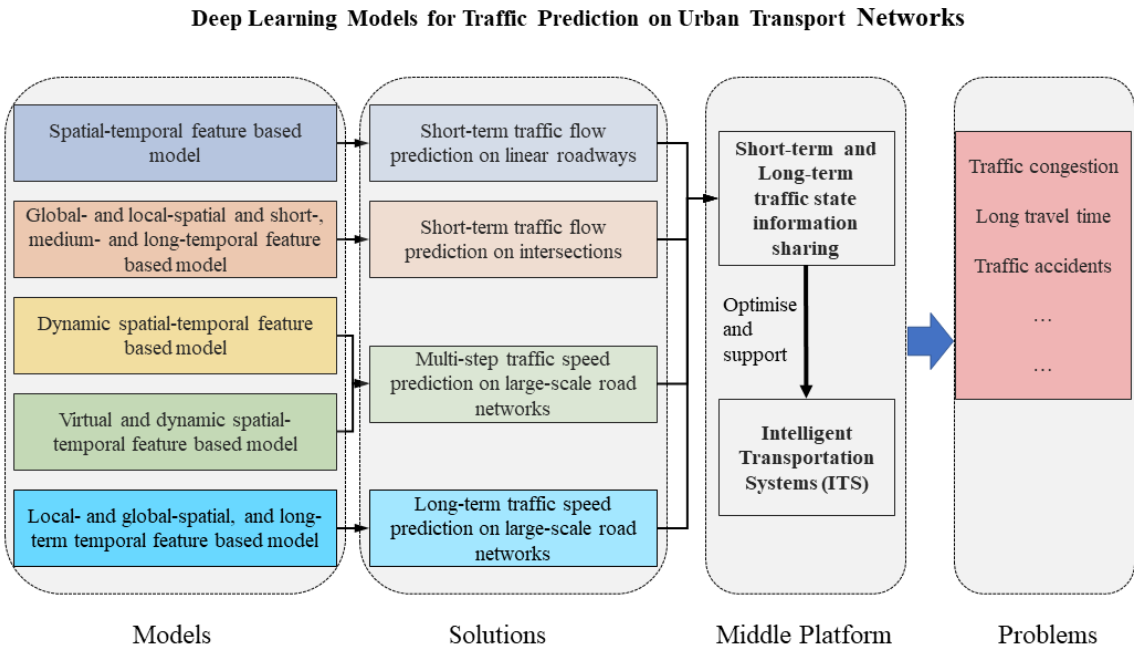


Figure 8.1: Summary logic map.

spatial-temporal features from original traffic flow data via LSTM, DAE and CNN modules for the final prediction. We collected real-world traffic data from two different places to evaluate our EM.

We then naturally proceeded to focus on short-term traffic flow prediction on intersections, in which traffic situations are more complex than linear roadways because of multiple entries and exits possibly impacted by roundabouts and/or traffic lights. For this, we developed a novel deep learning model, named ALLSCP, based on ARIMA, LSTM, SAE and CAPSNET modules. It is capable of analysing and extracting more detailed features including global- and local-spatial, short-, medium- and long-temporal features for the final prediction on intersections. ALLSCP is then evaluated on two real-world traffic datasets collected from two different locations.

Next, instead of focusing on a specific road segment, we extended and conducted traffic prediction on entire large-scale road networks and with multi-step predictions. Considering the network as a whole proved to be a much more complex task. Here, we first built a novel deep learning model (named SAGCN-SST) based on the latest deep learning technologies: GCN and the attention mechanism, for multi-step traffic speed prediction on large-scale road networks. The architecture of SAGCN-SST is designed for extracting dynamic-spatial and temporal features for the final prediction, specifically. The two large datasets with different traffic patterns are used to evaluate our SAGCN-SST.

After developing SAGCN-SST model, we further found that the hidden spatial depen-

dencies of road segments in road networks can be exploited to improve prediction accuracy. Therefore, we proposed a novel Virtual Dynamic Graph Convolution Network and Transformer-based model with Gated and Attention mechanisms (VDGCNTGA) by considering hidden spatial dependencies under real road networks and dynamic spatial-temporal features from original traffic data. Similar to our evaluation of the SAGCN-SST model, two large datasets are utilised to evaluate VDGCNTGA.

Finally, we moved on to consider long-term traffic prediction problems, again for large-scale road networks. This present different challenges to the model since the dependencies of the traffic at the targeted time is weaker (i.e., it is further in the unknown future). This research is important for ITS to devise long traffic control strategies/policies rather than reactionary responses to immediate traffic conditions or incidents. For this, we proposed a novel DL model, named GCNT-Seq2Seq, for long-term traffic prediction, which can capture the long-term dependencies of traffic data. Similar to other developed models, we used a large-scale road network dataset to evaluate GCNT-Seq2Seq model.

All models we built are compared against well-known models in the literature to validate the better performances of our models. For the more complex models involving higher number of constituent modules (i.e., ALLSCP, SAGCN-SST, VDGCNTGA), we also conducted ablation experiments to gain further insights into the properties of these models as well as to explain how and why our proposed models are able to perform better than the competing state-of-the-art models.

8.2 Future Research Directions

Traffic prediction as an important element of ITSs to solve traffic problems such as traffic congestion, accidents and long travel time is a very relevant topic in this era especially with increasing urbanisation. Many researchers have contributed their efforts to this topic but there are still challenges remained to be solved.

