Deep Learning for Short-term Network-wide Road Traffic Forecasting

Zhiyong Cui

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Yinhai Wang, Chair

Xuegang Ban

Ed McCormack

Program Authorized to Offer Degree:

Civil and Environmental Engineering

University of Washington

Abstract

Deep Learning for Short-term Network-wide Road Traffic Forecasting

Zhiyong Cui

Chair of the Supervisory Committee:
Yinhai Wang, Professor
Department of Civil and Environmental Engineering

Traffic forecasting is a critical component of modern intelligent transportation systems for urban traffic management and control. Learning and forecasting network-scale traffic states based on spatial-temporal traffic data is particularly challenging for classical statistical and machine learning models due to the time-varying traffic patterns and the complicated spatial dependencies on road networks. The existence of missing values in traffic data makes this task even harder. With the rise of deep learning, this work attempts to answer: how to design proper deep learning models to deal with complicated network-wide traffic data and extract comprehensive features to enhance prediction performance, and how to evaluate and apply existing deep learning-based traffic prediction models to further facilitate future research?

To address those key challenges in short-term road traffic forecasting problems, this work develops deep learning models and applications to: 1) extract comprehensive features from complex spatial-temporal data to enhance prediction performance, 2) address the missing value issue in traffic forecasting tasks, and 3) deal with multi-source data, evaluate existing deep learning-based traffic forecasting models, share model results as benchmarks, and apply those models into practice.

This work makes both original methodological and practical contributions to short-term network-wide traffic forecasting research. The traffic feature learning can categorized as learning traffic data as spatial-temporal matrices and learning the traffic network as a graph. Stacked bidirectional recurrent neural network is proposed to capture bidirectional temporal dependencies in traffic data. To learn localized features from the topological structure of the road network, two deep learning frameworks incorporating graph convolution and graph wavelet operations, respectively, are proposed to learn the interactions between roadway segments and predict their traffic states. To deal with missing values in traffic forecasting tasks, an imputation unit is incorporated into the recurrent neural network to increase prediction performance. Further, to fill in missing values in the graph-based traffic network, a graph Markov network is proposed, which can infer missing traffic states step by step along with the prediction process. In summary, the proposed graph-based models not only achieve superior forecasting performance but also increase the interpretability of the interaction between road segments during the forecasting process. From the practical perspective, to further facilitate future research, an open-source data and model sharing platform for evaluating existing traffic forecasting models as benchmarks is established. Additionally, a traffic performance measurement platform is presented which has the capability of taking the proposed network-wide traffic prediction models into practice.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Prof. Yinhai Wang, my PhD advisor and committee chair, for his support and guidance. I am so fortunate to have the opportunity to work with him and receive the encouragement and tremendous freedom from him to conduct research on my own pace to explore a broader range of topics and fields. Throughout my graduate research, he showed me how to be a constant source of new ideas and creative solutions. His ambition and persistence will also benefit my career and personal development.

I would also like to extend a great deal of appreciation to my doctoral supervisory committee members, Prof. Jeff Ban, Prof. Ed McCormack, and Prof. Qing Shen for their constructive advice and support through this process. I am grateful to Prof. Ying Long at Tsinghua University, Prof. Xiaolei Ma at Beihang University, and Prof. Lijun Sun at McGill University for their great support and guidance on my research. I want to thank Dr. Yan Gao at LinkedIn, Prof. Hsin-Mu Tsai at National Taiwan University, and Prof. Ge Gao at Peking University, who mentored me and whose encouragement led me to pursue a PhD.

I also greatly appreciate the support of my work and research from a variety of agencies and organizations over the past years, including but not limited to Washington State Department of Transportation (WSDOT) and Seattle Department of Transportation (WSDOT), the Pacific Northwest Transportation Consortium (PacTrans), and the Connected Cities with Smart Transportation (C2SMART). The Department of Civil and Environmental Engineering also provided me a great deal of assistance. I want to thank Nicholas Burmeister, a kind and reliable computer technician, for his great support to help prepare classes and maintain computer servers.

I feel this incredible journey cannot be successfully completed without the help of all the brilliant and supportive members of the Smart Transportation Applications and Research Laboratory (STAR Lab). I am fortunate to have worked and collaborated with two of my best friends, Dr. Ruimin Ke and Dr. Ziyuan Pu, on many research projects and studies. I will cherish our research discussion and our conversation about values. I also want to thank to Dr. Kristian Henrickson and Dr. John

Ash, two smart, diligent, and reliable man, for their help in terms of both research and daily life and our long-term friendship. I want to thank to Shuyi Yin, a brilliant and reliable guy, for his kind help during our studies, collaboration, and teaching. I would also like to thank all our former and current group members who had overlapped with me, particularly, Dr. Weibin Zhang, Dr. Zhibin Li, Dr. Ziqiang Zeng, Dr. Ying Jiang, Dr. Wei Sun, Dr. Wenbo Zhu, Mayuree Binjolkar, Yifan Zhuang, Frank Yang, Cole Kopca, Meixin Zhu, Chenxi Liu, Sam Ricord, Mingjiang Fu, and Sishuang Wang, who have collaborated with me on different projects or co-authored papers with me in the past few years.

My deep appreciation also goes to all my friends from universities, organizations, and companies, including my roommates, Guanjie Niu and Dr. Xiasen Wang, who always inspired me with their unlimited wisdom and passion. I will cherish the memory of the many weekends and holidays that we had spent and celebrated together. More importantly, I also want to thank to a lot of friends who may be far away but kindly tolerate and support me to pursue my dreams.

Finally, I dedicate this dissertation to my parents whose unconditional support made everything possible. Without their boundless love, I would not have gone so far.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

1.1 UNDERSTANDING MODERN URBAN TRAFFIC

The advancement of new smart traffic sensing and communication technologies has stimulated significant growth in the volume and variety of transportation data. A huge volume of newly generated data is playing an important role in modern urban transportation and smart city research and applications, including autonomous driving, real-time traffic update and prediction, network-level traffic control, connected vehicles, and smart infrastructure.

However, we are still facing challenges regarding how to fully and properly use these various massive transportation data sets. In recent years, new transportation sensing technologies have constantly emerged, which provide more options to transportation practitioners and researchers to collect necessary transportation data. Due to the diversity and variety of transportation data, both transportation practitioners and researchers are facing substantial opportunities and challenges.

Real transportation problems are normally intricate and related to many unexpected influential factors. The complexity of learning information from transportation data depends on size of the datasets and the scale of analyzed problem. The sizes of datasets used in many previous studies are too small to reflect the real-world complexity of these problems. Methodologies built based on complicated assumptions and small datasets are hard to be applied into the real transportation problems. To deal with real complicated data and understand the core of transportation problems, finding proper and practicable ways to process data and mining patterns from the datasets are important.

The spatial-temporal patterns are the fundamental properties of nearly all types of transportation data. Mining spatial-temporal traffic pattern is critical for modern intelligent transportation systems (ITS) and the planning for smart cities. For large-scale roadway networks in metropolitan cities, traffic patterns are hard to measure as a result of the huge amount of roadway links and the complex topological roadway structures. Since the traffic states vary along with the time, it is even more complicated to model the evolution of urban traffic patterns. Thus, a comprehensive recognition of urban traffic patterns is required for urban traffic analysis, such as identifying bottlenecks of traffic networks and enhancing the forecasting of future traffic states.

As a component of ITS, accurate network-wide traffic prediction is the prerequisite for many transportation research and applications, such as dynamic route planning and traffic assignment optimization. The growing need for short-term prediction of traffic parameters embedded in ITS has led to a great deal of research on traffic forecasting in the last three decades (Vlahogianni et al., 2004). Much previous research focusing on traffic prediction only studied a roadway segment or several consecutive segments on a corridor. It is proved that using the information of multiple sensors/locations can help prediction models track short-term trends and enhance prediction performance (Li et al., 2015). Thus, in order to enhance traffic prediction performance and bring the power of artificial intelligence into real applications in the transportation field, it is inevitable to take large-scale traffic networks as the study areas. Previous work (Park and Rilett, 1999; Vlahogianni et al., 2014) on the traffic prediction models roughly categorizes existing models into two categories: classical statistical methods and machine learning models. Most of the studies focusing on traffic forecasting using statistical methods were developed when traffic systems were less complex, and the sizes of traffic datasets were relatively small. However, statistical models' capability of handling high dimensional time series data is quite limited. With the more recent rapid development in computational power, as well as growth in traffic data volume, much of the more recent work on this topic focuses on machine learning methods for traffic forecasting.

1.2 CHALLENGES IN DEEP LEARNING-BASED TRAFFIC FORECASTING

There are several obvious hurdles in the traffic modeling and forecasting process. The first question is how to represent the complex structure of a roadway network accurately? Previous studies

attempted to convert the traffic states of sensing locations in a roadway network into a 2D spatial-temporal matrix (Ma et al., 2017) or convert the geometrical structure of urban roadway networks as colored images (Yu et al., 2017a) for learning traffic states' features. However, in these ways, the topology of a roadway network cannot be adequately represented and the relationship of the adjacent roadway segments can hardly be comprehensively learned. To address this issue, many studies (Cui et al., 2019; Yu et al., 2018) have proposed a more elegant way to consider the traffic network as a graph consisting of vertices and edges denoting roadway segments and intersections, respectively.

The second hurdle is designing an effective feature extraction process from the historical traffic time series data. Deep learning as a branch of machine learning has shown its superiority of capturing complex non-linear relationships. Recurrent neural network (RNN) as a type of deep learning models is suitable for dealing with time series data. RNN and its variants, including long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014) networks, have also shown great potential for solving traffic forecasting problems (Kong et al., 2019; Yu et al., 2017b). However, although RNN-based methods can learn the spatial dependencies, they tend to be over-complex and inevitably capture a certain amount of noise and spurious relationships which likely do not represent the true causal structure in a physical traffic network. Besides, although transportation is a proper scientific domain for applying novel deep learning models, most of the existing deep learning models are not originally designed for transportation problems. Hence, how to incorporating appropriate spatial-temporal dependencies in development of novel transportation problem oriented deep learning models is another challenge for transportation researchers and practitioners.

The third problem in the traffic modeling and forecasting process is related to the fundamental properties of deep learning based models. These models with powerful non-linear fitting capabilities can provide great power for dealing with complicated transportation problems that cannot be solved by classical methods. However, the deep learning-based models are normally highly flexible, and thus, the designs of these models need to be customized depending on specific problems and datasets. For a specific problem, the design of a model structure and the selection of hyper-parameters will directly affect the model performance. Thus, how to standardize the model design and selection process and how to effectively evaluate the performance of different traffic

modeling and forecasting models are important for achieving superior modeling performance and further facilitating related research.

This work is in part motivated by the needs to efficiently overcome these challenges and bring the power of artificial intelligence into the transportation field that will be expected to develop novel transportation problem oriented models to learn urban traffic patterns and compare their performance by testing on existing standard datasets to stimulate the emergence of new methodologies.

1.3 BACKGROUND

1.3.1 *Network-wide Traffic Data*

In recent years, with the developments in traffic sensing, data storage, and communication technologies, the availability and diversity of traffic data have increased substantially. Transportation data is normally collected by traffic sensors, such as inductive loop detectors, monitoring cameras, and Wi-Fi/Bluetooth sensors. By means of more novel intelligent systems, transportation data can also be collected via personal mobile computing devices and probe vehicles. Moreover, transportation related data should include weather data, traffic incident data, emission data, etc. Therefore, transportation data are very beneficial to public agencies and researchers for managing transportation systems and conducting comprehensive traffic analysis.

When the volume and variety of traffic data increase, fully making use of the existing multi-source data to provide accurate network-wide traffic information is becoming a challenging task (El Faouzi et al., 2011). Integrating and utilizing multi-source network-wide traffic data have several hurdles.

Dealing with various spatiotemporal resolutions is one of the most challenging tasks for integrating multi-source transportation data, due to datasets might be collected and formatted by various transportation related practitioners for different purposes. The temporal resolution (the minimum time interval units) of traffic monitoring data may range from seconds to hours. The spatial granularity of traffic data also varies significantly. Some types of traffic data are collected to monitor arterials and urban corridors in downtown areas, and some others may be only applicable

for freeways. Moreover, some datasets may overlap in terms of their spatial coverage, and some other datasets may have complementary monitoring areas.

The second challenge is that the geospatial representations of roadways in different datasets usually differ from each other because there is no uniform geometric referencing layer for those datasets. The OpenStreetMap (OSM) (Haklay and Weber, 2008) provides a comprehensive worldwide map and the detailed information of roadway networks. Although OSM roadway layers are editable and with adequate roadway metadata, OSM roadway layers are still different from the roadway layers of the National Performance Management Research Data Set (NPMRDS) (Kaushik et al., 2015), which is used for supporting national highway system performance measurement and management activities. At the current stage, there is no standard geometric representation or GIS map layer for universal transportation analysis. If multiple datasets with different geospatial representations are used, a process of map conflation is normally needed to combine geographic information from overlapping sources so as to retain accurate data, minimize redundancy, and reconcile data conflicts (Longley et al., 2001).

Further, modern transportation analysis and management are in great need of flexible and expandable transportation data integration frameworks. Data integration is the problem of combining data residing at different sources, and providing a unified view of these data to the users (Lenzerini, 2002). Designing data integration systems is an important procedure in a variety of real-world applications, especially in Intelligent Transportation Systems (ITS). Other conventional problems in transportation modeling are also concerned with multi-source processing, like planning problems, demand estimation, and traffic estimation (El Faouzi et al., 2011). Transportation data integration frameworks and tools have been designed and implemented for multiple scenarios in both research and practice.

1.3.2 *Network-wide Traffic Learning and Forecasting*

Classical traffic forecasting models can generally be classified into two categories, traditional statistical models and computational intelligence, i.e. machine learning-based, models. The statistical methods are mostly parametric approaches, including variants of auto-regressive integrated moving average (ARIMA) models, parametric Kalman filtering models, and other types of time-series

models, that are developed based on a predefined model structure with theoretical assumptions and the parameters are calibrated using historical data. With the ability to accommodate the stochastic and non-linear nature of traffic patterns, classical machine learning methods are widely adopted for the traffic forecasting task, such as support vector regression, Bayesian network approaches. In recent years, with the rise of AI, the performance of emerging deep learning-based traffic forecasting methods outperform that of classical methods.

Due to traffic prediction is based on historical sequence data, most of the existing deep learning models (Cui et al., 2017; Ma et al., 2015; Wang et al., 2019a) are built on RNN and its variants like LSTM and GRU. Because of the spatiotemporal characteristics of traffic data, a large amount of other types of deep learning models, including convolutional neural network (CNN) (Ma et al., 2017), stacked autoencoder (Lv et al., 2015), generative adversarial network (GAN) (Liang et al., 2018), and capsule network (Ma et al., 2020), and multiple combinations (Cui et al., 2019; Ke et al., 2020; Liao et al., 2018; Lv et al., 2018; Wu et al., 2018) of these models were utilized to capture spatial features and estimate traffic states. Since the traffic data contain both spatial and temporal attributes, the deep learning-based methods can be grouped by the ways to deal with spatial-temporal traffic data. One type of studies convert the spatial-temporal data into a 2-dimensional (2D) matrix and use neural networks to extract feature and forecast traffic states. However, a traffic network's spatial features cannot be completely represented by a 2D matrix. Thus, another type of methods convert the physical roadway networks as images according to roads' geospatial properties. Although the traffic network images demonstrate the true traffic network structure, those images contain too many noisy pixels and blank pixels without traffic state information. To analyze the network-wide traffic states in an efficient way, studies consider the traffic network as a graph and predict traffic state by incorporating the graph convolutional network. The majority of the proposed deep learning-based traffic forecasting methods in this dissertation fall into this type of work and learn the traffic network as a graph.

1.4 RESEARCH OBJECTIVES

This research is oriented towards addressing network-wide traffic modeling and forecasting problem. Comparing to existing traffic forecasting methods, the developed models are expected to

be novel, applicable, and reproducible, which can be considered as a fundamental contribution for modeling network-wide traffic network using deep learning methods. To achieve this goal, platforms for evaluating, sharing, and applying existing traffic prediction models will be developed to facilitate further studies in this research area.

The research objectives of this dissertation can be described from three perspectives:

1. Design advanced deep learning algorithms to learn comprehensive features from network-wide spatial-temporal traffic data to enhance short-term traffic prediction performance. Specifically, the spatial-temporal traffic data can be learned as 2-dimensional matrices and time-variant topological graphs.

2. Design customized neural network structures to deal with missing values in the traffic forecasting problem.

3. Integrate multi-source data and develop platforms to evaluate, share, and apply existing traffic forecasting models to facilitate future research.

1.5 DISSERTATION ORGANIZATION

This dissertation introduces the deep learning frameworks for large-scale short traffic prediction. We start with a review the historical and current status of spatial-temporal traffic data learning and prediction. We mainly introduce the challenges and opportunities in the traffic learning and prediction field. Aiming at facilitate future research, we also review the existing model sharing applications and platforms. The primary contributions of this dissertation and the organization of the remaining chapters are presented in Figure 1.1.