Based on the literature as well as our own work, we found that spatial and temporal features are most important features for traffic prediction. Focusing on analysing and extracting those features can predict traffic states in the future and also provide accurate prediction. However, we believe there is further possibilities to extract more detailed spatial and temporal features to further improve prediction accuracy. In line with this, we see that the following could further contribute to this line of research:

- For spatial feature analysis, most of the latest works in literature use the attention

mechanism to allocate different weights to neighbouring sensors so as to account for the different level of impacts of neighbouring sensors on the targeted sensor. However, the size of the neighbourhood is fixed for all sensors on the road network. In the real world, the size of the neighbourhood is dynamic. For example, at the same time interval and under serious traffic congestion, the size of the neighbourhood should be larger than its under smooth traffic situations. This is because more vehicles and/or road segments are affected by serious traffic congestion. Furthermore, at different time intervals, the size of the neighbourhood for the same targeted sensor should be treated differently because traffic state always changes over time. Therefore, how to define a dynamic neighbourhood for each sensor is one challenging part to predict traffic situation in the real world when analysing spatial dependencies.

- For temporal feature analysis, most of existing works consider the fixed length of historical traffic data to predict future traffic data on a sensor or road segment. The fact is that traffic state on a smooth road segment relies on the more immediate historical traffic states and, on a busy road segment, it is affected by longer periods of historical traffic states. Therefore, the length of historical traffic data used to predict future traffic data should be dynamic for each sensor on road networks. A model that can adaptively adjust the number of previous time epochs into consideration for making the final prediction could further improve the achieved accuracy.

- For dynamic road graph building, most of existing works consider the entire road network as a graph and then use graph convolution neural network on this physic road graph to analyse the spatial dependencies of road segments. This can help model analyse the spatial dependencies of the road network but not fully exploit hidden spatial dependencies due to constantly changing of traffic networks. To solve this problem, the dynamic road graph can be generated and used for further improving traffic prediction accuracy. The reinforcement learning can be potential method to generate dynamic graphs when considering that the reinforcement learning is capable of representing hard constraints by the design of environment dynamics and reward function (Peng et al. 2021). A sequence of dynamic graphs can be generated by the action sequence and help the model learn the complex and hidden spatial dependencies of road segments.

Combining accurate traffic prediction capabilities with other traffic management functions could enable ITSs or relevant stakeholders to offer novel traffic services to users. For this, we see two interesting research directions:

- Dynamic Traffic path rerouting – the idea of combining traffic prediction on large-scale road networks with path rerouting is a potentially useful one for travellers (Sarker et al. 2018, Li, Fu, Yuan, Zhang, Chen, Yang & Yang 2019, Wang et al. 2019, Chan et al. 2021). Most current navigation applications and devices suggest travel path based on the shortest distance and avoiding hitting congestion. When a user selects a path and unfortunately ended up in an emerging serious traffic congestion or accidents on the road, the navigator often could not change the path automatically based on real or emerging traffic situations on the road network. From this view, traffic prediction could be a part of path rerouting in the navigator. Traffic prediction can be used for finding vulnerable road segments in road networks and then a rerouting algorithm can be developed and embedded in the navigator to reroute path in real time.

- Traffic light optimisation – traffic prediction could be embedded into the control of traffic light either specifically focusing on a single intersection or more ambitiously the coordination of the entire traffic light systems in a city area to facilitate smooth flow of traffic predictively (Kim & Jeong 2020, Shengdong et al. 2019).

Bibliography

Ahmed, E., Saint, A., Shabayek, A. E. R., Cherenkova, K., Das, R., Gusev, G., Aouada, D. & Ottersten, B. (2018), 'A survey on deep learning advances on different 3d data representations', *arXiv preprint arXiv:1808.01462* .

Ahmed, M. S. & Cook, A. R. (1979), 'Analysis of freeway traffic time-series data by using box-jenkins techniques', pp. 1–9.

Alumni, K. (n.d.), 'Didi: Improving efficiency and road traffic with big data'. (last accessed: 06.07.2021).
URL: *https://digital.hbs.edu/platform-digit/submission/didi-improving-efficiency-and-road-traffic-with-big-data/*

Bahdanau, D., Cho, K. H. & Bengio, Y. (2015), Neural machine translation by jointly learning to align and translate, *in* '3rd International Conference on Learning Representations, ICLR 2015'.

Bastings, J., Titov, I., Aziz, W., Marcheggiani, D. & Sima'an, K. (2017), Graph convolutional encoders for syntax-aware neural machine translation, *in* 'Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing', pp. 1957–1967.

Bickel, P. J., Chen, C., Kwon, J., Rice, J., Van Zwet, E. & Varaiya, P. (2007), 'Measuring traffic', *Statistical Science* pp. 581–597.

Blandin, S., Salam, A. & Bayen, A. (2012), Individual speed variance in traffic flow: analysis of bay area radar measurements, *in* 'Transportation Research Board 91st Annual Meeting'.

Bottou, L. (2010), Large-scale machine learning with stochastic gradient descent, *in* 'Proceedings of COMPSTAT'2010', Springer, pp. 177–186.

Box, G. E., Jenkins, G. M., Reinsel, G. C. & Ljung, G. M. (1970), *Time series analysis*, John Wiley & Sons.

Cai, P., Wang, Y., Lu, G., Chen, P., Ding, C. & Sun, J. (2016), 'A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting', *Transportation Research Part C: Emerging Technologies* **62**, 21–34.

Caltrans, C. (n.d.), 'An introduction to the california department of transportation performance measurement system (pems)'. (last accessed: 10.11.2018).
URL: *http://pems.dot.ca.gov/*

Cao, X., Zhong, Y., Zhou, Y. & et.al. (2017), 'Interactive temporal recurrent convolution network for traffic prediction in data centers', *IEEE Access* **6**, 5276–5289.