Part i: Prediction. The first part of the dissertation proposes three types of neural network structures based on the network-wide traffic state datasets and their various characteristics to achieve specific traffic prediction goals. Chapter 3 describes a stacked bidirectional recurrent neural network to take the bidirectional dependencies in both time series and road segments into consideration to achieve good prediction performance. To incorporate roadway physical properties and extract localized features from the road network, We learn the traffic network as graph. Chapter 4 presents the traffic graph convolutional recurrent neural network, which

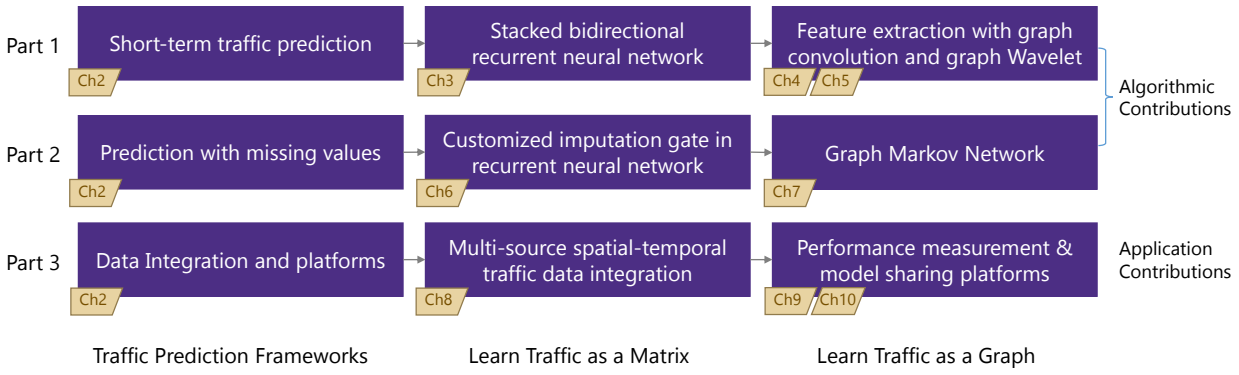


Figure 1.1: Dissertation organization and summary of contribution.

incorporates traffic speed and roadway length into the graph convolution operation. Chapter 5 moves one step further to apply graph wavelet operation into the recurrent neural network structure to dynamically capture the localized features in traffic network.

Part ii: Prediction with Missing Values. The second part mainly explores the challenges of missing values in traffic prediction tasks by designing neural network models. Chapter 6 adds an imputation unit in the LSTM structure to infer the missing values during the prediction process. Learning traffic network as graph, Chapter 7 presents a graph Markov process, which describes the network-wide traffic state transition process. Based on the graph Markov process, a graph Markov network is proposed with the capability of imputing missing values and predicting future traffic states at the same time.

Part iii: Integration and Platforms. The third part of the dissertation mainly describes how the proposed methods in the previous two parts can be shared and applied into practice to benefit future research. Chapter 8 describes a multi-source traffic data integration framework whose generated integrated data can be a source of traffic prediction tasks. Chapter 9 introduces an open source platform, TraffiX.ai, which shares standardized traffic prediction datasets and publishes the source code and evaluation results of existing deep learning traffic prediction models as benchmarks. Chapter 10 demonstrates a network-wide traffic data based performance measurement platform which analyzes traffic performance under special conditions and applies the proposed the proposed traffic prediction methods into real applications.

Part iv: Discussion and Concluding Remarks. This dissertation attempts to bridge the gap between classical transportation data modeling and the developing trend in the traffic pattern prediction and estimation field. We not only propose deep learning-based models to deal with specific traffic prediction tasks, but also apply those models into practice by establishing data sharing and model evaluation platforms. We therefore close with a summary of the research effort and a discussion of the road ahead. We also discuss the remaining gaps in the traffic prediction field and envision a broader spectrum of related research directions.

2

LITERATURE REVIEW

Artificial intelligence (AI) has the potential to solve problems that are hard for traditional methods to address, and various AI methods have achieved state-of-the-art performances in speech recognition, visual object recognition, object detection and many other domains (LeCun et al., 2015). Some AI-based methods even surpass human-level performance on some specific problems (Mnih et al., 2015). With increasing population, vehicles, and mobility demands, improving the safety, efficiency, and sustainability of the transportation system remains a challenge, and traditional methods may not be able to fully address it. To overcome these issues, an increasing amount of studies have been conducted to apply AI-based methods, especially deep learning methods, to solve complicated transportation problems including traffic signal control (Abdulhai et al., 2003; Arel et al., 2010), traffic prediction (Li et al., 2018; Lv et al., 2015), and microscopic traffic modeling (Wang et al., 2017; Zhou et al., 2017).

As a component of ITS, accurate network-wide traffic prediction is the prerequisite for dynamic route planning and traffic assignment optimization. The growing need for short-term prediction of traffic parameters embedded in ITS has led to a great deal of research on traffic forecasting in the last three decades (Vlahogianni et al., 2004). Before the rise of artificial intelligence, traffic forecasting methods have been gradually shifting from traditional statistical models to computational intelligence, or say machine learning methods . Previous work (Ma et al., 2017, 2015; Park and Rilett, 1999) on this topic roughly categorizes existing models into two categories: classical statistical methods and machine learning models. Most of the studies focusing on traffic forecasting using statistical methods were developed when traffic systems were less complex, and the sizes of traffic datasets were relatively small. However, statistical models' capability of handling

high dimensional time series data is quite limited. With the more recent rapid development in computational power, as well as growth in traffic data volume, much of the more recent work on this topic focuses on machine learning methods for traffic forecasting. In most recent years, deep learning as a branch of machine learning was widely used in the traffic forecasting research area.

2.1 CLASSICAL MODELS BASED TRAFFIC PREDICTION

Previous traffic prediction methods can be categorized into two main groups, i.e. parametric approaches and nonparametric approaches (Lv et al., 2015). Parametric traffic prediction approaches are developed based on a predefined model structure with several certain theoretical assumptions, and parameters are calibrated using historical data (Smith et al., 2002). A variety of parametric traffic prediction approaches were proposed, including many variants of autoregressive integrated moving average (ARIMA) models (Williams, 2001), parametric Kalman filtering models (Okutani and Stephanedes, 1984), and other types of time-series models (Ghosh et al., 2009). In order to accommodate the stochastic and nonlinear nature of traffic flow, nonparametric approaches were also widely adopted for traffic prediction or traffic data imputation, including k-nearest neighbor (k-NN) methods (Chang et al., 2012), support vector regression (SVR) (Wu et al., 2004), Bayesian network approaches (Sun et al., 2006), and tensor decomposition approach (Chen et al., 2019b). In spite of classical traffic prediction models are well studied and applied, it is pretty difficult for these models to deal with huge amount of large-scale network-wide traffic data.

2.2 DEEP LEARNING BASED TRAFFIC PREDICTION

Deep learning models have shown their superior capabilities of capturing nonlinear spatiotemporal effects for traffic forecasting (Polson and Sokolov, 2017). Ever since the precursory study (Lek et al., 2000) using the feed-forward NN for vehicle travel time estimation was proposed, many other NN-based models, including fuzzy NN (Yin et al., 2002), recurrent NN (Van Lint et al., 2002), convolution NN (Ma et al., 2017; Yu et al., 2017a), deep belief networks (Huang et al., 2014; Kong et al., 2019), auto-encoders (Liao et al., 2018; Lv et al., 2015), generative adversarial networks

(Liang et al., 2018; Lin et al., 2019), and combinations of these models have been applied to forecast traffic states. With the capability of capturing temporal dependencies, the recurrent NN or its variants, like LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014), was widely adopted as a component of a traffic forecasting model to forecast traffic speed (Ma et al., 2015), travel time (Duan et al., 2016b), and traffic flow (Zhao et al., 2017).

Further, in most recent years, various novel deep learning-based traffic forecasting models have been proposed through adjusting classical neural network models, combining existing methods, and incorporating auxiliary data. Multiple novel LSTM based models, such as bidirectional LSTM (Cui et al., 2017), deep LSTM (Yu et al., 2017b), shared hidden LSTM (Song et al., 2016), and nested LSTM (Ma et al., 2020), have been designed via reorganizing and combing single LSTM models and applied to capture comprehensive temporal dependencies for traffic prediction. In addition, sequence-to-sequence (seq2seq) architecture based models (Liao et al., 2018) have also been used for traffic state sequence forecasting. To deal with different types of features, multi-stream deep learning models (Ke et al., 2020; Wu et al., 2018; Yu et al., 2017b; Zhang and Kabuka, 2018) have also been well studied and tested for traffic forecasting problems. To improve the prediction performance, multiple deep learning based models also incorporate various traffic-related auxiliary data, including roadway geographical attribute data (Liao et al., 2018), accident data (Yu et al., 2017b), and weather data (Zhang and Kabuka, 2018).

To capture spatial relationships present in traffic networks, many forecasting models (Jin et al., 2018; Ma et al., 2017) incorporating CNNs to extract spatial features from 2D spatial-temporal traffic data. Due to the traffic structure is hard to be depicted by 2D spatial-temporal data, studies (Yu et al., 2017b) tried to convert traffic network structures to images and use CNNs to learn spatial features. However, these converted images have a certain amount of noise, inevitably resulting in spurious spatial relationships captured by CNNs. Recent studies (Chen et al., 2018b; Guo et al., 2019; Zhang and Patras, 2018) also attempted to convert traffic state data into three-dimensional (3D) matrices and use the 3D convolutional network to extract more effective features. However, conventional CNN based methods still cannot inherently deal with the topological structure and the physical attributes of the traffic network. To solve this problem, studies (Li et al., 2018; Yu et al., 2018) attempted to learn the traffic network as a graph and adopt the graph-based convolution operator to extract features from the graph-structured traffic network.

Traffic networks have already been analyzed as graphs for dynamic shortest path routing (Sun et al., 2017), traffic congestion analysis (Sun et al., 2014), and dynamic traffic assignment (Kalafatas, 2007). In the last couple of years, many studies attempt to generalize neural networks to work on arbitrarily structured graphs by designing graph convolutional networks. Generally, the graph convolutional networks utilize the adjacency matrix or the Laplacian matrix to depict the structure of a graph. The Laplacian matrix based graph convolution (Bruna et al., 2013), (Henaff et al., 2015) are designed based on the spectral graph theory (Shuman et al., 2013). As an extension, a localized spectral graph convolution (Defferrard et al., 2016) is also proposed to reduce the learning complexity. The adjacency matrix based graph convolution neural networks (Kipf and Welling, 2019; Zhou et al., 2017) incorporate the adjacency matrix and their network structures are more flexible. The traffic network can be considered as a graph consisting of nodes and edges, and thus, several graph convolution neural network based models, including the spectral graph convolution (Yu et al., 2018) and the diffusion graph convolution (Atwood and Towsley, 2016), are proposed to fulfill network-wide traffic forecasting. Several studies (Geng et al., 2019; Zhang et al., 2018) also incorporated multi-scale graph convolution operations into their proposed models to learn traffic features. Although these existing methods can extract spatial features from neighborhoods in the traffic network, the physical specialties of roadways, like length, speed limits, and the number of lanes, are normally neglected. A recent work (Xu et al., 2019) proposed the graph wavelet neural network to implement efficient convolution on graph data to solve semi-supervised classification problems.

2.3 TRAFFIC FORECASTING WITH MISSING DATA

Traffic forecasting performance will be influenced by the missing values (Cui et al., 2017). A bunch of data imputation methods has been developed to solve the missing values issues, including the probabilistic principal component analysis (PCA) (Li et al., 2014), tensor decomposition-based methods (Chen et al., 2019b; Ran et al., 2016; Tan et al., 2016), clustering approaches (Ku et al., 2016; Tang et al., 2015). There are also some deep learning-based data imputation methods proposed in the most recent years, such as denoising stacked auto-encoder (Duan et al., 2016a) and generative adversarial imputation network (Yoon et al., 2018). However, the PCA-based and tensor

decomposition-based models normally need hundreds or thousands of iterations to converge and achieve their best data imputation performance. Further, the aforementioned deep learning models for data imputation are not originally designed for solving the traffic forecasting with missing values issue.

To combine the data imputation and traffic forecasting together, a few RNN-based approaches, such as the LSTM-M (Tian et al., 2018), have been proposed based the GRU-D (Che et al., 2018) for processing multivariate time series with missing values. Even though these RNN-based methods can recurrently fill missing values in each time step and forecasting the future traffic state, they cannot capture spatial interactions between road links in the traffic network. Further, although recent research (Barredo-Arrieta et al., 2019; Du et al., 2019) is trying to interpret RNN-based models, for the traffic forecasting problem, it is still hard for RNN-based models to interpret the spatial relationship between neighboring links and the links' temporal dependencies between different time steps.

2.4 TRANSPORTATION ARTIFICIAL INTELLIGENCE PLATFORM

Well-designed platforms or systems are capable of properly utilizing the existing immense transportation data sets and AI methods. For traffic signal control the DeepDrive platform [1] is developed to provide adaptive traffic signal control based on deep reinforcement learning. For traffic congestion detection and traffic prediction, PTV Optima [2] can generate traffic prediction information for up to 60 minutes in the future. Traffic congestion can be detected by the speed and traffic flow detected in the field or calculated from roadway traffic states data (e.g., floating car and license plate identification data). The Miovision TrafficLink platform [3] is developed to assist traffic engineers to create more responsive and efficient traffic networks. TIMON (Osaba et al., 2016) is a European research project, whose main objective is to provide real-time services through a web-based platform and a mobile application for drivers. Microsoft research team also pioneered the use of machine learning methods to build predictive models for traffic (Amershi

1 https://deepdrive.berkeley.edu/
2 https://www.ptvgroup.com/en-us/solutions/products/ptv-optima/
3 https://miovision.com/trafficlink/

et al., 2019). The developed models can infer and predict traffic flow at different time periods in the future based on the analysis of large amounts of data over months and years.

Part I

PREDICTION

3

STACKED DIRECTIONAL RECURRENT NEURAL NETWORK FOR TRAFFIC PREDICTION

3.1 OVERVIEW

The performances of intelligent transportation systems (ITS) applications largely rely on the quality of traffic information. Recently, with the significant increases in both the total traffic volume and the data they generate, opportunities and challenges exist in transportation management and research in terms of how to efficiently and accurately understand and exploit the essential information underneath these massive datasets. Short-term traffic forecasting based on data driven models for ITS applications has been one of the biggest developing research areas in utilizing massive traffic data, and has great influence on the overall performance of a variety of modern transportation systems. In the last three decades, a large number of methods have been proposed for traffic forecasting in terms of predicting speed, volume, density and travel time. Studies in this area normally focus on the methodology components, aiming at developing different models to improve prediction accuracy, efficiency, or robustness. Previous literature indicates that the existing models can be roughly divided into two categories, i.e. classical statistical methods and computational intelligence (CI) approaches (Ma et al., 2015). Most statistical methods for traffic forecasting were proposed at an earlier stage when traffic condition were less complex and transportation datasets were relatively small in size. Later on, with the rapid development in traffic sensing technologies and computational power, as well as traffic data volume, the majority of more recent work focuses on CI approaches for traffic forecasting.

3.1.1 *Background*

With the ability to deal with high dimensional data and the capability of capturing non-linear relationship, CI approaches tend to outperform the statistical methods, such as auto-regressive integrated moving average (ARIMA) (Ye et al., 2012), with respect to handling complex traffic forecasting problems (Karlaftis and Vlahogianni, 2011). However, the full potential of artificial intelligence was not exploited until the rise of neural networks (NN) based methods. Ever since the precursory study of utilizing NN into the traffic prediction problem was proposed (Hua and Faghri, 1994), many NN-based methods, like feed forward NN (Park and Rilett, 1999), fuzzy NN (Yin et al., 2002), recurrent NN (RNN) (Van Lint et al., 2002), and hybrid NN (Yu et al., 2017b), are adopted for traffic forecasting problems. Recurrent Neural Networks (RNNs) model sequence data by maintaining a chain-like structure and internal memory with loops (Jozefowicz et al., 2015) and, due to the dynamic nature of transportation, are especially suitable to capture the temporal evolution of traffic status. However, the chain-like structure and the depth of the loops make RNNs difficult to train because of the vanishing or blowing up gradient problems during the back-propagating process. There have been a number of attempts to overcome the difficulty of training RNNs over the years. These difficulties were successfully addressed by the Long Short-Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997), which is a type of RNN with gated structure to learn long-term dependencies of sequence-based tasks.

As a representative deep learning method handling sequence-data, LSTMs have been proved to be able to process sequence data (Jozefowicz et al., 2015) and applied in many real-world problems, like speech recognition (Graves et al., 2013), image captioning (Vinyals et al., 2015), music composition (Eck and Schmidhuber, 2002) and human trajectory prediction (Alahi et al., 2016). In recent years, LSTMs have been gaining popularity in traffic forecasting due to their ability to model long-term dependencies. Several studies (Chen et al., 2016; Ma et al., 2015; Song et al., 2016; Wu and Tan, 2016; Yu et al., 2017b) have been done to examine the applicability of LSTMs in traffic forecasting, and the results demonstrate the advantages of LSTMs. However, the potential of LSTMs is far from being fully exploited in the domain of transportation. The three primary limitations in previous work on LSTMs in traffic forecasting can be summarized as follows: 1) traffic forecasting has generally focused on a small collection of network level. 2) Most

of the structures of LSTM-based methods are shallow. 3) The long-term dependencies are normally learned from chronologically arranged input data considering only forward dependencies, while backward dependencies learned from reverse-chronological ordered data has never been explored. From the perspective of the scale of prediction area, predicting large-scale transportation network traffic has become an important and challenging topic. Most existing studies utilize traffic data at a sensor location or along a corridor, and thus, network-wide prediction could not be achieved unless N models were trained for a traffic network with N nodes (Duan et al., 2016b). While, learning complex spatial-temporal features of a large-scale traffic network by only one model should be explored.

Regarding depth of the structure of LSTM-based models, the structure should have the ability to capture the dynamic nature of the traffic system. Most of the newly proposed LSTM-based prediction models have relatively shallow structures with only one hidden layer to deal with time series data (Duan et al., 2016b; Fu et al., 2016; Ma et al., 2015). Existing studies (Fu et al., 2016; LeCun et al., 2015) have shown that deep LSTM architectures with several hidden layers can build up progressively higher levels of representations of sequence data. Although some studies (Chen et al., 2016; Wu and Tan, 2016; Yu et al., 2017b) utilized more than one hidden LSTM layer, the influences of the number of LSTM layers in different LSTM-based models need to be further compared and explained.