Castro-Neto, M., Jeong, Y.-S., Jeong, M.-K. & Han, L. D. (2009), 'Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions', *Expert Syst. with App.* **36**(3), 6164–6173.

Chan, R. K. C., Lim, J. M.-Y. & Parthiban, R. (2021), 'A neural network approach for traffic prediction and routing with missing data imputation for intelligent transportation system', *Expert Systems with Applications* **171**, 114573.

Chen, C. (1994), 'Freeway performance measurement system (pems)', *Public Roads* **57**(3), 8–14.

Chen, C., Li, K., Teo, S. G., Zou, X., Wang, K., Wang, J. & Zeng, Z. (2019), Gated residual recurrent graph neural networks for traffic prediction, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 33, pp. 485–492.

Chen, Y. (2015), Convolutional Neural Network for Sentence Classification, PhD thesis, University of Waterloo.

Chen, Z., Liu, Y. & Liu, S. (2017), Mechanical state prediction based on lstm neural netwok, *in* '2017 36th Chinese Control Conference (CCC)', IEEE, pp. 3876–3881.

Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K. & Bengio, Y. (2015), Attention-based models for speech recognition, *in* 'Advances in neural information processing systems', pp. 577–585.

Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. (2014), Empirical evaluation of gated recurrent neural networks on sequence modeling, *in* 'NIPS 2014 Workshop on Deep Learning, December 2014'.

Costello, Z. & Martin, H. G. (2018), 'A machine learning approach to predict metabolic pathway dynamics from time-series multiomics data', *NPJ systems biology and applications* **4**(1), 1–14.

Csikós, A., Viharos, Z. J., Kis, K. B., Tettamanti, T. & Varga, I. (2015), Traffic speed prediction method for urban networks—an ann approach, *in* '2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)', IEEE, pp. 102–108.

Cui, Z., Henrickson, K., Ke, R. & Wang, Y. (2019), 'Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting', *IEEE Transactions on Intelligent Transportation Systems* **21**(11), 4883–4894.

Cui, Z., Ke, R., Pu, Z. & Wang, Y. (2018), 'Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction', *arXiv e-prints* pp. arXiv–1801.

Cui, Z., Lin, L., Pu, Z. & et.al. (2020), 'Graph markov network for traffic forecasting with missing data', *Transportation Research Part C: Emerging Technologies* **117**, 102671.

Cuturi, M. (2011), Fast global alignment kernels, *in* 'Proceedings of the 28th international conference on machine learning (ICML-11)', pp. 929–936.

Diehl, F., Brunner, T., Le, M. T. & Knoll, A. (2019), Graph neural networks for modelling traffic participant interaction, *in* '2019 IEEE Intelligent Vehicles Symposium (IV)', IEEE, pp. 695–701.

Ding, A., Zhao, X. & Jiao, L. (2002), Traffic flow time series prediction based on statistics learning theory, *in* 'Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems', IEEE, pp. 727–730.

Do, L. N., Vu, H. L., Vo, B. Q., Liu, Z. & Phung, D. (2019), 'An effective spatial-temporal attention based neural network for traffic flow prediction', *Transportation research part C: emerging technologies* **108**, 12–28.

Duan, Z., Yang, Y., Zhang, K., Ni, Y. & Bajgain, S. (2018), 'Improved deep hybrid networks for urban traffic flow prediction using trajectory data', *IEEE Access* **6**, 31820–31827.

Dunne, R. & Campbell, N. (1997), On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function, *in* 'Proc. Aust. Conf. on the Neural Networks'.

England, H. (n.d.), 'Highways england network journey time and traffic flow data'. (accessed: 21.11.2018).
URL: *https://data.gov.uk/dataset/9562c512-4a0b-45ee-b6ad-afc0f99b841f/highways-england-network-journey-time-and-traffic-flow-data*

Essien, A. E., Petrounias, I., Sampaio, P. & Sampaio, S. (2019), Deep-presimm: Integrating deep learning with microsimulation for traffic prediction, *in* '2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)', IEEE, pp. 4257–4262.

Estrach, J. B., Zaremba, W., Szlam, A. & LeCun, Y. (2014), Spectral networks and deep locally connected networks on graphs, *in* '2nd International Conference on Learning Representations, ICLR 2014'.

Everaers, R. & Kremer, K. (1994), 'A fast grid search algorithm for molecular dynamics simulations with short-range interactions', *Computer Physics Communications* **81**(1-2), 19–55.

Fang, M., Tang, L., Yang, X., Chen, Y., Li, C. & Li, Q. (2022), 'Ftpg: A fine-grained traffic prediction method with graph attention network using big trace data', *IEEE Transactions on Intelligent Transportation Systems* **23**(6), 5163–5175.

Fei, X., Lu, C.-C. & Liu, K. (2011), 'A bayesian dynamic linear model approach for real-time short-term freeway travel time prediction', *Transportation Research Part C: Emerging Technologies* **19**(6), 1306–1318.

Finn, C. et al. (2016), Deep spatial autoencoders for visuomotor learning, *in* 'IEEE Int'l Conf. on Robotics and Automation (ICRA)'.

Fu, R., Zhang, Z. & Li, L. (2016), Using lstm and gru neural network methods for traffic flow prediction, *in* '31st Youth Academic Annual Conference of Chinese Association of Automation', IEEE, pp. 324–328.