In terms of the dependency in prediction problems, all of the information contained in time series data should be fully utilized. Normally, the dataset fed to an LSTM model is chronologically arranged, with the result that the information in the LSTMs is passed in a positive direction from the time step t-1 to the time step t along the chain-like structure. Thus, the LSTM structure only makes use of the forward dependencies (Lipton et al., 2016). But in this process, it is highly possible that useful information is filtered out or not efficiently passed through the chain-like gated structure. Therefore, it may be informative to consider backward dependencies, which pass information in a negative direction, into consideration. Another reason for including backward dependency into our study is the periodicity of traffic. Unlike wind speed forecasting (Wang et al., 2012), traffic incident forecasting (Zheng and Liu, 2009), or many other time series forecasting problems with strong randomness, traffic conditions have strong periodicity and regularity, and even short-term periodicity can be observed (Jiang and Adeli, 2004). Analysing the periodicity

of time series data, especially for recurring traffic patterns, from both forward and backward temporal perspectives will enhance the predictive performance (Box et al., 2015). However, based on our review of the literature, few studies on traffic analysis utilized the backward dependency. To fill this gap, a bidirectional LSTM (BDLSTM) with the ability to deal with both forward and backward dependencies is adopted as a component of the network structure in this study.

In addition, when predicting the network-wide traffic speed, rather than the speed at a single location, the impact of upstream and downstream speeds on each location in the traffic network should not be neglected. Previous studies (Chandra and Al-Deek, 2009; Kamarianakis et al., 2010) which only making use of the forward dependencies of time series data have found that the past speed values of upstream as well as downstream locations influence the future speed values of a location along a corridor. However, for complicated traffic networks with intersections and loops, upstream and downstream both refer to relative positions, and two arbitrary locations can be upstream and downstream of each other. Upstream and downstream are defined with respect to space, while forward and backward dependencies are defined with respect to time. With the help of forward and backward dependencies of spatial-temporal data, the learned feature will be more comprehensive.

3.1.2 *Contribution and Organization of the Chapter*

In this chapter, we propose a stacked bidirectional and unidirectional LSTM (SBU-LSTM) neural network, combining LSTM and BDLSTM, for network-wide traffic speed prediction. The proposed model is capable of handling input data with missing values and is tested on both large-scale freeway and urban traffic networks in the Seattle area. Experimental results show that our model achieves network-wide traffic speed prediction with a high prediction accuracy. The influence of the number of layers, the number of time lags (the length of time series input), the dimension of weight matrices in LSTM/BDLSTM layers, and the impact of additional volume and occupancy data are further analysed. The model's scalability and its potential applications are also discussed. In summary, our contributions can be stated as follows:

1. The traffic forecasting area is expanded from a specific location or several adjacent locations along a corridor to large-scale traffic networks, varying from freeway traffic network to complex urban traffic network;

2. A deep architecture considering backward dependencies by combining LSTM and BDLSTM is proposed to enhance the feature learning from the large-scale spatial time series data;

3. we evaluate many of the model's internal and external influential factors.

The rest of this chapter is organized as follows. In Section 3.2, we introduce preliminaries of traffic prediction problems and several representative recurrent neural network-based methods. In Section 3.3, we introduce the proposed architecture, which combines LSTM and BDLSTM. In Section 3.4 and Section 3.5, experimental settings and results are presented, respectively. Finally, Section 3.6 provides some concluding remarks.

3.2 PRELIMINARIES

3.2.1 *Network-wide Traffic Speed Data*

Traffic speed prediction at one location normally uses a sequence of speed values with n historical time steps as the input data, which can be represented by a vector,

$$X_T = [x_{T-n}, x_{T-(n-1)}, \ldots, x_{T-2}, x_{T-1}] \tag{3.1}$$

But the traffic speed at one location may be influenced by the speeds of nearby locations or even locations faraway, especially when traffic jam propagates through the traffic network. To take these network-wide influences into account, the proposed and compared models in this study take the network-wide traffic speed data as the input. Suppose the traffic network consists of P

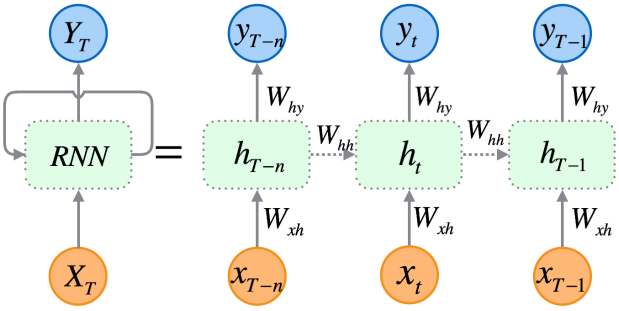


Figure 3.1: Standard RNN architecture and an unfolded structure with T time steps

locations and we need to predict the traffic speeds at time T using n historical time frames (steps), the input can be characterized as a speed data matrix,

$$X_T^P = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^P \end{bmatrix} = \begin{bmatrix} x_{T-n}^1 & x_{T-n+1}^1 & \cdots & x_{T-2}^1 & x_{T-1}^1 \\ x_{T-n}^2 & x_{T-n+1}^2 & \ddots & x_{T-2}^2 & x_{T-1}^2 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{T-n}^P & x_{T-n+1}^P & \cdots & x_{T-2}^P & x_{T-1}^P \end{bmatrix} \tag{3.2}$$

where each element x_t^p represents the speed of the t-th time frame at the p-th location. To reflect the temporal attributes of the speed data and simplify the expressions of the equations in the following subsections, the speed matrix is represented by a vector, $\boldsymbol{X}_T^P = \left[x_{T-n}, x_{T-(n-1)}, \ldots, x_{T-2}, x_{T-1}\right]$, in which each element is a vector of the P locations' speed values.

3.2.2 *Recurrent Neural Network*

RNN is a class of powerful deep neural network using its internal memory with loops to deal with sequence data. The architecture of RNNs, which also is the basic structure of LSTMs, is illustrated in Figure 3.1. For a hidden layer in RNN, it receives the input vector, $\boldsymbol{X}_T^P$, and generates the output vector, $\boldsymbol{Y}_T$. The unfolded structure of RNNs, shown in the right part of Figure 3.1, presents the calculation process that, at each time iteration, t, the hidden layer maintains a hidden state, h_t,

and updates it based on the layer input, x_t, and previous hidden state, h_{t-1}, using the following equation:

$$h_t = \sigma_h(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{3.3}$$

where W_{xh} is the weight matrix from the input layer to the hidden layer, W_{hh} is the weight matrix between two consecutive hidden states (h_{t-1} and h_t), b_h is the bias vector of the hidden layer and σ_h is the activation function to generate the hidden state. The network output can be characterized as:

$$y_t = \sigma_y(W_{hy}h_t + b_y) \tag{3.4}$$

where W_{hy} is the weight matrix from the hidden layer to the output layer, b_y is the bias vector of the output layer and σ_y is the activation function of the output layer. By applying the Equation 3.1 and Equation 3.2, the parameters of the RNN is trained and updated iteratively via the back-propagation (BP) method. In each time step t, the hidden layer will generate a value, y_t, and the last output, y_T, is the desired predicted speed in the next time step, namely $\hat{x}_{T+1} = y_T$.

Although RNNs exhibit the superior capability of modeling nonlinear time series problems (Ma et al., 2015), regular RNNs suffering from the vanishing or blowing up gradient during the BP process, and thus, being incapable of learning from long time lags (Gers et al., 1999), or saying long-term dependencies (Bengio et al., 1994).

3.2.3 *Long Short-Term Memory Network (LSTM)*

To handle the aforementioned problems of RNNs, several sophisticated recurrent architectures, like LSTM architecture (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) architecture (Cho et al., 2014) are proposed. It has been showed that the LSTMs work well on sequence-based tasks with long-term dependencies, but GRU, a simplified LSTM architecture, is only recently introduced and used in the context of machine translation (Chung et al., 2014). Although there are a variety of typical LSTM variants proposed in recent year, a large-scale analysis of LSTM variant shows that none of the variants can improve upon the standard LSTM architecture significantly (Greff et al., 2017). Thus, the standard LSTM architecture is adopted in this study as a part of the proposed network structure and introduced in this section.

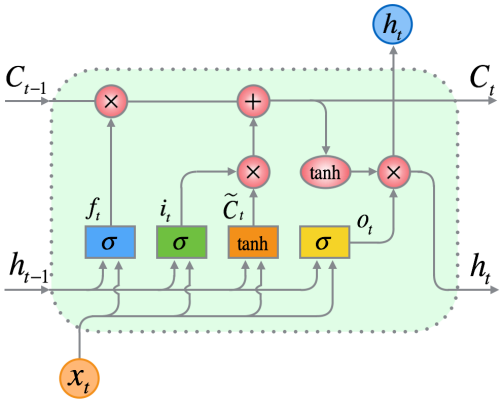


Figure 3.2: LSTM architecture. The pink circles are arithmetic operators and the colored rectangles are the gates in LSTM.

The only different component between standard LSTM architecture and RNN architecture is the hidden layer (Gers et al., 1999). The hidden layer of LSTM is also named as LSTM cell, which is shown in Figure 3.2. Like RNNs, at each time iteration, t, the LSTM cell has the layer input, x_t, and the layer output, h_t. The complicated cell also takes the cell input state, $\widetilde{C}_t$, the cell output state, C_t, and the previous cell output state, C_{t-1}, into account while training and updating parameters. Due to the gated structure, LSTM can deal with long-term dependencies to allow useful information pass along the LSTM network. There are three gates in a LSTM cell, including an input gate, a forget gate, and an output gate. The gated structure, especially the forget gate, helps LSTM to be an effective and scalable model for several learning problems related to sequential data (Greff et al., 2017). At time t, the input gate, the forget gate, and the output gate, denoted as i_t, f_t, and o_t respectively. The input gate, the forget gate, the output gate and the input cell state, which are represented by colorful boxes in the LSTM cell in Figure 3.2, can be calculated using the following equations:

$$f_t = \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{3.5}$$

$$i_t = \sigma_g\left(W_i x_t + U_i h_{t-1} + b_i\right) \tag{3.6}$$

$$o_t = \sigma_g\left(W_o x_t + U_o h_{t-1} + b_o\right) \tag{3.7}$$

$$\widetilde{C}_t = \tanh(W_C x_t + U_C h_{t-1} + b_C) \tag{3.8}$$

where W_f, W_i, W_o, and W_C are the weight matrices mapping the hidden layer input to the three gates and the input cell state, while the U_f, U_i, U_o, and U_C are the weight matrices connecting the previous cell output state to the three gates and the input cell state. The b_f, b_i, b_o, and b_C are four bias vectors. The σ_g is the gate activation function, which normally is the Sigmoid function, and the tanh is the hyperbolic tangent function. Based on the results of four above equations, at each time iteration t, the cell output state, C_t, and the layer output, h_t, can be calculated as follows:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.9}$$

$$h_t = o_t * \tanh(C_t) \tag{3.10}$$

The final output of a LSTM layer should be a vector of all the outputs, represented by $\mathbf{Y}_T = [h_{T-n}, \ldots, h_{T-1}]$. Here, when taking the speed prediction problem as an example, only the last element of the output vector, h_{T-1}, is what we want to predict. Thus, the predicted speed value ($\hat{x}$) for the next time iteration, T, is h_{T-1}, namely $\hat{x}_T = h_{T-1}$.

3.2.4 *Bidirectional LSTM*

The idea of BDLSTMs comes from bidirectional RNN (Schuster and Paliwal, 1997), which processes sequence data in both forward and backward directions with two separate hidden layers. BDLSTMs connect the two hidden layers to the same output layer. It has been proved that the bidirectional networks are substantially better than unidirectional ones in many fields. But bidirectional LSTMs have not been used in traffic prediction problem.

In this section, the structure of an unfolded BDLSTM layer, containing a forward LSTM layer and a backward LSTM layer, is introduced and illustrated in Figure 3.3. The forward layer output sequence, $\overrightarrow{\boldsymbol{h}}$, is iteratively calculated using inputs in a positive sequence from time $T-n$ to time $T-1$, while the backward layer output sequence, $\overleftarrow{\boldsymbol{h}}$, is calculated using the reversed inputs from

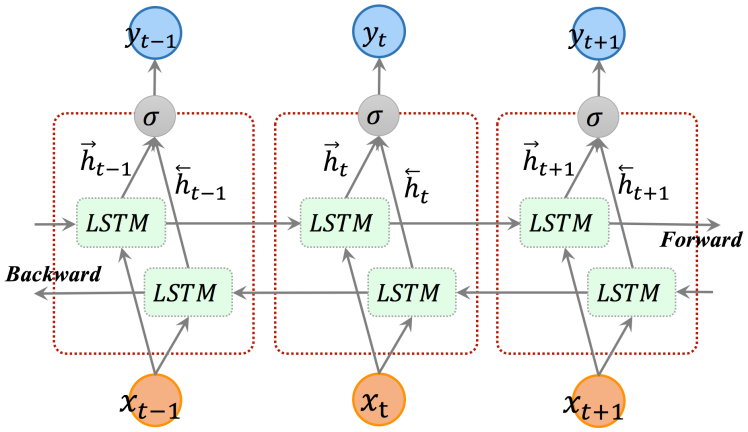


Figure 3.3: Unfolded architecture of bidirectional LSTM with three consecutive steps

time $T-n$ to $T-1$. Both the forward and backward layer outputs are calculated by using the standard LSTM updating equations, Equations 3.3 - 3.8. The BDLSTM layer generates an output vector, $\mathbf{Y}_T$, in which each element is calculated by using the following equation:

$$y_t = \sigma(\overrightarrow{h}_t, \overleftarrow{h}_t) \tag{3.11}$$

where σ function is used to combine the two output sequences. It can be a concatenating function, a summation function, an average function or a multiplication function. Similar to the LSTM layer, the final output of a BDLSTM layer can be represented by a vector, $\mathbf{Y}_T = [y_{T-n}, \ldots, y_{T-1}]$, in which the last element, y_{T-1}, is the predicted speed for the next time iteration when taking speed prediction as an example.

3.3 STACKED BIDIRECTIONAL AND UNIDIRECTIONAL LSTM

Existing studies (Graves et al., 2013; LeCun et al., 2015) have shown that deep LSTM architectures with several hidden layers can build up progressively higher level of representations of sequence data, and thus, work more effective. The deep LSTM architectures are networks with several stacked LSTM hidden layers, in which the output of a LSTM hidden layer will be fed as the input into the subsequent LSTM hidden layer. This stacked-layers mechanism, which can enhance the power of neural networks, is adopted in this study. As mentioned in previous sections, BDLSTMs can make use of both forward and backward dependencies. When feeding the spatial-temporal

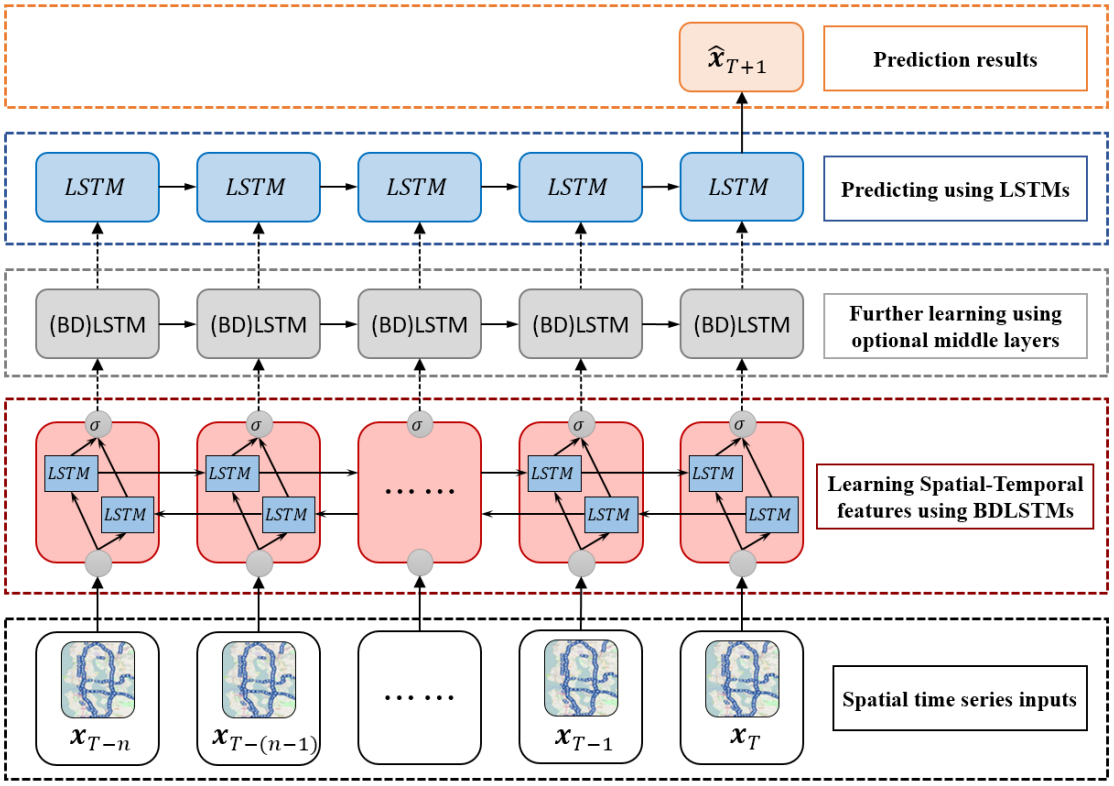


Figure 3.4: SBULSTM architecture necessarily consists of a BDLSTM layer and a LSTM layer. Intermediate LSTM or BDLSTM layers as middle layers are optional.

information of the traffic network to the BDLSTMs, both the spatial correlation of the speeds in different locations of the traffic network and the temporal dependencies of the speed values can be captured during the feature learning process. In this regard, the BDLSTMs are very suitable for being the first layer of a model to learn more useful information from spatial time series data. When predicting future speed values, the top layer of the architecture only needs to utilize learned features, namely the outputs from lower layers, to calculate iteratively along the forward direction and generate the predicted values. Thus, an LSTM layer, which is fit for capturing forward dependency, is a better choice to be the last (top) layer of the model.

In this study, we propose a novel deep architecture named stacked bidirectional and unidirectional LSTM network (SBU-LSTM) to predict the network-wide traffic speed values. Figure 3.4 illustrates the graphical architecture of the proposed model. If the input contains missing values, a masking layer should be adopted by the SBU-LSTM. Each SBU-LSTM contains a BDLSTM layer as the first feature-learning layer and a LSTM layer as the last layer. For sake of making full

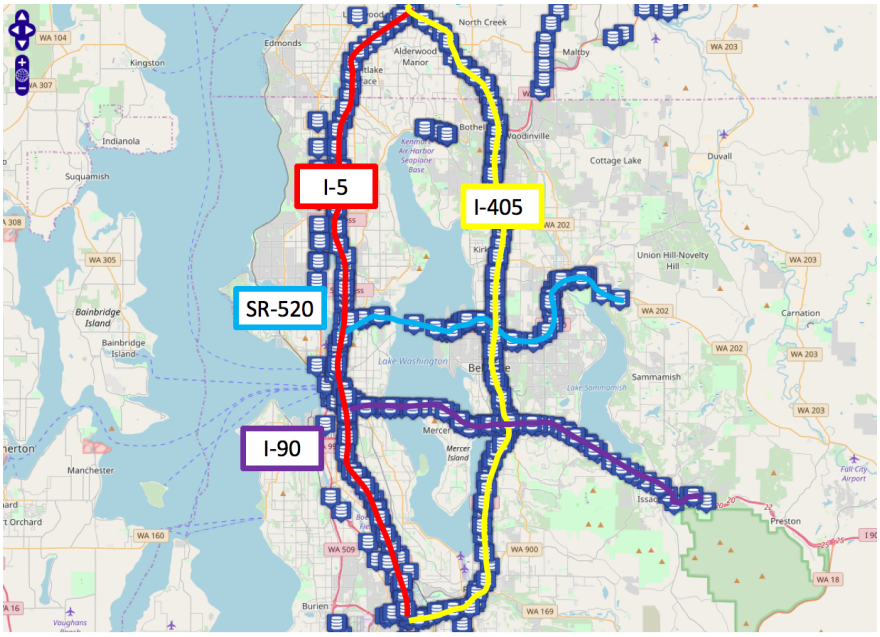


Figure 3.5: Loop detector dataset covering the freeway network in Seattle area.

use of the input data and learning complex and comprehensive features, the SBU-LSTM can be optionally filled with one or more LSTM/BDLSTM layers in the middle. Figure 3.4 shows that the SBU-LSTM takes the spatial time series data as the input and predict future speed values for one time-step. The SBU-LSTM is also capable of predicting values for multiple future time steps based on historical data. But this property is not shown in Figure 3.4, since the target of this study is to predict network-wide traffic speed for one future time step. The detailed spatial structure of input data is described in the experiment section.

3.4 EXPERIMENTAL SETTING

3.4.1 *Dataset Description*

In this study, a loop detector dataset (Wang et al., 2016), collected by inductive loop detectors deployed on roadway surface, is utilized to carry out experiments to test the proposed model. Multiple loop detectors are connected to a detector station deployed around every half a mile. The collected data from each station are grouped and aggregated as station-based traffic state

data according to directions. This aggregated and quality controlled dataset contains traffic speed, volume, and occupancy information. In the experiments, the loop detector data cover four connected freeways, which are I-5, I-405, I-90 and SR-520 in the Seattle area, and are extracted from the Digital Roadway Interactive Visualization and Evaluation Network (DRIVE Net) system (Cui et al., 2016; Ma et al., 2011). The traffic sensor stations are shown in Figure 3.5, which is represented by small blue icons. This dataset contains traffic state data of 323 sensor stations in 2015 and the time step interval of this dataset is 5 minutes.

3.4.2 *Experiment Result Analysis*

In this sub-section, only the loop detector data, due to its high data quality (Wang et al., 2016), are used to measure the performance of the proposed approach and compare with other models. Hence, the network-wide traffic is characterized by the 323 station speed values and the spatial dimension of the input data is set as, P= 323. Since, the unit of a time step in loop detector data is 5 minutes, the dataset has $\frac{60(min)}{5} \times 24\,(hour) \times 365\,(year) = 105120$ time steps in total. Suppose the number of the time lags is set as n= 10, which means the model uses a set of data with 10 consecutive time steps (covering 50 minutes) to predict the following 5-minute speed value, the dataset is separated into samples with 10 time lags and the sample size is N= 105110 $(105120 - -10)$.

Based on the descriptions of the model, each sample of the input data, $\boldsymbol{X}_T^P$, is a 2-D vector with the dimension of $[n,\, P] = [10,\, 323]$, and each sample of the output data is a 1-dimension vector with 323 components. The input of the model is a 3-D vector, whose dimension is $[N,\, n,\, P]$. Before fed into the model, all the samples are randomized and divided into training set, validation set, and test set with the ratio 7:2:1.

In the training process, mini-batch gradient descent method is used when the model optimizes the mean squared error (MSE) loss using RMSProp optimizer and early stopping mechanism is used to avoid over-fitting. To measure the effectiveness of different traffic speed prediction

algorithms, the Mean Absolute Errors (MAE) and Mean Absolute Percentage Errors (MAPE) are computed using the following equations:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i| \tag{3.12}$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}|\frac{x_i - \hat{x}_i}{x_i}| \tag{3.13}$$

where x_i is the observed traffic speed, and $\hat{x}_i$ is the predicted speed. All the compared models in this section are trained and tested multiple times to eliminate outliers, and the results of them presented are averaged to reduce random errors.

In this section, the results of the proposed SBU-LSTMs are analyzed and compared with classical methods and other RNN-based models. Further analysis about the influence of the number of time lags, the dimension of weight matrices, the number of layers, the impact of volume and occupancy information, spatial feature learning, and model robustness are carried out to shed more light on the characteristics of proposed model.

3.5 NUMERICAL RESULTS AND ANALYSIS

3.5.1 *Performance Comparison*

3.5.1.1 *Single Site Traffic Prediction*

Several classical baseline models used in traffic forecasting problems, like ARIMA (Zhao et al., 2017) Support Vector Regression (SVR) (Wu et al., 2004), Kalman filter (Guo et al., 2014). Based on our literature review (Ma et al., 2015), the performances of ARIMA and Kalman filter method are far behind the others, and thus, these two methods are not compared in this study. Most of mentioned classical models are not suitable for predicting network-wide traffic speed via a single model, since they normally cannot process 3-D spatial temporal vectors. To compare our proposed model with these baseline models, experiments are carried out for single loop detector stations, whose input data is a 2-D vector without spatial dimension. The results are averaged to measure the overall performance of these models.

Table 3.1: Performance comparison of the proposed model with other baseline models for single detector stations speed prediction.

Models	MAE (mph)	MAPE (%)
SVM	9.23	20.39
Random Forest	2.64	6.3
Feed-forward NN (3-layers)	2.63	6.41
GRU NN	3.43	8.02
SBU-LSTMs	2.42	5.67

We compared the performance of the SBU-LSTMs with SVR, random forest, feed-forward NN, GRU NN. In this comparison, the proposed model does not use the masking layer and optional middle layers. Among these baseline models, the feed-forward NN model, also called Multi-layer Perceptron (MLP), has superior performance for the traffic flow prediction (Lv et al., 2015), and decision tree and SVR are very efficient models for prediction (Chen et al., 2016; Wu et al., 2004). For the SVR method, the Radial Basis Function (RBF) kernel is utilized, and for the Random Forest method, 10 trees are built, and no maximum depth of the trees is limited. In this experiment, the feed-forward NN model consists of two hidden layers with 323 nodes in each layer.

Table 3.1 demonstrates the prediction performance of different algorithms for the single detector stations. The number of input time lags in this experiment is set as 10. Among the non-neural network algorithms, random forest performs much better, with the MAE of 2.64, than the SVM method, which makes sense due to the majority votes mechanism of random forest. The feed-forward NN whose MAE is 2.63 performs very close to the random forest method. Although GRU NN is a kind of recurrent NN, its performance obviously cannot outperform those of feed-forward NN and random forest. The single layer structure and the simplified gates in GRU NN may be the reasons. To sum up, the proposed SBU-LSTM model is clearly superior to the other four methods in this single detector station based experiment.

Table 3.2: Performance comparison of the proposed model with other LSTM-based models for network-wide traffic speed prediction

Model	Number of LSTM / BDLSTM layers									
	N=0		N=1		N=2		N=3		N=4	
	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE
N-layers LSTM	N/A	N/A	2.886	6.585	2.502	5.929	2.483	5.95	2.529	6.114
N-layers LSTM + 1-layer DNN	N/A	N/A	2.652	6.489	2.581	6.332	2.63	6.438	2.646	6.586
N-layers LSTM + 1-layer DNN	N/A	N/A	2.668	6.506	2.557	6.274	2.595	6.447	2.647	6.602
N-layers BDLSTM	N/A	N/A	3.021	6.758	2.472	5.819	2.476	5.846	2.526	5.988
SBU-LSTMs: 1-layer BDLSTM + **N** middle BDLSTM layers + 1-layer LSTM	**2.426**	**5.674**	2.465	5.787	2.502	5.95	2.549	6.191	2.576	6.227

3.5.1.2 *Network-wide Traffic Prediction*

The SBU-LSTMs is proposed aiming at predicting the network-wide traffic speed, and thus, other methods with the ability of predicting multi-dimensional time series data are compared in this section. Since the proposed model combines BDLSTMs and LSTMs, the pure deep (N-layers) BDLSTMs and LSTMs are compared. A deep LSTM NN adding a fully connected deep neural network (DNN) layer, which is proven to be able to boost the LSTM NN (Sainath et al., 2015), is also compared. To measure the influence of temporal information to the network-wide traffic speed, a multi-layer LSTM model combining day of week and hour of day is also tested in this experiment.

Meanwhile, the influence of depth of the neural networks, namely the number of layers of the models, is tested in this section. All the experiments undertook in this section used the dataset covering the whole traffic network with 10-time lags. The number of time lags, 10, is set within a reasonable range for traffic forecasting based on existing literature (Lv et al., 2015; Yu et al., 2017b) and our experiments. The spatial dimension of weight matrices in each LSTM or BDLSTM layer in this experiment is set as the number of loop detector stations, 323, to ensure the spatial feature can be fully captured. The comparison results are averaged from multiple tests to remove random errors.

Table 3.2 shows the comparison results, where the headers on horizontal axis show the amount of the LSTM or BDLSTM layers owned by the models. In terms of the influence of depth of the neural network, all the compared models achieve their best performance when they have two layers and their performances have the same trends that the values of MAE and MAPE increase as the number of layers increases from two to four. Table 3.2 contains a special "(N=0)" column, denoting no middle layer, to represent the basic structure of the SBU-LSTM. The performance of SBU-LSTM is in conformity with the trends of the compared models that the MAE and MAPE increase as the number of layers rises from zero to four.

The proposed SBU-LSTM outperforms the others for all the layer numbers. When the SBU-LSTM has no middle layer, it achieves the best MAE, 2.426 mph, and MAPE, 5.674%. The test errors of multi-layer LSTM NN and BDLSTM NN turn out to be larger than that of the proposed model. They achieve their best MAEs of 2.502 and 2.472, respectively, when they both have two layers. It should be noted that, for the one-layer case, the BDLSTM NN model gets the worst performance in our experiments shown in the Table 3.2. It indicates that one-layer BDLSTM may be good enough for capturing features, but it is not satisfactory to predict the results. Except for the one-layer case, the model combining deep LSTM and DNN are not comparable with others. This test results show that adding DNN layers to deep LSTM cannot make improvements for the network-wide traffic prediction problem is consistent with the finding in a previous study (Sainath et al., 2015). The performance of the temporal information added multi-layer LSTM is very close to that of the LSTM combined with DNN. Thus, incorporating the day of week and time of day features cannot improve the performance for this study. This is in accordance with the results of previous works (Chen et al., 2016; Wu and Tan, 2016).

3.5.2 *Spatial-temporal Influential Factor Analysis*

3.5.2.1 *Impact of the Number of Time Lags*

The number of time lags, n, is the temporal dimension of the input data, which may influence the performance of the proposed model. Figure 3.6 shows the boxplot of the MAE versus the number of time lags, in which the spatial dimensions of all weight matrices are all set as $P = 323$. When

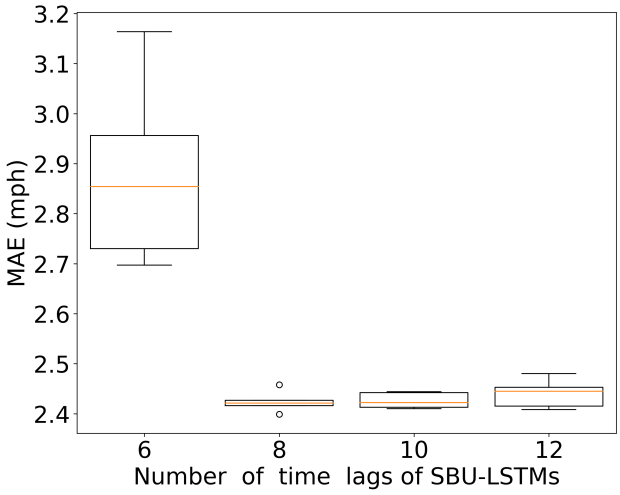


Figure 3.6: Boxplot of MAE versus number of time lags in SBU-LSTMs. (unit of time lag is 5 minutes)

the number of time lags equals 8, 10, and 12, the MAEs are very close, around 2.4. The deviations of these MAEs are relatively small. When the number of time lags is set as 6, the MAE is much higher, and the deviation is much larger than other cases. That means, given the 5-minute time step interval and the studied traffic network, input data with 6 time steps are not enough for the model to accurately predict network-wide traffic speed. To sum up, the number of time lags tends to influence the predictive performance, especially when the number is relatively small.

3.5.2.2 *Impact of the Dimension of Weight Matrices*

In the experiment, the dimension of each data sample is $[n,\ P]$, where P is the spatial dimension representing the number of loop detector stations. According to the matrix multiplication rule, the spatial dimension of the weight matrices in the first layer of the SBU-LSTM must be accordance with the value of P. But the spatial dimension of weight matrices in other layers can be customized. In this section, we measure the influence of the dimension of weight matrices in the basic SBU-LSTM.

When the model's last LSTM layer has different spatial dimensions, including $\lceil \frac{1}{4}P \rceil$, $\lceil \frac{1}{2}P \rceil$, P, $2P$ and $4P$, very close prediction results are observed. Here, P equals 323 and $\lceil \cdot \rceil$ is the ceiling function. Table 3.3 shows the comparison results. The MAE, MAPE, and standard deviations are nearly the same. Hence, the variation of the dimension of the weight matrices in the LSTM layer

Table 3.3: Performances comparison of SBU-LSTMs with different spatial dimensions of weight matrices.

Spatial dimension	MAE	MAPE	STD
1/4P	2.486	5.903	0.675
1/2P	2.425	5.68	0.643
P (=323)	2.426	5.674	0.63
2P	2.431	5.736	0.636
4P	2.411	5.696	0.636

almost has no influence on the predictive performance, if the dimension is set as a reasonable value close to the number of sensor locations.

3.5.2.3 *Spatial Features Learning*

Spatial features of a traffic network are critical for predicting network-wide traffic states. By carefully studying the LSTM methodology, we can find that the spatial features can be inherently learned by the weights in LSTM or BDLSTM layers at the training process. No matter what the network's spatial structure is, and no matter what the spatial order of the input data is, the traffic speed relationship between each pair of two locations in the traffic network can be captured by the LSTM weight matrices.

In this section, we measure the influence of spatial order of the input data on the spatial feature learning. Firstly, we order the spatial dimension of input data based on the milepost and direction of freeways. Figure 3.7 displays the heatmap of true speed and predicted speed for the freeway network on a randomly selected day, taking 09/01/2015, a Friday, for an example. The extremely similarity between the shapes in the two heatmaps shows that the proposed model is capable of learning spatial features. Then, we randomly rearrange the spatial dimension of input data. By training and testing the model for multiple times, we find that the predictive performance nearly does not change, and the MAEs are all around 2.42 mph. To the best of our knowledge, at least two aspects of reasons lead the good performance. One is that the BDLSTM, measuring both forward and backward dependencies, helps learn better features. The other one is that the

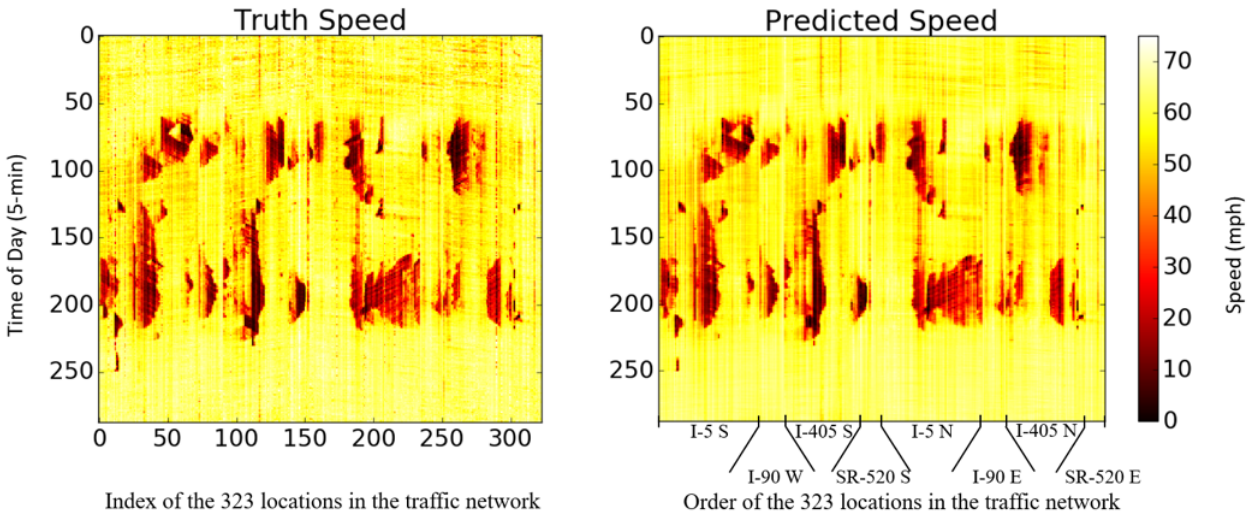


Figure 3.7: Heatmaps of ground truth and predicted speed values for the freeway traffic network on 01/09/2015. The two plots share the same meanings of the two axes, where the two horizontal axes represent the index and the arrangement order of sensor stations based on the mileposts and directions of the four freeways, respectively.

inherent spatial correlation between locations is learned and stored in the weight matrices during the training process. Hence, the order of spatial dimension of input data basically does not affect the model performance.