Ghosh, B., Basu, B. & O'Mahony, M. (2007), 'Bayesian time-series model for short-term traffic flow forecasting', *Journ. Transp. Eng.* **133**(3), 180–189.

Government, N. (n.d.), 'Tlc trip record data'. accessed: 21.06.2021.
URL: *https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page*

Grover, A. & Leskovec, J. (2016), node2vec: Scalable feature learning for networks, *in* 'Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining', pp. 855–864.

Guo, S., Lin, Y., Feng, N., Song, C. & Wan, H. (2019), Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 33, pp. 922–929.

Habtemichael, F. G. & Cetin, M. (2016), 'Short-term traffic flow rate forecasting based on identifying similar traffic patterns', *Transp. Res. Part C Emerg. Technol.* **66**, 61–78.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

He, Z., Chow, C.-Y. & Zhang, J.-D. (2019), Stcnn: A spatio-temporal convolutional neural network for long-term traffic prediction, *in* '2019 20th IEEE International Conference on Mobile Data Management (MDM)', IEEE, pp. 226–233.

Hedges, C. (n.d.), 'How many cars are there in the world in 2021'. last accessed: 07.06.2021.
URL: *https://hedgescompany.com/blog/2021/06/how-many-cars-are-there-in-the-world/*

Ho, S. & Xie, M. (1998), 'The use of arima models for reliability forecasting and analysis', *Computers & industrial engineering* **35**(1-2), 213–216.

Hobeika, A. G. & Kim, C. K. (1994), Traffic-flow-prediction systems based on upstream traffic, *in* 'Proceedings of VNIS'94-1994 Vehicle Navigation and Information Systems Conference', IEEE, pp. 345–350.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural Computation* **9**(8), 1735–1780.

Hogg, R. V., McKean, J. & Craig, A. T. (2005), *Introduction to mathematical statistics*, Pearson Education.

Hong, W.-C., Dong, Y., Zheng, F. & Lai, C.-Y. (2011), 'Forecasting urban traffic flow by svr with continuous aco', *Applied Mathematical Modelling* **35**(3), 1282–1291.

Huang, W., Song, G., Hong, H. & Xie, K. (2014), 'Deep architecture for traffic flow prediction: deep belief networks with multitask learning', *IEEE Trans. Intell. Transp. Syst.* **15**(5), 2191–2201.

INRIX (n.d.*a*), '2019 global traffic scorecard'. last accessed: 08.12.2021.
URL: *https://inrix.com/scorecard/*

INRIX (n.d.*b*), '2020 global traffic scorecard'. last accessed: 08.12.2021.
URL: *https://inrix.com/scorecard/*

INRIX (n.d.*c*), 'Inrix: Congestion costs each american nearly 100 hours, $1,400 a year'. accessed: 12.20.2020.
URL: *https://inrix.com/press-releases/2019-traffic-scorecard-us/*

INRIX (n.d.*d*), 'Inrix global traffic scorecard: Congestion cost uk economy £6.9 billion in 2019'. last accessed: 12.20.2020.
URL: *https://inrix.com/press-releases/2019-traffic-scorecard-uk/*

Jafari, S., Shahbazi, Z. & Byun, Y.-C. (2021), 'Improving the performance of single-intersection urban traffic networks based on a model predictive controller', *Sustainability* **13**(10), 5630.

Jagadish, H., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R. & Shahabi, C. (2014), 'Big data and its technical challenges', *Communications of the ACM* **57**(7), 86–94.

Jensen, L. (1990), 'Guidelines for the application of arima models in time series', *Research in nursing & health* **13**(6), 429–435.

Ji, S., Xu, W., Yang, M. & Yu, K. (2012), '3d convolutional neural networks for human action recognition', *IEEE transactions on pattern analysis and machine intelligence* **35**(1), 221–231.

Jia, T. & Yan, P. (2020), 'Predicting citywide road traffic flow using deep spatiotemporal neural networks', *IEEE Transactions on Intelligent Transportation Systems* **22**(5), 3101–3111.

Jia, Y., Wu, J. & Du, Y. (2016), Traffic speed prediction using deep learning method, *in* '2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)', IEEE, pp. 1217–1222.

Jiang, W. & Zhang, L. (2018), 'Geospatial data to images: A deep-learning framework for traffic forecasting', *Tsinghua Science and Technology* **24**(1), 52–64.

Jin, W., Lin, Y., Wu, Z. & Wan, H. (2018), Spatio-temporal recurrent convolutional networks for citywide short-term crowd flows prediction, *in* 'Proceedings of the 2nd International Conference on Compute and Data Analysis', pp. 28–35.

Jordan, M. I. & Mitchell, T. M. (2015), 'Machine learning: Trends, perspectives, and prospects', *Science* **349**(6245), 255–260.

Kalchbrenner, N. & Blunsom, P. (2013), Recurrent continuous translation models, *in* 'Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing', pp. 1700–1709.

Kang, D., Lv, Y. & Chen, Y.-y. (2017), Short-term traffic flow prediction with lstm recurrent neural network, *in* '2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)', IEEE, pp. 1–6.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R. & Fei-Fei, L. (2014), Large-scale video classification with convolutional neural networks, *in* 'Proceedings of the IEEE conference on Computer Vision and Pattern Recognition', pp. 1725–1732.

Karthiga, R., Usha, G., Raju, N. & Narasimhan, K. (2021), Transfer learning based breast cancer classification using one-hot encoding technique, *in* '2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)', IEEE, pp. 115–120.