3.5.2.4 *Influence of volume and occupancy*

Speed, volume (flow), and occupancy are the three fundamental factors to analyze traffic flow. Considering the loop detector data contains speed, volume, and occupancy information, it is informative to investigate the influence of these factors on the proposed model's predictive performance. In previous experiments, each element of the model input, x_t^p, is the speed (s) at a specific location, p, at time t, where $x_t^p = s_t^p$. While, in this experiment, an element of the model input combine speed (s) with volume (v) and occupancy (o), where x_t^p can be $(s, o)_t^p$, $(s, v)_t^p$, or $(s, v, o)_t^p$.

Before investigating the influence of volume and occupancy, the relationship between these factors are directly evaluated by three scatter diagrams, plotted based on the loop detector dataset and shown in Figure 3.8. Although obvious noise points can be seen from Figure 3.8 (b) and (c), the main distributions in the three diagrams follows the traffic flow theory [47]. Our experiments

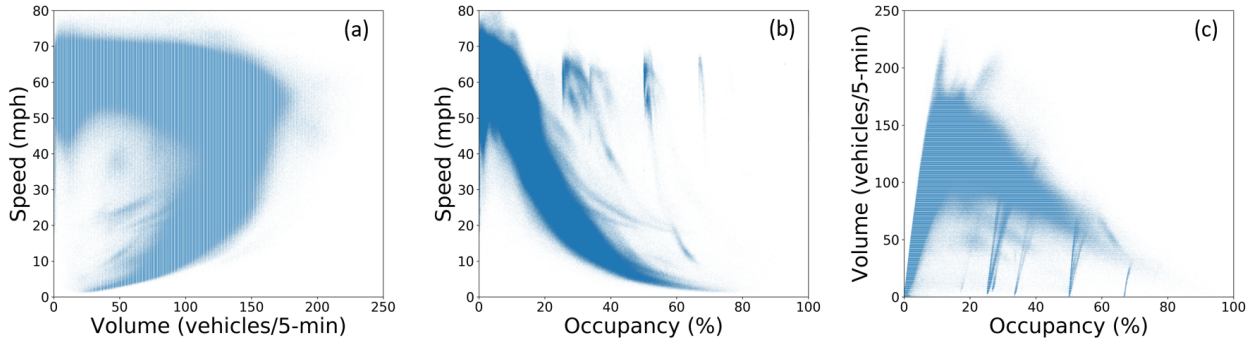


Figure 3.8: Fundamental scatter diagrams of traffic flow: (a) speed-volume diagram, (b) speed-occupancy diagram, and (c) volume-occupancy diagram.

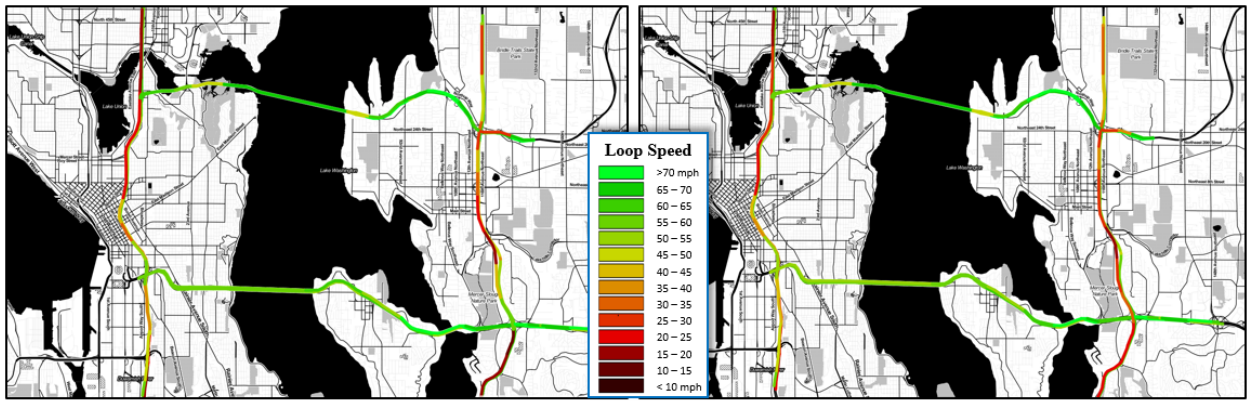


Figure 3.9: Visualization of ground truth (left) and predicted (right) traffic speed on the Seattle freeway network.

show that when solely combining volume data, there is nearly no improvement over the prediction accuracy shown in the Section *2)*. But when inputting speed and occupancy, or all the three factors, the model performs slightly better, which has less than 5% increase in the prediction accuracy. Therefore, the volume and occupancy has slightly influence on the traffic speed prediction based on our experiment results.

3.5.3 *Visualization and Potential Applications*

Besides theoretical contribution, the proposed model has potential impact on the traffic speed prediction related applications. Figure 3.9 presents the ground truth and predicted network-wide traffic speed at a specific time point. By investigating the predicted traffic speed for single locations,

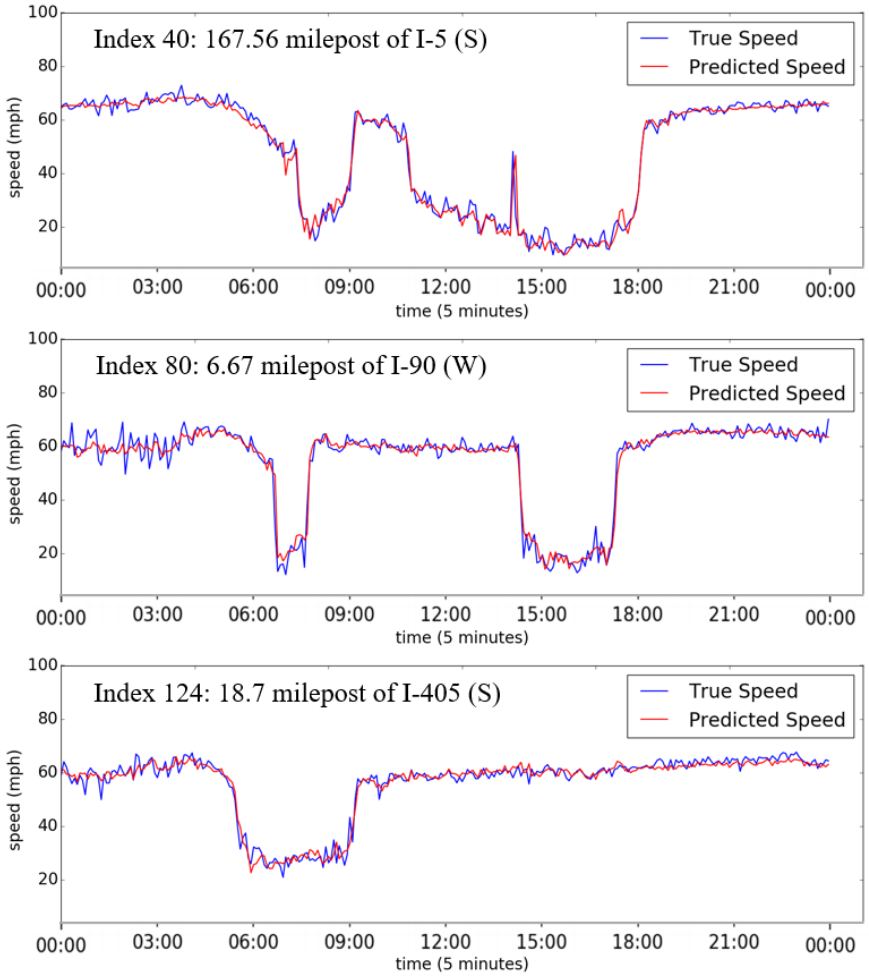


Figure 3.10: Comparison between ground truth and predictions of four different locations on 09/10/2015.

we find that the prediction performance is also very good and the trends of predicted and true values are pretty similar. Figure 3.10 shows three plots of comparison ground truth and predictions of four different locations on 09/01/2015. The three subplots in Figure 3.10 presenting totally different traffic patterns show that the prediction values are pretty accurate, which proves that SBU-LSTM has the ability to predict different patterns over the traffic network at the same time.

3.6 CHAPTER SUMMARY

A deep stacked bidirectional and unidirectional LSTM neural network is proposed in this chapter for network-wide traffic speed prediction. The improvements and contributions in this study mainly focus on four aspects: 1) we expand the traffic forecasting area to the whole traffic

network, including freeway and urban traffic networks; 2) we propose a deep architecture stacked architecture considering both forward and backward dependencies of network-wide traffic data; 3) multiple influential factors for the proposed model are analysed in detail; and 4) a masking mechanism is adopted to handle missing values.

Experiment results indicate that the two-layers SBU-LSTM without middle layers is the best structure for network-wide traffic speed prediction. Comparing to LSTM, BDLSTM and other LSTM-based methods, the structure of stacking BDLSTM and LSTM layers turns out to be more efficient to learn spatial-temporal features from the dataset. If the number of time lags of historical data is not large enough, prediction performance may decrease. The spatial order of input data and the spatial dimension of weight matrices in the last layer of the model almost has no influence on the prediction results. Additional information, like volume and occupancy, cannot significantly improve the predictive performance. Further, it is proved that the proposed model is suitable for predicting traffic speed on different types of traffic network.

Further improvements and extensions can be made based on this study. The model will be improved towards graph-based structure to learn and interpret spatial features. The model will be implemented on an artificial intelligence based transportation analytical platform. Potential applications, like non-recurring congestion detection, will be explored by combining other datasets. [1]

1 This chapter is a slightly modified version of "**Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction**" presented in *6th International Workshop on Urban Computing (UrbComp 2017).*

4

TRAFFIC GRAPH CONVOLUTIONAL RECURRENT NEURAL NETWORK: INCORPORATING ROADWAY PHYSICAL PROPERTIES INTO NETWORK-SCALE TRAFFIC PREDICTION

4.1 OVERVIEW

Traffic forecasting is one of the most challenging components of Intelligent Transportation Systems (ITS). The goal of traffic forecasting is to predict future traffic states in the traffic network given a sequence of historical traffic states and the physical roadway network. Since the volume and variety of traffic data has been increasing in recent years, data-driven traffic forecasting methods have shown considerable promise in their ability to outperform conventional and simulation-based methods (Yu et al., 2017a).

Previous work (Ma et al., 2015; Park and Rilett, 1999; Vlahogianni et al., 2014) on this topic roughly categorizes existing models into two categories: classical statistical methods and machine learning models. Most of the studies focusing on traffic forecasting using statistical methods were developed when traffic systems were less complex, and the sizes of traffic datasets were relatively small. However, statistical models' capability of handling high dimensional time series data is quite limited. With the more recent rapid development in computational power, as well as growth in traffic data volume, much of the more recent work on this topic focuses on machine learning methods for traffic forecasting.

Machine learning methods with the capability of capturing complex non-linear relationships, like support vector regression (SVR) (Smola and Schölkopf, 2004), tend to outperform the statistical methods, such as autoregressive integrated moving average (ARIMA) (Hamed et al., 1995) and its

variants, with respect to handling complex traffic forecasting problems (Ma et al., 2015). However, the full potential of artificial intelligence approaches to traffic forecasting was not exploited until the rise of deep neural network (NN) models (also referred to as deep learning models). Following early works (Park and Rilett, 1999; Van Lint et al., 2002) applying NNs to the traffic prediction problem, many NN-based methods have been adopted for traffic forecasting.

4.1.1 *Background of Deep Learning-based Traffic Forecasting*

Deep learning models for traffic forecasting, such as deep belief networks (DBN) (Huang et al., 2014) and stacked auto-encoders (Lv et al., 2015), can effectively learn high dimensional features and achieve good forecasting performance. Recurrent neural network (RNN) and its variants, including long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014) networks, have also shown great potential for solving traffic forecasting problems (Cui et al., 2017; Kong et al., 2019; Ma et al., 2015; Yu et al., 2017b). Although RNN-based methods can learn the spatial dependencies, they tend to be over-complex and inevitably capture a certain amount of noise and spurious relationships which likely do not represent the true causal structure in a physical traffic network. Moreover, interpreting the network parameters in terms of real-world spatial dependencies is most often impossible. To address this, other works (Ma et al., 2017; Yu et al., 2017a; Zhang et al., 2017) attempt to model spatial dependencies with convolutional neural network (CNN). However, conventional CNNs are most appropriate for spatial relationships in the Euclidean space as represented by two-dimensional (2D) matrices or images. Thus, spatial features learned in CNN are not optimal for representing the traffic network structure (Li et al., 2018; Yu et al., 2019).

Recently, substantial research has focused on extending the convolution operator to more general, graph-structured data, which can be applied to capture the spatial relationships present in a traffic network. There are two primary ways to conduct graph convolution. The first class of methods (Atwood and Towsley, 2016; Defferrard et al., 2016; Estrach et al., 2014; Kipf and Welling, 2019) makes use of spectral graph theory, by designing spectral filter/convolutions based on the graph Laplacian matrix. Spectral-based graph convolution has been adopted and combined with RNN (Li et al., 2018) and CNN (Yu et al., 2017a) to forecast traffic states. These models successfully

apply convolution to graph-structured data, but they do not fully capture the unique properties of graphs (Zhou and Li, 2017), like traffic networks. These models (Defferrard et al., 2016; Henaff et al., 2015), usually adopt multiple graph convolution layers, and thus, their learned spatial dependencies are hard to interpret. The other form of graph convolution proposed in several newly-published studies is conducted on graph data dynamically, for example, the dynamic edge-conditioned filters in graph convolution (Simonovsky and Komodakis, 2017), the high-order adaptive graph convolutional network (Abu-El-Haija et al., 2019; Zhou and Li, 2017). Still, these methods are not capable of fully accommodating the physical specialties of traffic networks.

One of the deficiencies of the previous graph convolution-based models is that the receptive field of the convolution operators is not confined in the graph according to the real structure of the traffic network. The traffic states of two locations far apart from each other in the traffic network should not be influenced by each other in a short time period. Though the spectral graph convolution models (Defferrard et al., 2016; Li et al., 2018)can capture features from K-localized neighbors of a vertex in the graph, how to choose the value of K and whether the localized neighbors truly affect the vertex are still questions to be answered. Thus, we propose a free-flow reachable matrix based on the free-flow speed of the real traffic and apply it on the graph convolution operator to learn features from the truly influential neighborhood in the traffic network.

4.1.2 *Contribution and Organization of the Chapter*

In this chapter, we learn the traffic network as a graph and conduct convolution on the traffic network-based graph. To learn localized features and incorporate roadway physical characteristics, we proposed a traffic graph convolution operator. Base on this operator, we propose a traffic graph convolutional LSTM (TGC-LSTM) to model the dynamics of the traffic flow and capture the spatial dependencies. Evaluation results show that the proposed TGC-LSTM outperforms multiple state-of-the-art traffic forecasting baselines. More importantly, the proposed model turns out to be capable of identifying the most influential roadway segments in the real-world traffic networks. The main contributions of our work include:

1. A traffic graph convolution operator is proposed to accommodate physical specialties of traffic networks and extract comprehensive features.

2. A traffic graph convolutional LSTM neural network is proposed to learn the complex spatial and dynamic temporal dependencies presented in traffic data.

3. To make learned localized graph convolution features more consistent and interpretable, we proposed two regularization terms, including an L1-norm on traffic the graph convolution weights and an L2-norm on the traffic graph convolution features, that can be optionally added to the model's loss function.

4. The real-world traffic speed data, including the graph structure of the traffic network, used in this chapter is published via a publicly available website[1] to facilitate further research on this problem.

The remainder of the chapter is organized as follows: In Section 4.2 presents existing and representative related work. We introduce several notions that will be used for building the traffic graph convolution module in Section 4.3. Then, we introduce the traffic graph convolution module and compare it with other related graph convolution modules in Section 4.4. Further in Section 4.5, we introduce the proposed TGC-LSTM neural network, as well as two regularization terms. Section 4.6 shows the experimental settings. Section 4.7 presents the experimental results and related analysis. Section 4.8 concludes the chapter by showing the contribution of the proposed model and identifying future directions.

4.2 RELATED WORK

Deep learning models have shown their superior capabilities of capturing nonlinear spatiotemporal effects for traffic forecasting (Polson and Sokolov, 2017). Ever since the precursory study (Hua and Faghri, 1994) using the feed-forward NN for vehicle travel time estimation was proposed, many other NN-based models, including fuzzy NN (Yin et al., 2002), recurrent NN (Van Lint et al., 2002), convolution NN (Ma et al., 2017; Yu et al., 2017a), deep belief networks (Huang et al.,

1 https://github.com/zhiyongc/Seattle-Loop-Data

2014; Kong et al., 2019), auto-encoders (Liao et al., 2018; Lv et al., 2015), generative adversarial networks (Liang et al., 2018; Lin et al., 2019), and combinations of these models have been applied to forecast traffic states. With the capability of capturing temporal dependencies, the recurrent NN or its variants, like LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014), was widely adopted as a component of a traffic forecasting model to forecast traffic speed (Ma et al., 2015), travel time (Duan et al., 2016a), and traffic flow (Zhao et al., 2017).