Khan, Z., Khan, S. M., Dey, K. & Chowdhury, M. (2019), 'Development and evaluation of recurrent neural network-based models for hourly traffic volume and annual average daily traffic prediction', *Transportation Research Record* **2673**(7), 489–503.

Kim, D. & Jeong, O. (2020), 'Cooperative traffic signal control with traffic flow prediction in multi-intersection', *Sensors* **20**(1), 137.

Kim, Y., Wang, P. & Mihaylova, L. (2019), 'Scalable learning with a structural recurrent neural network for short-term traffic prediction', *IEEE Sensors Journal* **19**(23), 11359–11366.

Kim, Y., Wang, P., Zhu, Y. & Mihaylova, L. (2018), A capsule network for traffic speed prediction in complex road networks, *in* '2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF)', IEEE, pp. 1–6.

Kingma, D. P. & Ba, J. (2014), Adam: A method for stochastic optimization, *in* 'Proceedings of the 3th International Conference on Learning Representations (ICLR)'.

Kipf, T. N. & Welling, M. (2017), Semi-supervised classification with graph convolutional networks, *in* 'Proceedings of the International Conference on Learning Representations (ICLR)'.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

Lawrence, S., Giles, C. L., Tsoi, A. C. & Back, A. D. (1997), 'Face recognition: A convolutional neural-network approach', *IEEE transactions on neural networks* **8**(1), 98–113.

Le Nguyen, P., Ji, Y. et al. (2019), Deep convolutional lstm network-based traffic matrix prediction with partial information, *in* '2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)', IEEE, pp. 261–269.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *nature* **521**(7553), 436–444.

LeCun, Y., Bengio, Y. et al. (1995), 'Convolutional networks for images, speech, and time series', *The handbook of brain theory and neural networks* **3361**(10), 1995.

Lee, S. & Fambro, D. B. (1999), 'Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting', *Transp. Res. Record* **1678**(1), 179–188.

Li, J., Fu, D., Yuan, Q., Zhang, H., Chen, K., Yang, S. & Yang, F. (2019), 'A traffic prediction enabled double rewarded value iteration network for route planning', *IEEE Transactions on Vehicular Technology* **68**(5), 4170–4181.

Li, L., He, S., Zhang, J. & Ran, B. (2016), 'Short-term highway traffic flow prediction based on a hybrid strategy considering temporal–spatial information', *Journ. Adv. Transp.* **50**(8), 2029–2040.

Li, Y., Chai, S., Ma, Z. & Wang, G. (2021), 'A hybrid deep learning framework for long-term traffic flow prediction', *IEEE Access* **9**, 11264–11271.

Li, Y., Yu, R., Shahabi, C. & Liu, Y. (2018), Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, *in* 'Proceedings of the International Conference on Learning Representations (ICLR)'.

Li, Y. & Yuan, Y. (2017), Convergence analysis of two-layer neural networks with relu activation, *in* 'Advances in neural information processing systems', pp. 597–607.

Li, Z., Xiong, G., Chen, Y., Lv, Y., Hu, B., Zhu, F. & Wang, F.-Y. (2019), A hybrid deep learning approach with gcn and lstm for traffic flow prediction, *in* '2019 IEEE Intelligent Transportation Systems Conference (ITSC)', IEEE, pp. 1929–1933.

Liao, B., Zhang, J., Wu, C., McIlwraith, D., Chen, T., Yang, S., Guo, Y. & Wu, F. (2018), Deep sequence learning with auxiliary information for traffic prediction, *in* 'Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining', pp. 537–546.

Lippi, M., Bertini, M. & Frasconi, P. (2013), 'Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning', *IEEE Transactions on Intelligent Transportation Systems* **14**(2), 871–882.

Liu, Y., Yang, C., Gao, Z. & Yao, Y. (2018), 'Ensemble deep kernel learning with application to quality prediction in industrial polymerization processes', *Chemometrics and Intelligent Laboratory Systems* **174**, 15–21.

Liu, Y., Zhang, Z. & Chen, J. (2015), 'Ensemble local kernel learning for online prediction of distributed product outputs in chemical processes', *Chemical Engineering Science* **137**, 140–151.

Liu, Y., Zheng, H., Feng, X. & Chen, Z. (2017), Short-term traffic flow prediction with conv-lstm, *in* '9th Int'l. Conf. Wireless Communications and Signal Processing (WCSP)', IEEE, pp. 1–6.

Long, J., Gao, Z., Ren, H. & Lian, A. (2008), 'Urban traffic congestion propagation and bottleneck identification', *Science in China Series F: Information Sciences* **51**(7), 948.

Lv, Y. et al. (2014), 'Traffic flow prediction with big data: a deep learning approach', *IEEE Trans. Intell. Transp. Syst.* **16**(2).

Lv, Y. et al. (2015), 'Traffic flow prediction with big data: A deep learning approach.', *IEEE Trans. Intell. Transp. Syst.* **16**(2).

Ma, T., Antoniou, C. & Toledo, T. (2020), 'Hybrid machine learning algorithm and statistical time series model for network-wide traffic forecast', *Transportation Research Part C: Emerging Technologies* **111**, 352–372.

Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y. & Wang, Y. (2017), 'Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction', *Sensors* **17**(4), 818.

Ma, X., Zhong, H., Li, Y., Ma, J., Cui, Z. & Wang, Y. (2020), 'Forecasting transportation network speed using deep capsule networks with nested lstm models', *IEEE Transactions on Intelligent Transportation Systems* **22**(8), 4813–4824.

Ma, X. et al. (2015), 'Long short-term memory neural network for traffic speed prediction using remote microwave sensor data', *Transp. Res. Part C Emerg. Technol.* **54**, 187–197.