Further, in most recent years, various novel deep learning-based traffic forecasting models have been proposed through adjusting classical neural network model, combining existing methods, and incorporating auxiliary data. Multiple novel LSTM based models, such as bidirectional LSTM (Cui et al., 2017), deep LSTM (Yu et al., 2017b), and nested LSTM (Ma et al., 2020), have been designed via reorganizing and combing single LSTM models and applied to capture comprehensive temporal dependencies for traffic prediction. In addition, sequence-to-sequence (seq2seq) architecture based models (Li et al., 2018; Liao et al., 2018) have also been used for traffic state sequence forecasting. To deal with different types of features, multi-stream deep learning models (Ke et al., 2020; Wu et al., 2018; Yu et al., 2017b; Zhang and Patras, 2018) have also been well studied and tested for traffic forecasting problems. To improve the prediction performance, multiple deep learning based models also incorporate various traffic-related auxiliary data, including roadway geographical attribute data (Liao et al., 2018), accident data (Yu et al., 2017a), and weather data (Zhang and Kabuka, 2018).

To capture spatial relationships present in traffic networks, many forecasting models (Jin et al., 2018; Ma et al., 2017) incorporating CNNs to extract spatial features from 2D spatial-temporal traffic data. Due to the traffic structure is hard to be depicted by 2D spatial-temporal data, studies (Yu et al., 2017a) tried to convert traffic network structures to images and use CNNs to learn spatial features. However, these converted images have a certain amount of noise, inevitably resulting in spurious spatial relationships captured by CNNs. Recent studies (Chen et al., 2018a; Guo et al., 2019; Zhang and Kabuka, 2018) also attempted to convert traffic state data into three-dimensional (3D) matrices and use the 3D convolutional network to extract more effective features. However, conventional CNN based methods still cannot inherently deal with the topological structure and the physical attributes of the traffic network. To solve this problem, studies (Li et al., 2018; Yu et al.,

2018) attempted to learn the traffic network as a graph and adopt the graph-based convolution operator to extract features from the graph-structured traffic network.

Traffic networks have already been analyzed as graphs for dynamic shortest path routing (Sun et al., 2017), traffic congestion analysis (Sun et al., 2014), and dynamic traffic assignment (Kalafatas, 2007). In the last couple of years, many studies attempt to generalize neural networks to work on arbitrarily structured graphs by designing graph convolutional networks. Generally, the graph convolutional networks utilize the adjacency matrix or the Laplacian matrix to depict the structure of a graph. The Laplacian matrix based graph convolution (Estrach et al., 2014; Henaff et al., 2015) are designed based on the spectral graph theory (Shuman et al., 2013). As an extension, a localized spectral graph convolution (Defferrard et al., 2016) is also proposed to reduce the learning complexity. The adjacency matrix based graph convolution neural networks (Kipf and Welling, 2019; Zhou and Li, 2017) incorporate the adjacency matrix and their network structures are more flexible. The traffic network can be considered as a graph consisting of nodes and edges, and thus, several graph convolution neural network based models, including the spectral graph convolution (Yu et al., 2018) and the diffusion graph convolution (Atwood and Towsley, 2016), are proposed to fulfill network-wide traffic forecasting. Several studies (Geng et al., 2019; Zhang and Kabuka, 2018) also incorporated multi-scale graph convolution operations into their proposed models to learn traffic features. Although these existing methods can extract spatial features from neighborhoods in the traffic network, the physical specialties of roadways, like length, speed limits, and the number of lanes, are normally neglected.

4.3 PRELIMINARIES AND NOTIONS

4.3.1 *Traffic Network based Graph*

Normally, a graph consists of nodes (vertices) and edges. The graph representing a traffic network is distinct from social network graphs, document citation graphs, or molecule graphs, in several respects: 1) there are no isolated nodes/edges in traffic network based graphs and the traffic network structure seldom changes; 2) the traffic status of each road in a traffic network varies over time; and 3) the roads in a traffic network have meaningful physical characteristics, such as

the length, type, speed limit, and lane numbers of a road. Further, traffic state data is collected by different types of sensors such that some types of data detect location-based traffic states, but others may measure road segment based averaged traffic states. Due to traffic states vary over time, it is better to let the graph nodes possess the varying traffic states and keep the graph structure fixed. Thus, to ensure the consistency of the definition in a graph, we use **nodes** to represent the traffic sensing locations, which can be sensor stations or road segments. Then, the **edges** in a graph represent the intersections or road segments connecting those traffic sensing locations.

The traffic network and the relationship between traffic locations can be represented by an undirected graph $\mathcal{G}$ where $\mathcal{G}=(\mathcal{V},\mathcal{E})$ with N nodes $v_i \in \mathcal{V}$ and edges $(v_i, v_j) \in \mathcal{E}$. Even though some roads are directed in the reality, due to the impact of traffic congestions occurring on these roads will be bi-directionally propagated to upstream and downstream roads (Cui et al., 2017), we take the bidirectional impact into account and thus let $\mathcal{G}$ be an undirected graph.

4.3.2 *Adjacency Matrix and Neighborhood Matrix*

The connectedness of nodes in $\mathcal{G}$ is represented by an adjacency matrix $A \in \mathbb{R}^{N\times N}$, in which each element $A_{i,j}=1$ if there is an edge connecting node i and node j and $A_{i,j}=0$ otherwise ($A_{i,i}=0$). Based on the adjacency matrix, the degree matrix of $\mathcal{G}$, which measures the number of edges attached to each vertex, can be defined as $D \in \mathbb{R}^{N\times N}$ in which $D_{ii}=\sum_j A_{ij}$. D is a diagonal matrix and all non-diagonal elements are zeros.

Based on the adjacency matrix, an edge counting function $d(v_i, v_j)$ can be defined as counting the minimum number of edges traversed from node i to node j. Then, the set of k-hop (k-th order) neighborhood of each node i, including node i itself, can be defined as $\{v_j \in \mathcal{V} \mid d(v_i, v_j) \leq k\}$. However, since the traffic states are time series data and the current traffic state on a road will definitely influence the future state, we consider the all roads are self-influenced. Thus, we consider the neighborhood of a node contains the node itself and a neighborhood matrix to characterize the one-hop neighborhood relationship of the whole graph, denoted as

$$\tilde{A} = A + I \tag{4.1}$$

where I is the identity matrix. Then, the k-hop neighborhood relationship of the graph nodes can be characterized by $(A+I)^k$. However, some elements in $(A+I)^k$ will inevitably exceed one. Owing to the k-hop neighborhood of a node is only used for describing the existence of all the k-hop neighbors, it is not necessary to make a node's k-hop neighbors weighted by the number of hops. Thus, we clip the values of all elements in $(A+I)^k$ to be in $\{0,1\}$ and define a new k-hop neighborhood matrix $\tilde{A}^k$, in which each element $\tilde{A}^k_{i,j}$ satisfies

$$\tilde{A}^k_{i,j} = \min\left((A+I)^k_{i,j}, 1\right) \tag{4.2}$$

where min refers to minimum. In this case, $\tilde{A}^1 = A^1 = A$. An intuitive example of k-hop neighborhood with respect to a node (a red star) is illustrated by blue points on the left side of Figure 4.1.

4.3.3 *Free-Flow Reachable Matrix*

Based on the length of each road in the traffic network, we define a distance matrix $Dist \in \mathbb{R}^{N\times N}$, where each element $Dist_{i,j}$ represents the real roadway distance from node i to j ($Dist_{i,i} = 0$). When taking the underlying physics of vehicle traffic on a road network into consideration, we need to understand that the impact of a roadway segment on adjacent segments is transmitted in two primary ways: 1) slowdowns and/or blockages propagating upstream; and 2) driver behavior and vehicle characteristics associated with a particular group of vehicles traveling downstream. Thus, for a traffic network-based graph or other similar graphs, the traffic impact transmission between non-adjacent nodes cannot bypass the intermediate node/nodes, and thus, we need to consider the reachability of the impact between adjacent and nearby node pairs. To ensure the traffic impact transmission between k-hop adjacent nodes follow the established traffic flow theory (Daganzo, 1994), we define a free-flow reachable matrix, $\mathcal{FFR} \in \mathbb{R}^{N\times N}$, that

$$\mathcal{FFR}_{i,j} = \begin{cases} 1, & S^{\mathcal{FF}}_{i,j} m\Delta t - Dist_{i,j} \geq 0 \\ 0, & otherwise \end{cases}, \forall v_i, v_j \in V \tag{4.3}$$

where $S^{\mathcal{FF}}_{i,j}$ is the free-flow speed between node i and j, and free-flow speed (Hunter-Zaworski et al., 2003) refers to the average speed that a motorist would travel if there were no congestion or

other adverse conditions (such as severe weather). Δt is the duration of time quantum and m is a number counting how many time intervals are considered to calculate the distance travelled under free-flow speed. Thus, m determines the temporal influence of formulating the $\mathcal{FFR}$. Each element $\mathcal{FFR}_{i,j}$ equals one if vehicles can traverse from node i to j in m time-step, $m \cdot \Delta t$, with free-flow speed, and $\mathcal{FFR}_{i,j} = 0$ otherwise. Intuitively, the $\mathcal{FFR}_{i,j}$ measures whether a vehicle can travel from node i to node j with the free-flow speed under a specific time interval. We consider each road is self-reachable, and thus, all diagonal values of $\mathcal{FFR}$ are set as one. An example $\mathcal{FFR}$ with respect to a node (a red star) is shown by green lines on the left side of Figure 4.1.

4.3.4 *Traffic Forecasting formulation*

Traffic forecasting refers to predicting future traffic states, such as traffic speed, travel time, or volume, given previously observed traffic states from a road network. In this chapter, the traffic network is converted into a graph consisting of all N nodes, representing N traffic sensing locations, and a set of edges. During a period of time t, the signals of these nodes representing the collected traffic states, can be denoted as $x_t \in \mathbb{R}^N$.

To formulate the traffic forecasting problem, the main aforementioned notations are summarized in the following list:

- $\mathcal{G}$: Traffic network-based graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- $\mathcal{V}$: Set of vertices in $\mathcal{G}$ with the size of $|\mathcal{V}| = N$
- $\mathcal{E}$: Set of edges in $\mathcal{G}$ with the size of $|\mathcal{E}|$
- $A \in \mathbb{R}^{N \times N}$: Adjacency matrix of $\mathcal{G}$
- $D \in \mathbb{R}^{N \times N}$: Degree matrix of $\mathcal{G}$
- $\tilde{A} \in \mathbb{R}^{N \times N}$: Neighborhood matrix defined by Equation 4.1
- $\tilde{A}^k \in \mathbb{R}^{N \times N}$: k-hop neighborhood matrix defined by Equation 4.2
- $Dist \in \mathbb{R}^{N \times N}$: Distance matrix

- $\mathcal{FFR} \in \mathbb{R}^{N \times N}$: Free-flow reachable matrix by Equation 4.4
- $x_t \in \mathbb{R}^N$: Vector of speed of all graph nodes at time t

The short-term traffic forecasting problem aims to learn a function $F(\cdot)$ to map T time steps of historical graph signals, i.e. $\boldsymbol{X}_T = [x_1, \ldots, x_t, \ldots, x_T]$, to the graph signals in the subsequent one or multiple time steps. In this chapter, the function attempts to forecast the graph signals in the subsequent one step, i.e. x_{T+1}, and the formulation of $F(\cdot)$ is defined as

$$F\left([x_1, \ldots, x_t, \ldots, x_T]; G\left(\mathcal{V}, \mathcal{E}, \tilde{A}^k, \mathcal{FFR}\right)\right) = x_{T+1} \tag{4.4}$$

Further, another goal of this study is to learn the traffic impact transmission between adjacent and neighboring nodes in a traffic network-based graph by learning the weight parameters in the function $F(\cdot)$.

4.4 TRAFFIC GRAPH CONVOLUTION

Previous work (Abu-El-Haija et al., 2019; Kipf and Welling, 2019; Zhou and Li, 2017) has defined the graph convolution based the adjacency matrix. The core idea of a convolution layer in a neural network is to extract localized features from input data in a 2D or 3D matrices structure. The localized region of the input space which affects the convolution operation results is called receptive field. Analogously, the core idea of a graph convolution layer is to extract localized features from input data in a graph structure. Thus, the product of the neighborhood matrix $\tilde{A}$, the input data x_t, and a trainable weight matrix W, i.e. $\tilde{A}x_tW$, can be considered as a graph convolution operation to extract features from one-hop neighborhood (Kipf and Welling, 2019; Zhou and Li, 2017). Then, the receptive field of the graph convolution operation on a node is the one-hop neighborhood.

However, in this way, the receptive field is confined, and it only concentrates on one-hop neighboring nodes. To overcome this shortcoming, we extend the receptive field of graph convolution by replacing the one-hop neighborhood matrix $\tilde{A}$ with the k-hop neighborhood matrix $\tilde{A}^k$. Meanwhile, existing studies either neglect the properties of the edges in a graph, such as the distances between different sensing locations (the lengths of the graph edges) and the free-flow

reachability defined in Equation 4.3, or fail to consider high-order neighborhood of nodes in the graph. Hence, to comprehensively solve the network-wide forecasting problem, we consider both graph edge properties and high-order neighborhood in the traffic network-based graph. Hence, we define the k-order (k-hop) **Traffic Graph Convolution** (TGC) operation as

$$GC_t^k = (W_{gc_k} \bigodot \tilde{A}^k \bigodot \mathcal{FFR}) x_t \tag{4.5}$$

where $\odot$ is the Hadamard product operator, i.e. the element-wise matrix multiplication operator, and $x_t \in \mathbb{R}^N$ is the vector of traffic states (speed) of all nodes at time t. The $W_{gc\_k} \in \mathbb{R}^{N \times N}$ is a trainable weight matrix for the k-order traffic graph convolution and the $GC^k \in \mathbb{R}^N$ is the extracted k-order traffic graph convolution feature. Due to $\tilde{A}^k$ and $\mathcal{FFR}$ are both sparse matrices only containing 0 and 1 elements, the result of $W_{gc_k} \bigodot \tilde{A}^k \bigodot \mathcal{FFR}$ is also sparse. Further, the trained weight W_{gc_k} has the potential to measure the interactive influence between graph nodes, and thus, enhance the interpretability of the model.

In Equation 4.5, k should be a positive integer. The larger the order k is, the larger the size of the receptive field of the TGC is, and then the more neighborhood-based features can be extracted from the graph. However, k is not infinite, and it can be easily proved that, for a specific graph, when increasing the value of k, $\tilde{A}^k \bigodot \mathcal{FFR}$ will eventually converge to $\mathcal{FFR}$ such that $k = K_{max}$ and $\tilde{A}^{K_{max}} \bigodot \mathcal{FFR} = \mathcal{FFR}$. It should be noted that, while extracting traffic graph convolution features to solve real traffic prediction problems, it is not necessary to set k as the max value K_{max}. The trade-off between the prediction accuracy and the feature richness, which is directly related to the computational cost, should be considered and balanced.

Let $K \leq K_{max}$ denote the largest hop for traffic graph convolution in this chapter, and the corresponding traffic graph convolution feature is GC_t^K with respect to input data x_t. Different hops of neighborhood in TGC will result in different extracted features. To enrich the feature space, the features extracted from different orders (from 1 to K) of traffic graph convolution with respect to X_t are concatenated together as a vector defined as follows

$$\boldsymbol{GC}_t^K = [GC_t^1, GC_t^2, \ldots, GC_t^K] \tag{4.6}$$

The $\boldsymbol{GC}_t^{\{K\}} \in \mathbb{R}^{N \times K}$ contains all the K orders of traffic graph convolutional features, as intuitively shown in the left part of Figure 4.1. In this chapter, after operating the TGC on input data x_t, the

generated $\boldsymbol{GC}_t^{\{K\}}$ will be fed into the following layer in the proposed neural network structure described in the following section.

Comparing TGC with Spectral Graph Convolution: The proposed traffic graph convolution is based on adjacency matrix A, but the spectral graph convolution (SGC) is defined in the Fourier domain (Shuman et al., 2013) based on the Laplacian matrix L, which equals

$$L{=}D{-}A \tag{4.7}$$

where D is the degree matrix. The Laplacian matrix L is symmetric positive semi-definite such that it can be diagonalized via eigen-decomposition as

$$L{=}U\Lambda U^T \tag{4.8}$$

where Λ is a diagonal matrix containing the eigenvalues, U consists of the eigenvectors, and U^T is the transpose of U.

The spectral convolution on graph is defined as the multiplication of a signal $x_t{\in}\mathbb{R}^N$ with a filter $h_\theta = \text{diag}\,(\theta)$ parameterized by $\theta{\in}\mathbb{R}^N$ (Kipf and Welling, 2019). The $\text{diag}\,(\theta)$ is the diagonalized matrix given θ. The spectral graph convolution operation can be described as

$$h_\theta *_{\mathcal{G}} x_t{=}Uh_\theta U^T x_t = U\text{diag}\,(\theta)\,U^T x_t \tag{4.9}$$

where $*_{\mathcal{G}}$ is the spectral graph convolution operator. The filter h_θ that can be considered as a learnable convolutional kernel weight.

Further, for saving computational cost, the localized spectral graph convolution (LSGC) is proposed by employing a polynomial filter $h_{\theta'} = \sum_{j=0}^{k-1}\theta'_j\Lambda^j$ (Defferrard et al., 2016) and the learnable parameter $\theta' \in \mathbb{R}^K$. Then K-hop localized spectral graph convolution can be formulated as:

$$h_{\theta'} *_{\mathcal{G}} x_t{=}U\sum_{j=0}^{K-1}\theta'_j\Lambda^j U^T x_t{=}\sum_{j=0}^{K-1}\theta'_j L^j x_t \tag{4.10}$$

The advantages of the LSGC is that it only has K parameters and does not need eigen-decomposition. It is well spatial localized and each convolution operation on a centered vertex extracts the summed weighted feature of the vertex's K-hop neighbors. The details of SGC and LSGC can be found in the literature (Defferrard et al., 2016; Estrach et al., 2014; Kipf and Welling, 2019).

Table 4.1: Comparison between TGC, SGC, and LSGC

Graph convolution definition	K-hop TGC	SGC	K-hop LSGC
Graph convolution on signal x_t	$(W_{gc_k} \odot \tilde{A}^k \odot FFR)$	$U\text{diag}(\theta)U^T$	$\sum_{j=1}^{K-1} \theta'_j L^j$
Weight parameters	$W_{gc_k} \in \mathbb{R}^{N \times N}$	$\theta \in \mathbb{R}^N$	$\theta' \in \mathbb{R}^K$
Computational time complexity	$O(N^2)$	$O(N^2)$	$O(K\|\epsilon\|)$
Extract Localized features	Yes. k-localized incorporating roadway physical properties	No	Yes. It is exactly k-localized.

The comparison between TGC, SGC, and LSGC in terms of the number of parameters, computational time, and localized feature extraction, is shown in Table 4.1. Comparing to SGC and LSGC, the TGC is better in terms of spatial localization because it can extract local features based on physical properties of roadways by incorporating the $\mathcal{FFR}$. TGC with more parameters has better capabilities of representing the relationships between connected nodes in the graph. Further, SGC and LSGC normally need multiple convolutional layers, which leads the SGC and LSGC to lose their interpretability. However, TGC only needs one convolution layer and its parameters can be better interpreted.

4.5 TRAFFIC GRAPH CONVOLUTIONAL LSTM

We propose a Traffic Graph Convolutional LSTM (TGC-LSTM) recurrent neural network, as shown on the right side of the Figure 4.1, which learns both the complex spatial dependencies and the dynamic temporal dependencies presented in traffic data. In this model, the gates structure in the vanilla LSTM (Hochreiter and Schmidhuber, 1997) and the hidden state are unchanged, but the input is replaced by the graph convolution features, which are reshaped into a vector $\boldsymbol{GC}^{\{K\}} \in \mathbb{R}^{KN}$. The forget gate f_t, the input gate i_t, the output gate o_t, and the input cell state $\tilde{C}_t$ in terms of time step t are defined as follows

$$\mathrm{f}_t = \sigma_g \left(W_f \cdot \boldsymbol{GC}_t^{\{K\}} + U_f \cdot h_{t-1} + b_f \right) \tag{4.11}$$

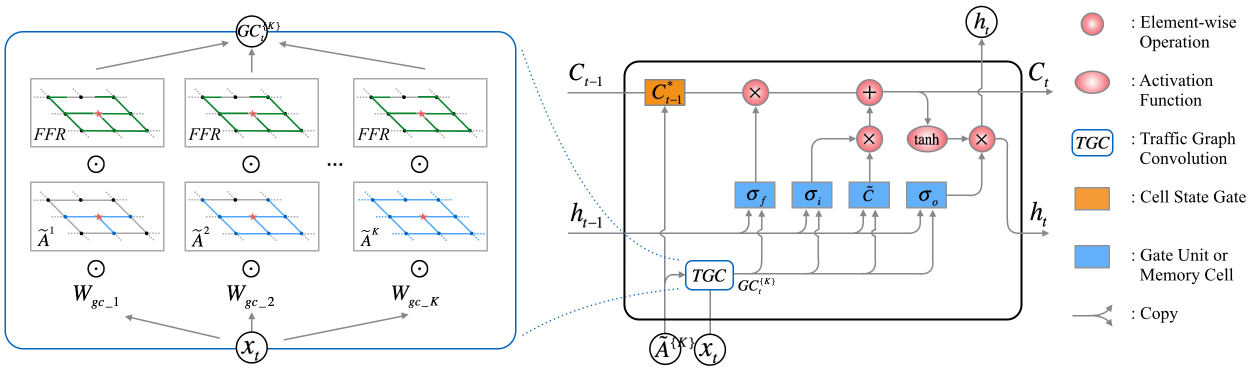


Figure 4.1: The architecture of the proposed Traffic Graph Convolution LSTM is shown on the right side. The traffic graph convolution (TGC) as a component of the proposed model is shown on the left side in detail by unfolding the traffic graph convolution at time t, in which $\tilde{A}^k$s and $\mathcal{FFR}$ respect to a red star node are demonstrated.

$$\mathrm{i}_t = \sigma_g\left(W_i \cdot \boldsymbol{GC}_t^{\{K\}} + U_i \cdot h_{t-1} + b_i\right) \tag{4.12}$$

$$\mathrm{o}_t = \sigma_g\left(W_o \cdot \boldsymbol{GC}_t^{\{K\}} + U_o \cdot h_{t-1} + b_o\right) \tag{4.13}$$

$$\tilde{C}_t = tanh\left(W_C \cdot \boldsymbol{GC}_t^{\{K\}} + U_C \cdot h_{t-1} + b_C\right) \tag{4.14}$$

where $\cdot$ is the matrix multiplication operator. W_f, W_i, W_o, and $W_C \in \mathbb{R}^{N \times KN}$ are the weight matrices, mapping the input to the three gates and the input cell state, while U_f, U_i, U_o, and $U_C \in \mathbb{R}^{N \times N}$ are the weight matrices for the preceding hidden state. b_f, b_i, b_o, and $b_C \in \mathbb{R}^N$ are four bias vectors. The σ_g is the gate activation function, which typically is the sigmoid function, and tanh is the hyperbolic tangent function.

Due to each node in a traffic network graph is influenced by the preceding states of itself and its neighboring nodes, the LSTM cell state of each node in the graph should also be affected by neighboring cell states. Thus, a cell state gate is designed and added in the LSTM cell. The cell state gate, as shown in Figure 4.1, is defined as follows

$$C_{t-1}^* = W_{\mathcal{N}} \bigodot (\tilde{A}^K \bigodot \mathcal{FFR}) \cdot C_{t-1} \tag{4.15}$$

where $W_{\mathcal{N}}$ is a weight matrix to measure the contributions of neighboring cell states. To correctly reflect the traffic network structure, the $W_{\mathcal{N}}$ is constrained by multiplying a $\mathcal{FFR}$ based K-hop adjacency matrix, $\tilde{A}^K \bigodot \mathcal{FFR}$. With this gate, the influence of neighboring cell states will be considered when the cell state is recurrently input to the subsequent time step. Then, the final cell state and the hidden state are calculated as follows

$$C_t = \mathrm{f}_t \bigodot C^*_{t-1} + \mathrm{i}_t \bigodot \tilde{C}_t \tag{4.16}$$

$$h_t = \mathrm{o}_t \bigodot \tanh\left(C_t\right) \tag{4.17}$$

At the final time step T, the hidden state h_T is the output of TGC-LSTM, namely the predicted value $\hat{y}_T = h_T$. Let $y_T \in \mathbb{R}^N$ denote the label of the input data $\boldsymbol{X}_T \in \mathbb{R}^{N \times N}$. For the sequence prediction problem in this chapter, the label of time step T is the input of the next time step $(T+1)$ such that $y_T = x_{T+1}$. Then the loss during the training process is defined as

$$Loss = \mathrm{L}\left(y_T, \hat{y}_T\right) = \mathrm{L}\left(x_{T+1}, h_T\right) \tag{4.18}$$

where $\mathrm{L}\left(\cdot\right)$ is a function to calculate the residual between the predicted value $\hat{y}_T$ and the true value y_T. Normally, the $\mathrm{L}\left(\cdot\right)$ function is a Mean Squared Error (MSE) function for predicting continuous values.

Algorithm 4.1: Calculation the output of the TGC-LSTM layer

Input: $\mathbf{X}_T = [x_1, ..., x_T], \{\tilde{A}^1, ..., \tilde{A}^K\}, \mathcal{FFR}$

Parameter: $\{W_{gc_1}, ..., W_{gc_K}\}$, W, U, and b in Eq. (4.11-4.14)

Result: the prediction result h_T

Initialize: $h_0 = 0 \in \mathbb{R}^N$, $C_0 = \in \mathbb{R}^N$;

for $t = 1$ *to* T **do**

  for $k = 1$ *to* K **do**

    $GC_t^k \leftarrow (W_{gc_k} \odot \tilde{A}^k \odot \mathcal{FFR}) x_t$

  end

  $\mathbf{GC}_t^K \leftarrow [GC_t^1, GC^1 2_t, ..., GC_t^K]$;

  $h_t, C_t = \text{TCG-LSTM}(x - t, \mathbf{GC}_t^k, h_{t-1}, C_{t-1})$

end

To explain the proposed method in a clearer way, a pseudo-code of the TGC-LSTM algorithm is presented in Algorithm 4.1. Given the traffic state data $\boldsymbol{X}_T$ and the graph related matrices as input, the pseudo-code mainly describes the process of generating the final output h_T after T steps of iteration. For simplicity, the pseudo-code does not include the mini-batch gradient descent process and the backpropagation-based parameter updating process. In Algorithm 4.1, Eq. is short for Equation and the function TGC-LSTM(·) refers to the whole calculation process described in Equations 4.11-4.17 in this section.

4.5.1 *Traffic Graph Convolution Regularization*

Since the proposed model contains a traffic graph convolution operation, the generated set of TGC features $\boldsymbol{GC}_t^{\{K\}}$ and the learned TGC weights $\{W_{gc_1}, \ldots, W_{gc_K}\}$ provide an opportunity to make the proposed model interpretable via analyzing the learned TGC weights. To confine the graph convolution features within a reasonable scale and make the learned weights more stable and interpretable, we propose two optional regularization terms that can be added to the loss function described in Equation 4.18.

4.5.1.1 *Regularization on Graph Convolution weights*

Because the graph convolution weights are not confined to be positive and each node's extracted features are influenced by multiple neighboring nodes, the graph convolution weights can vary a lot while training. Ideally, the convolution weights would be themselves informative, so that the relationships between different nodes in the network could be interpreted and visualized by plotting the convolution weights. This is not likely to be possible without regularization, because very high or low weights tend to appear somewhat randomly, with the result that high/low weights tend to cancel each other out. In combination, such weights can still represent informative features for the network, but they cannot reflect the true relationship between nodes in the graph. Thus, we add L1-norm of the graph convolution weight matrices to the loss function as a

regularization term to make these weight matrices as sparse as possible. The L1 regularization term is defined as follows

$$R^{\{1\}}=\left\|\boldsymbol{W}_{gc}\right\|_1=\sum_{i=1}^{K}\left|W_{gc_i}\right| \tag{4.19}$$

In this way, the trained graph convolution weight can be sparse and stable, and thus, it will be more intuitive to distinguish which neighboring node or group of nodes contribute most.

4.5.1.2 *Regularization on Graph Convolution features*

Considering that the impact of neighboring nodes with respect to a specific node must be transmitted through all nodes between the node of interest and the influencing node, features extracted from different hops in the graph convolution should not vary dramatically. Thus, to restrict the difference between features extracted from adjacent hops of graph convolution, an L2-norm based TGC feature regularization term is added on the loss function at each time step. The regularization term is defined as follows

$$R^{\{2\}}=\left\|\boldsymbol{GC}_{\boldsymbol{T}}^{\{K\}}\right\|_2=\sqrt{\sum_{i=1}^{K-1}\left(GC_T^i-GC_T^{i+1}\right)^2} \tag{4.20}$$

In this way, the features extracted from adjacent hops of graph convolution should not differ dramatically, and thus, the graph convolution operator should be more in keeping with the physical realities of the relationships present in a traffic network.

Then, the total loss function at time t can be defined as follows

$$Loss=\mathrm{L}\left(h_T-x_{T+1}\right)+\ \lambda_1 R^{\{1\}}+\lambda_2 R^{\{2\}} \tag{4.21}$$

where λ_1 and λ_2 are penalty terms to control the weight magnitude of the regularization terms on graph convolution weights and features.

4.6 EXPERIMENTAL SETTINGS

4.6.1 *Dataset Description*

In this chapter, two real-world network-scale traffic speed datasets are utilized. The first contains data collected from inductive loop detectors deployed on four connected freeways (I-5, I-405, I-90,

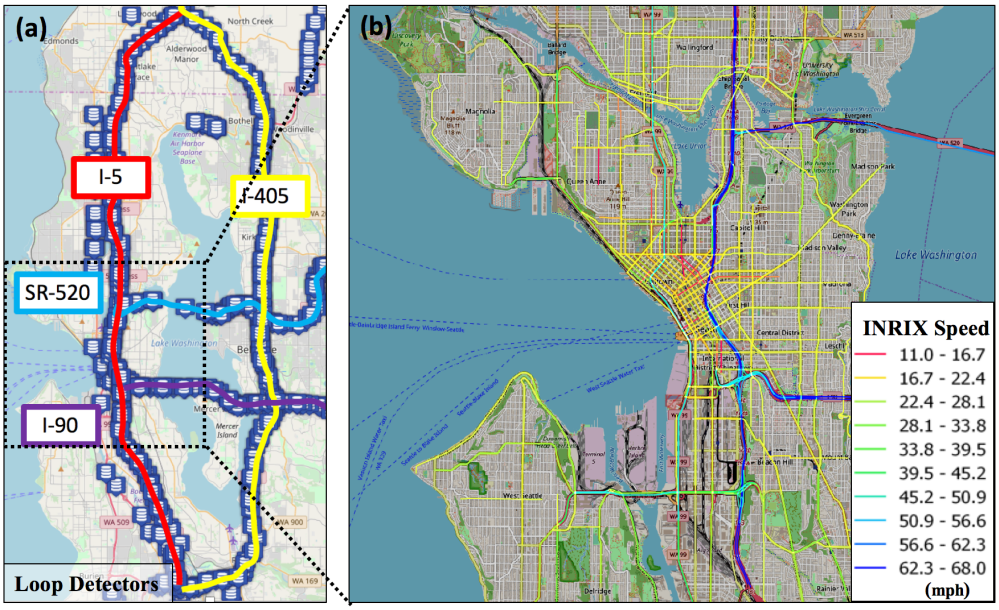


Figure 4.2: (a) LOOP dataset covering the freeway network in Seattle area; (b) INRIX dataset covering the downtown Seattle area, where traffic segments are plotted with colors.

and SR-520) in the Greater Seattle Area, shown in Figure 4.2 (a). This dataset, which is publicly accessible[2], contains traffic state data from 323 sensor stations over the entirety of 2015 at 5-minute intervals. The second contains road link-level traffic speeds aggregated from GPS probe data collected by commercial vehicle fleets and mobile apps provided by the company INRIX. The INRIX traffic network covers the Seattle downtown area, shown in Figure 4.2 (b). This dataset describes the traffic state at 5-minute intervals for 1014 road segments and covers the entire year of 2012. We use LOOP data and INRIX data to denote these two datasets, respectively, in this study.

We adopt the speed limit as the free-flow speed, which for the segments in the LOOP traffic network is 60mph in all cases. The INRIX traffic network contains freeways, ramps, arterials, and urban corridors, and so the free-flow speeds of INRIX traffic network range from 20mph to 60mph. The distance adjacency matrices $Dist$ and free-flow reachable matrices $\mathcal{FFR}$ for both datasets are calculated based on the roadway characteristics and topology.

2 https://github.com/zhiyongc/Seattle-Loop-Data

4.6.1.1 *Baselines*

We compare TGC-LSTM with the following baseline models:

1. ARIMA: Auto-Regressive Integrated Moving Average model (Hamed et al., 1995)
2. SVR: Support Vector Regression (Smola and Schölkopf, 2004)
3. FNN: Feed forward neural network with two hidden layers, i.e. the multilayer perceptron, whose hidden layer size is N;
4. LSTM: Long Short-Term Memory recurrent neural network (Hochreiter and Schmidhuber, 1997)
5. DiffGRU (Li et al., 2018): an adjusted version of diffusion convolutional gated recurrent network whose gate units are defined based on diffusion convolution. Since the graph is undirected in this study, we replace the diffusion convolution with spectral graph convolution in DiffGRU;
6. LSGC+LSTM: stacking a one-layer localized spectral graph convolution layer (Defferrard et al., 2016) whose K=3 and an LSTM layer.

All the LSTM/GRU layers have the same weight dimensions. The baseline models do not include auto-encoder based models and pure CNN based models, due to the core ideas of these methodologies are totally different from the tested baseline models which are mostly single RNN layer-based models. All the neural networks are implemented based on PyTorch 1.0.1 and they are trained and evaluated on a single NVIDIA GeForce GTX 1080 Ti with 11GB memory.

4.6.1.2 *TGC-LSTM Model*

For both datasets, the dimensions of the hidden states of the TGC-LSTM are set as the amount of the nodes in the traffic network graphs. The size of hops in the graph convolution can vary, but we set it as 3, $K = 3$, for the model evaluation and comparison in this experiment. In this case, the $\mathcal{FFR}$ is calculated based on three time steps. The two regularization terms ($R^{\{1\}}$ and $R^{\{2\}}$) can not only confine the learnt graph convolution weights, they also can avoid overfitting causing the decrease of the prediction accuracy. Thus, there is a trade-off between the prediction accuracy

and the scale of the penalty terms (λ_1 and λ_2). Based on empirically adjusting the regularization rates, the values of the λ_1 and λ_2 are both set as 0.01. We train our model by minimizing the mean square error with the batch size of 10 and the initial learning rate of 10^{-5}. Since the RMSProp (Tieleman and Hinton, 2012) can solve the gradient exploding and vanishing problems, it is used as the gradient descent optimizer whose alpha (smoothing constant) is set as 0.99 and epsilon (the term added to the denominator to improve numerical stability) is set as 10^{-8}.

4.6.1.3 *Evaluation*

In this study, the samples of the input are traffic time series data with 10 time steps. The output/label is the next subsequent data of the input sequence. The performance of the proposed and the compared models are evaluated by three commonly used metrics in traffic forecasting, including 1) Mean Absolute Error (MAE), 2) Mean Absolute Percentage Error (MAPE), and 3) Root Mean Squared Error (RMSE).