Manual, H. C. (2000), 'Highway capacity manual', *Washington, DC* **2**(1).

Maravall, A., Del Rio, A. et al. (2001), *Time aggregation and the Hodrick-Prescott filter*, Banco de España.

Mathew, T. V. & Rao, K. (2006), 'Fundamental relations of traffic flow', *Introduction to Transportation Engineering* **1**, 1–8.

May, A. D. (1990), *Traffic flow fundamentals*, Englewood Cliffs, NJ: Pretice-Hall.

Mehdi, H., Pooranian, Z. & Vinueza Naranjo, P. G. (2019), 'Cloud traffic prediction based on fuzzy arima model with low dependence on historical data', *Transactions on Emerging Telecommunications Technologies* p. e3731.

Menard, S. (2002), *Applied logistic regression analysis*, Vol. 106, Sage.

Michal, P., Natasa, S.-D. & Matthias, G. (n.d.), 'The epfl/mobility dataset (v. 2009-02-24)'. last accessed: 21.06.2021.
URL: *https://crawdad.org/ crawdad/epfl/mobility/20090224/*

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J. & Khudanpur, S. (2010), Recurrent neural network based language model, *in* 'Eleventh annual conference of the international speech communication association', pp. 1045–1048.

Min, W. & Wynter, L. (2011), 'Real-time road traffic prediction with spatio-temporal correlations', *Transportation Research Part C: Emerging Technologies* **19**(4), 606–616.

Mondal, R., Mukherjee, D., Singh, P. K., Bhateja, V. & Sarkar, R. (2020), 'A new framework for smartphone sensor-based human activity recognition using graph neural network', *IEEE Sensors Journal* **21**(10), 11461–11468.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J. & Bronstein, M. M. (2017), Geometric deep learning on graphs and manifolds using mixture model cnns, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 5115–5124.

Nguyen, H., Liu, W. & Chen, F. (2016), 'Discovering congestion propagation patterns in spatio-temporal traffic data', *IEEE Transactions on Big Data* **3**(2), 169–180.

Niu, K., Zhang, H., Zhou, T., Cheng, C. & Wang, C. (2019), 'A novel spatio-temporal model for city-scale traffic speed prediction', *IEEE Access* **7**, 30050–30057.

Okawa, M., Kim, H. & Toda, H. (2017), Online traffic flow prediction using convolved bilinear poisson regression, *in* '2017 18th IEEE International Conference on Mobile Data Management (MDM)', IEEE, pp. 134–143.

Organisation, W. H. (n.d.), 'Road traffic injuries'. last accessed: 08.12.2021.
URL: *https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries*

Paisley, J., Blei, D. M. & Jordan, M. I. (2014), Bayesian nonnegative matrix factorization with stochastic variational inference, *in* 'Handbook of Mixed Membership Models and Their Applications', Chapman and Hall/CRC, pp. 239–258.

Pan, Z., Liang, Y., Wang, W., Yu, Y., Zheng, Y. & Zhang, J. (2019), Urban traffic prediction from spatio-temporal data using deep meta learning, *in* 'Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining', pp. 1720–1730.

Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W. & Lin, D. (2019), Libra r-cnn: Towards balanced learning for object detection, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 821–830.

Peng, H., Du, B., Liu, M., Liu, M., Ji, S., Wang, S., Zhang, X. & He, L. (2021), 'Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning', *Information Sciences* **578**, 401–416.

Ramakrishnan, N. & Soni, T. (2018), Network traffic prediction using recurrent neural networks, *in* '2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)', IEEE, pp. 187–193.

Rice, J. & Van Zwet, E. (2004), 'A simple and effective method for predicting travel times on freeways', *IEEE Transactions on Intelligent Transportation Systems* **5**(3), 200–207.

Sabour, S., Frosst, N. & Hinton, G. E. (2017), 'Dynamic routing between capsules', *Advances in neural information processing systems* **30**.

Sarker, A., Shen, H. & Stankovic, J. A. (2018), 'Morp: Data-driven multi-objective route planning and optimization for electric vehicles', *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* **1**(4), 1–35.

Seo, T., Bayen, A. M., Kusakabe, T. & Asakura, Y. (2017), 'Traffic state estimation on highway: A comprehensive survey', *Annual reviews in control* **43**, 128–151.

Seymour, M. (n.d.), 'Transport and environment statistics 2021 annual report'. last accessed: 05.06.2021.
URL: *https://www.gov.uk/government/statistics/transport-and-environment-statistics-2021*

Shalev-Shwartz, S. & Ben-David, S. (2014), *Understanding machine learning: From theory to algorithms*, Cambridge university press.

Shengdong, M., Zhengxian, X. & Yixiang, T. (2019), 'Intelligent traffic control system based on cloud computing and big data mining', *IEEE Transactions on Industrial Informatics* **15**(12), 6583–6592.

Shi, X., Qi, H., Shen, Y., Wu, G. & Yin, B. (2020), 'A spatial–temporal attention approach for traffic prediction', *IEEE Transactions on Intelligent Transportation Systems* **22**(8), 4909–4918.

Shi, Y., Feng, H., Geng, X., Tang, X. & Wang, Y. (2019), A survey of hybrid deep learning methods for traffic flow prediction, *in* 'Proceedings of the 2019 3rd International Conference on Advances in Image Processing', pp. 133–138.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A. & Vandergheynst, P. (2013), 'The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains', *IEEE signal processing magazine* **30**(3), 83–98.

Singh, D. & Mohan, C. K. (2018), 'Deep spatio-temporal representation for detection of road accidents using stacked autoencoder', *IEEE Transactions on Intelligent Transportation Systems* **20**(3), 879–887.

Smith, L. N. (2017), Cyclical learning rates for training neural networks, *in* 'IEEE Winter Conference on Applications of Computer Vision (WACV)', pp. 464–472.

Song, C., Lin, Y., Guo, S. & Wan, H. (2020), Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 34, pp. 914–921.

Sun, P., Aljeri, N. & Boukerche, A. (2020), 'Machine learning-based models for real-time traffic flow prediction in vehicular networks', *IEEE Network* **34**(3), 178–185.

Sun, T., Yang, C., Han, K., Ma, W. & Zhang, F. (2020), 'Bidirectional spatial–temporal network for traffic prediction with multisource data', *Transportation Research Record* **2674**(8), 78–89.

Sutskever, I., Vinyals, O. & Le, Q. V. (2014), Sequence to sequence learning with neural networks, *in* 'Advances in neural information processing systems', pp. 3104–3112.

Tan, M.-C. et al. (2009), 'An aggregation approach to short-term traffic flow prediction', *IEEE Trans. Intell. Transp. Syst.* **10**(1).

Tieleman, T. & Hinton, G. (2012), 'Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude', *COURSERA: Neural networks for machine learning* **4**(2), 26–31.

Toncharoen, R. & Piantanakulchai, M. (2018), Traffic state prediction using convolutional neural network, *in* '2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)', IEEE, pp. 1–6.

Treiber, M., Kesting, A. & Thiemann, C. (2008), How much does traffic congestion increase fuel consumption and emissions? applying a fuel consumption model to the ngsim trajectory data, *in* '87th Annual Meeting of the Transportation Research Board, Washington, DC', Vol. 71, pp. 1–18.

(UK), D. T. (2021), 'Number of cars on the road in the united kingdom (uk) 2000-2020', Website. last checked: 21.06.2022.
URL: *https://www.statista.com/statistics/299972/average-age-of-cars-on-the-road-in-the-united-kingdom/*

Vaa, T., Penttinen, M. & Spyropoulou, I. (2007), 'Intelligent transport systems and effects on road traffic accidents: state of the art', *IET Intelligent Transport Systems* **1**(2), 81–88.

Van Aerde, M. & Rakha, H. (1995), Multivariate calibration of single regime speed-flow-density relationships, *in* 'Proc. 6th Int'l. Conf. on Vehicle Navigation & Information Systems (VNIS)', IEEE, pp. 334–341.

Van Der Voort, M., Dougherty, M. & Watson, S. (1996), 'Combining kohonen maps with arima time series models to forecast traffic flow', *Transp. Res. Part C Emerg. Technol.* **4**(5), 307–318.

Vaswani, A., Shazeer, N., Parmar, N. & et.al. (2017), Attention is all you need, *in* 'Proceedings of the 31st International Conference on Neural Information Processing Systems', pp. 6000–6010.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. & Bengio, Y. (2018), Graph attention networks, *in* 'Proceedings of the International Conference on Learning Representations (ICLR)'.

Vinayakumar, R., Soman, K. & Poornachandran, P. (2017), Applying deep learning approaches for network traffic prediction, *in* '2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)', IEEE, pp. 2353–2358.

Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X. & Tang, X. (2017), Residual attention network for image classification, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3156–3164.

Wang, J., Zhu, W., Sun, Y. & Tian, C. (2021), 'An effective dynamic spatiotemporal framework with external features information for traffic prediction', *Applied Intelligence* **51**(6), 3159–3173.

Wang, Y., Assogba, K., Fan, J., Xu, M., Liu, Y. & Wang, H. (2019), 'Multi-depot green vehicle routing problem with shared transportation resource: Integration of time-dependent speed and piecewise penalty cost', *Journal of Cleaner Production* **232**, 12–29.

webmaster, T. M. (n.d.), 'Gateway traveler information system'. last accessed: 21.06.2021.
URL: *https://www.travelmidwest.com/lmiga/home.jsp*

Williams, B. M., Durvasula, P. K. & Brown, D. E. (1998), 'Urban freeway traffic flow prediction: application of seasonal autoregressive integrated moving average and exponential smoothing models', *Transportation Research Record* **1644**(1), 132–141.

Williams, B. M. & Hoel, L. A. (2003), 'Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results', *Journ. Transp. Eng.* **129**(6), 664–672.

Wu, Y. & Tan, H. (2016), 'Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework', *arXiv preprint arXiv:1612.01022* .

Wu, Y. et al. (2018), 'A hybrid deep learning based traffic flow prediction method and its understanding', *Transp. Res. Part C Emerg. Technol.* **90**, 166–180.

Xie, Z., Lv, W., Huang, S., Lu, Z., Du, B. & Huang, R. (2019), 'Sequential graph neural network for urban road traffic speed prediction', *IEEE Access* **8**, 63349–63358.

Xu, M., Dai, W. & et. al. (2020), 'Spatial-temporal transformer networks for traffic flow forecasting', *arXiv preprint arXiv:2001.02908* .

Yan, C., Tu, Y., Wang, X., Zhang, Y., Hao, X., Zhang, Y. & Dai, Q. (2019), 'Stat: spatial-temporal attention mechanism for video captioning', *IEEE transactions on multimedia* **22**(1), 229–241.

Yang, G., Wang, Y., Yu, H., Ren, Y. & Xie, J. (2018), 'Short-term traffic state prediction based on the spatiotemporal features of critical road sections', *Sensors* **18**(7), 2287.

Yao, H., Tang, X., Wei, H., Zheng, G. & Li, Z. (2019), Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 33, pp. 5668–5675.

Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., Gong, P., Ye, J. & Li, Z. (2018), Deep multi-view spatial-temporal network for taxi demand prediction, *in* 'Proceedings of the AAAI conference on artificial intelligence'.

Yin, X., Wu, G., Wei, J., Shen, Y., Qi, H. & Yin, B. (2021*a*), 'Deep learning on traffic prediction: Methods, analysis, and future directions', *IEEE Transactions on Intelligent Transportation Systems* **23**(6), 4927–4943.

Yin, X., Wu, G., Wei, J., Shen, Y., Qi, H. & Yin, B. (2021*b*), 'Multi-stage attention spatial-temporal graph networks for traffic prediction', *Neurocomputing* **428**, 42–53.

Yu, B., Yin, H. & Zhu, Z. (2018), Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting, *in* 'Proceedings of the 27th International Joint Conference on Artificial Intelligence', pp. 3634–3640.

Yu, H., Ji, N., Ren, Y. & Yang, C. (2019), 'A special event-based k-nearest neighbor model for short-term traffic state prediction', *Ieee Access* **7**, 81717–81729.

Yu, H., Wu, Z., Wang, S., Wang, Y. & Ma, X. (2017), 'Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks', *Sensors* **17**(7), 1501.

Yu, J., Wang, L. & Gong, X. (2013), 'Study on the status evaluation of urban road intersections traffic congestion base on ahp-topsis modal', *Procedia-Social and Behavioral Sciences* **96**, 609–616.

Zang, D., Ling, J., Wei, Z., Tang, K. & Cheng, J. (2018), 'Long-term traffic speed prediction based on multiscale spatio-temporal feature learning network', *IEEE Transactions on Intelligent Transportation Systems* **20**(10), 3700–3709.

Zhang, G. P. (2003), 'Time series forecasting using a hybrid arima and neural network model', *Neurocomputing* **50**, 159–175.

Zhang, J., Zheng, Y. & Qi, D. (2017), Deep spatio-temporal residual networks for citywide crowd flows prediction, *in* 'Thirty-first AAAI conference on artificial intelligence'.

Zhang, L., Liu, Q., Yang, W., Wei, N. & Dong, D. (2013), 'An improved k-nearest neighbor model for short-term traffic flow prediction', *Procedia-Social and Behavioral Sciences* **96**, 653–662.

Zhang, N., Guan, X., Cao, J., Wang, X. & Wu, H. (2019), 'Wavelet-hst: A wavelet-based higher-order spatio-temporal framework for urban traffic speed prediction', *IEEE Access* **7**, 118446–118458.

Zhang, S., Yao, Y., Hu, J., Zhao, Y., Li, S. & Hu, J. (2019), 'Deep autoencoder neural networks for short-term traffic congestion prediction of transportation networks', *Sensors* **19**(10), 2229.

Zhang, Z., Li, M., Lin, X., Wang, Y. & He, F. (2019), 'Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies', *Transp. Res. Part C Emerg. Technol.* **105**, 297–322.

Zhao, J., Qu, H., Zhao, J. & Jiang, D. (2019), 'Spatiotemporal traffic matrix prediction: A deep learning approach with wavelet multiscale analysis', *Transactions on Emerging Telecommunications Technologies* **30**(12), e3640.

Zhao, L., Peng, X., Tian, Y., Kapadia, M. & Metaxas, D. N. (2019), Semantic graph convolutional networks for 3d human pose regression, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 3425–3435.

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M. & Li, H. (2019), 'T-gcn: A temporal graph convolutional network for traffic prediction', *IEEE Transactions on Intelligent Transportation Systems* **21**(9), 3848–3858.

Zhao, N. et al. (2014), 'A practical method for estimating traffic flow characteristic parameters of tolled expressway using toll data', *Procedia-Social and Behavioral Sciences* **138**, 632–640.

Zhao, S., Lin, S. & Xu, J. (2019), Time series traffic prediction via hybrid neural networks, *in* '2019 IEEE Intelligent Transportation Systems Conference (ITSC)', IEEE, pp. 1671–1676.

Zheng, C., Fan, X., Wang, C. & et.al. (2020), Gman: A graph multi-attention network for traffic prediction, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', pp. 1234–1241.

Zheng, G., Chai, W. K. & Katos, V. (2022), 'A dynamic spatial–temporal deep learning framework for traffic speed prediction on large-scale road networks', *Expert Systems with Applications* **195**, 116585.

Zheng, G., Chai, W. K., Katos, V. & Walton, M. (2021), 'A joint temporal-spatial ensemble model for short-term traffic prediction', *Neurocomputing* **457**, 26–39.

Zheng, Z., Yang, Y., Liu, J., Dai, H.-N. & Zhang, Y. (2019), 'Deep and embedded learning approach for traffic flow prediction in urban informatics', *IEEE Transactions on Intelligent Transportation Systems* **20**(10), 3927–3939.

Zhu, J., Wang, Q., Tao, C., Deng, H., Zhao, L. & Li, H. (2021), 'Ast-gcn: Attribute-augmented spatiotemporal graph convolutional network for traffic forecasting', *IEEE Access* **9**, 35973–35983.

Zhu, L., Yu, F. R., Wang, Y., Ning, B. & Tang, T. (2018), 'Big data analytics in intelligent transportation systems: A survey', *IEEE Transactions on Intelligent Transportation Systems* **20**(1), 383–398.