4.7 EXPERIMENTAL RESULTS

Table 4.2 demonstrates the results of the TGC-LSTM and other baseline models on the two datasets. The proposed method outperforms other models with all the three metrics on the two datasets. The ARIMA and SVR cannot compete with other methods, which suggest that non-neural-network approaches are less appropriate for this network-wide prediction task, due to the complex spatiotemporal dependencies and the high dimension features in the datasets. The basic FNN does not perform well on predicting spatial-temporal sequence. The DiffGRU performs nearly the same as the FNN. The reason might be that GRU has the no cell state to store historical information in its gate units comparing to LSTM. This can reduce the prediction capability of DiffGRU. Both LSTM and Conv+LSTM work well and they have similar performance. The SGC+LSTM performs better than vanilla LSTM, which demonstrates the feature extraction by using spectral graph convolution is beneficial for traffic forecasting. However, the LSGC+LSTM does not outperform LSTM resulting from utilizing one-layer LSGC, whose parameters is not enough for representing the network features. The proposed TGC-LSTM, which capture graph-based features while accommodating the physical specialties of traffic networks, performs better

Model	LOOP			IRNIX		
	MAE (mph) ± STD	MAPE	RMSE	MAE (mph) ± STD	MAPE	RMSE
ARIMA	6.10 ± 1.09	13.85%	10.65	4.80 ± 0.32	13.51%	10.85
SVR	6.85 ± 1.17	14.39%	11.12	4.78 ± 0.37	13.37%	10.44
FNN	4.45 ± 0.81	10.19%	7.83	2.31 ± 0.17	8.35%	5.92
LSTM	2.70 ± 0.18	6.83%	4.97	1.14 ± 0.09	3.88%	2.43
DiffGRU	4.64 ± 0.32	11.18%	8.22	2.44 ± 0.09	8.91%	6.34
Conv+LSTM	2.71 ± 0.12	6.79%	5.02	1.13 ± 0.08	3.80%	2.37
LSGC+LSTM	3.16 ± 0.23	7.51%	6.18	1.38 ± 0.12	4.54%	2.82
SGC+LSTM	2.64 ± 0.12	6.52%	4.8	1.07 ± 0.08	3.74%	2.28
TGC-LSTM	**2.57 ± 0.10**	**6.01%**	**4.63**	**1.02 ± 0.07**	**3.28%**	**2.18**

Table 4.2: Performance comparison of different approaches. (The number of hops k is set as 3 in the graph convolution related model)

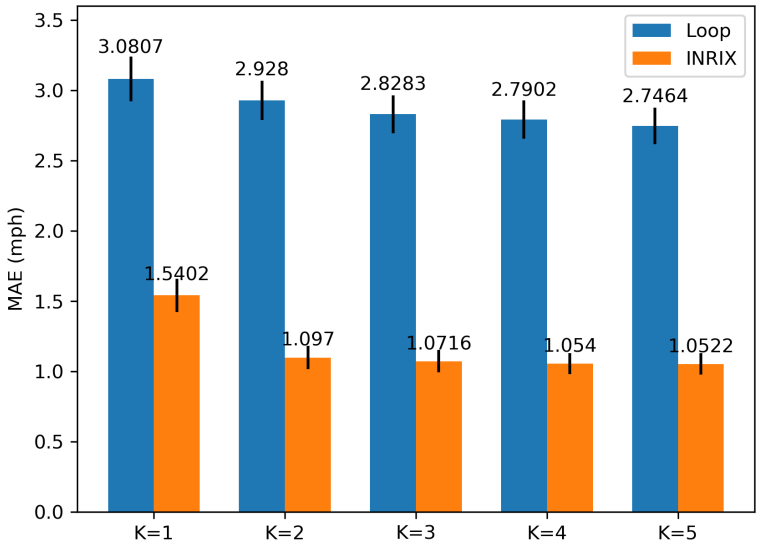


Figure 4.3: Histogram of performance comparison for the influence of orders (hops) of graph convolution in the TGC LSTM on INRIX and LOOP datasets.

than all other approaches. It should be noted that, for the INRIX data, during the nighttime or off-peak hours when there are no observed speed values on specific roads, the missing speed values are comprehensively imputed by the data provider. Thus, there are few variations at the non-peak hours in the INRIX data. Further, the speed values in the INRIX data are all integers. Therefore, the calculated errors of the INRIX data is less than that of the LOOP data and the evaluated performance on INRIX data is inflated somewhat.

Figure 4.3 shows a histogram of performance comparison on the effects of orders (hops) of the graph convolution in the TGC-LSTM. The model performance is improved when the value of K increases. For the LOOP data, the performance improves slightly when K is gradually increased. But for the INRIX data, there is a big improvement in when K increases to two from one. The complex structure and the various road types in the INRIX traffic network could be the main reason for this performing difference. Further, when K is larger than two, the improvement of the prediction is quite limited. This is also the reason why we choose K=3 in the model comparison part, as shown in TABLE 4.2.

4.7.1 *Training Efficiency*

In this subsection, we compare the training efficiency of the proposed model and other LSTM-based models. Figure 4.4a shows the validation loss curves versus the training epoch. Due to the early stopping mechanism is used in the training process, the numbers of training epochs are different. The TGC-LSTM needs less epochs to converge than the SGC+LSTM and the LSGC+LSTM. In addition, the loss of the TGC-LSTM decreases fastest among the compared models. Figure 4.4b shows the comparison of the training time per epoch of different models. The training cost of Conv+LSTM is between that of LSTM and SGC+LSTM. TGC-LSTM costs twice as much as LSTM does. The time required for SGC+LSTM is less than that for TGC-LSTM, while LSGC+LSTM costs slightly more than TGC-LSTM. Figure 4.4c shows the training losses of TGC-LSTM with different hops of graph convolution components. The rate of convergence increases when increasing the number of hops, k. In our experiments, when k is larger than 3, the training and validation results improve only marginally for both INRIX and LOOP datasets.

The model's loss function can add regularization terms to avoid overfitting. The proposed L1-norm on the graph convolution weights and L2-norm on the graph convolutional features can further help the model to confine the learned weights and features. However, there is a trade-off between the prediction accuracy and the scale of the penalty terms (λ_1 and λ_2). As tested, by adding the regularization terms to the loss function with the penalty rates setting as 0.01, the MAEs of the proposed model tested on the two datasets increase around 0.02, which are still superior to baseline models. Meanwhile, the TGC weight sparsity is increased and the value of the feature regularization $R^{\{2\}}$ is lower than that of the proposed model without regularization terms in the loss function, which means the TGC features' consistency is enhanced. Thus, it is worth adding these regularization terms to the loss function to help the trained model to be more interpretable. Figure 4.5 5 (a) and (b) show portions of the averaged graph convolution weight matrices for the INRIX data and the LOOP data, respectively, where $K = 3$ and the average weight is calculated by $\frac{1}{K}\sum_{i=1}^{K} W_i \odot \tilde{A}^i \odot \mathcal{F}\mathcal{F}R$. The road segment names, which are not displayed, are aligned on the vertical and horizontal axes with the same order in each figure. The colored dots in the matrices in Figure 4.5 (a) illustrate the weight of the contribution of a single node to its neighboring nodes. Since we align the traffic states of roadway segments based on their

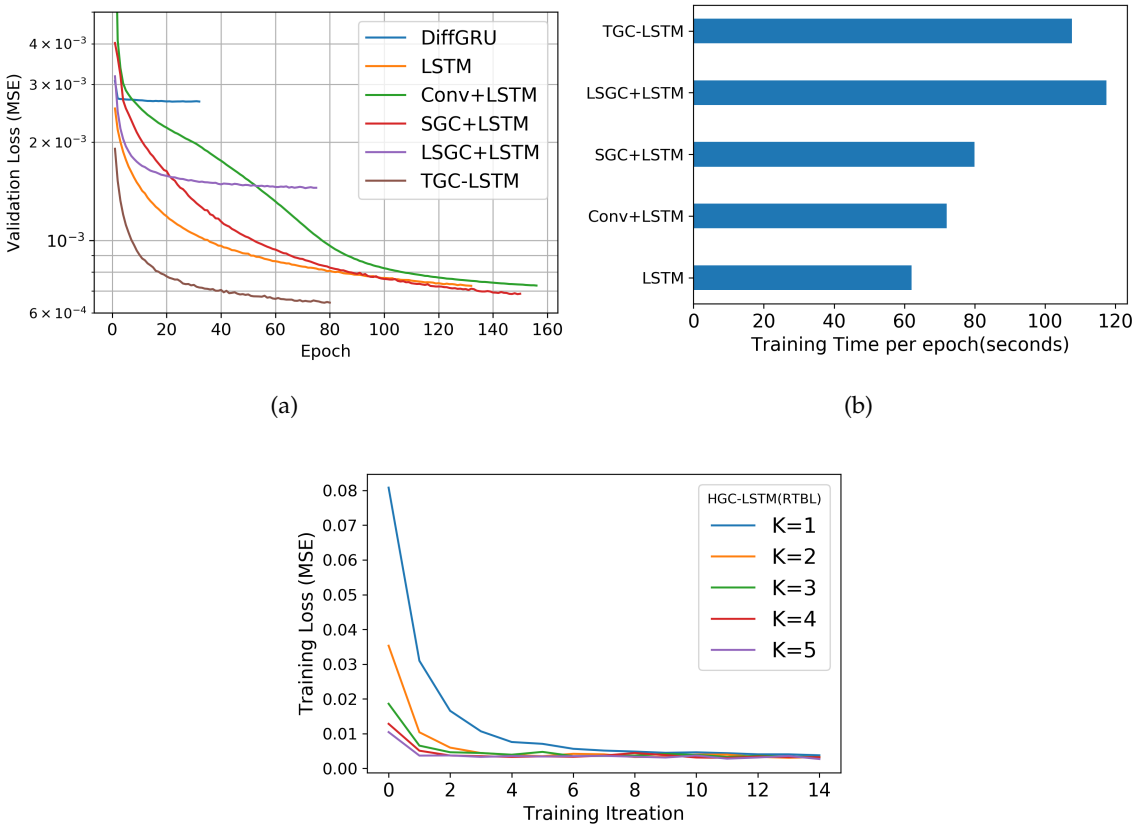
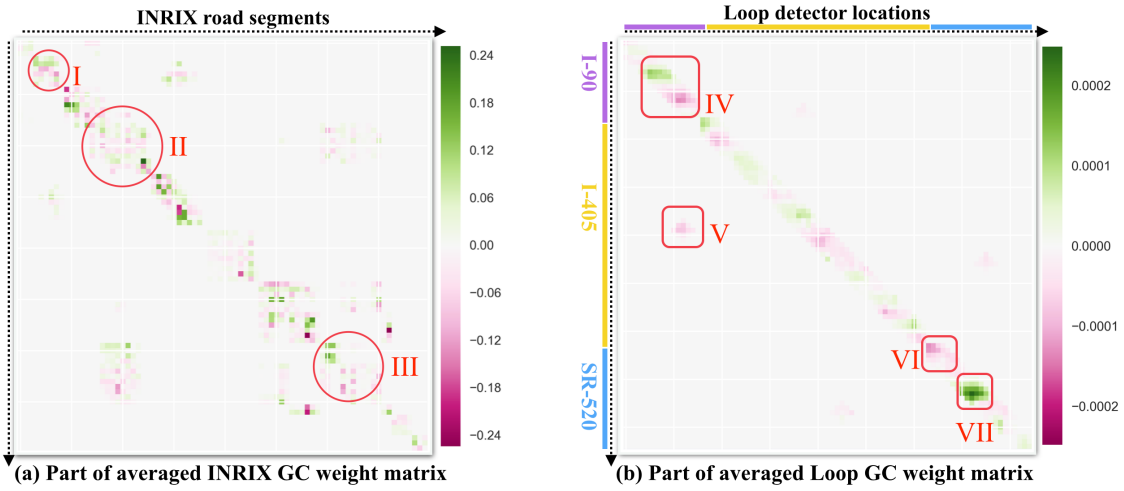


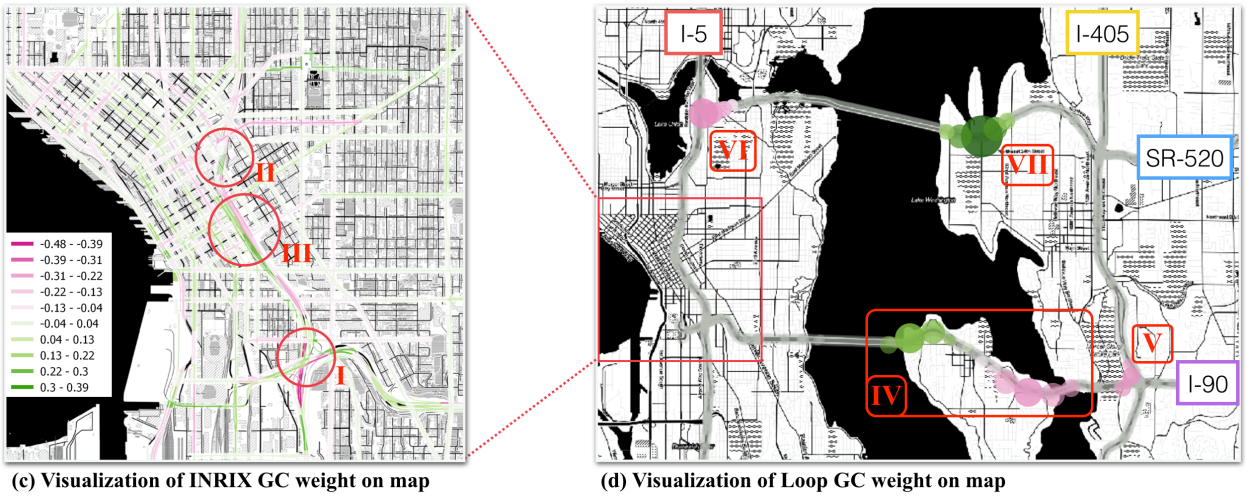
Figure 4.4: Validation loss versus training epoch (batch size = 40 and early stopping patience = 10 epochs). (b) Histogram of model's training time per epochs. (c) Compare training efficiency with different K hops of TGC, i.e. training loss versus training iteration when batch size = 40. (The figures are generated based on the LOOP data)

interconnectedness in the training data, most of the weights are distributed around the diagonal line of the weight matrix. The INRIX network is more complex and the average degree of nodes in the INRIX graph is higher than that in the LOOP graph. Hence, the dots in the average weight matrix of the INRIX graph convolution are more scattered. But these dots still form multiple clusters demonstrating the weights of several nearby or connected road segments. Considering roadway segments are influenced by their neighboring or nearby connected segments, the nodes with the large absolute weight in a cluster are very likely to be key road segments in the local traffic network. In this way, we can infer the bottlenecks of the traffic network from the traffic graph convolution weight matrices.

4.7.2 *Model Interpretation and Visualization*



(a)



(b)

Figure 4.5: (a) Visualization of a proportion of the INRIX GC weight matrix, in which three representative weight areas are tagged. (Visualization of a proportion of the LOOP GC weight matrix, in which four representative weight areas are tagged. (c) Visualization of the INRIX graph convolution weight on the real traffic network using colored lines. (d) Visualization of the four tagged weight areas in the LOOP graph convolution weight on the Seattle freeway network using colorful circles.

To better understand the contribution of the graph convolution weight, we mark seven groups of representative weights in Figure 4.5 (a) and (b) and visualize their physical locations on the real

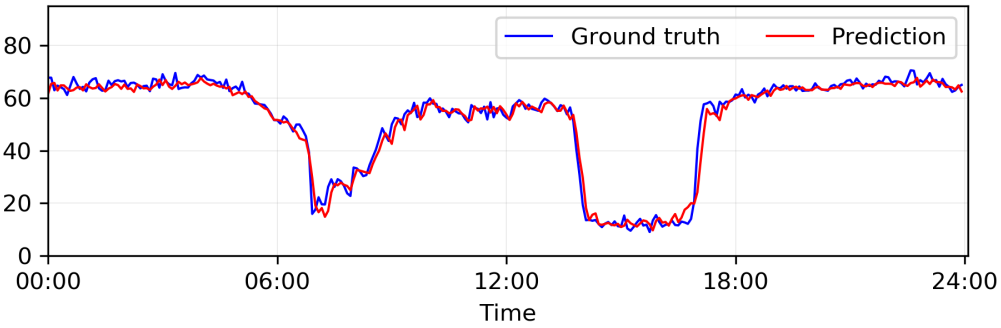
map in 4.5 (c) and (d), by highlighting them with Roman numerals and red boxes. The influence of these marked weights on neighboring nodes in the INRIX and LOOP data are visualized by lines and circles, respectively, considering the INRIX traffic network is too dense to use circles. The darkness of the green and pink colors and the sizes of the circles represent the magnitude of influence. It should be noted that the darkness of colors on lines on the INRIX map and the size of the circles on the LOOP map will change when the model is trained with different scales of regularization terms (λ_1 and λ_2).

From 4.5 (c), we can find the marked areas with dark colors in the INRIX GC weight matrix, (I), (II), and (III), are all located at very busy and congested freeway entrance and exit ramps in Seattle downtown area. In 4.5 (d), the area tagged with (IV) is quite representative because the two groups of circles are located at the intersections between freeways and two main corridors that represent the entrances to an island (Mercer Island). Areas (V) and (VI) are the intersections between I-90 and I-405 and between I-5 and SR-520, respectively. The VII area located on SR-520 contains a frequent-congested ramp connecting to the city of Bellevue, the location of which is highlighted by the biggest green circle. Additionally, there are many other representative areas in the graph convolution weight matrix, but we cannot show all of them due to space limits. By comparing the weight matrix with the physical realities of the traffic network, it can be shown that the proposed method effectively captures spatial dependencies and helps to identify the most influential points/segments in the traffic network.

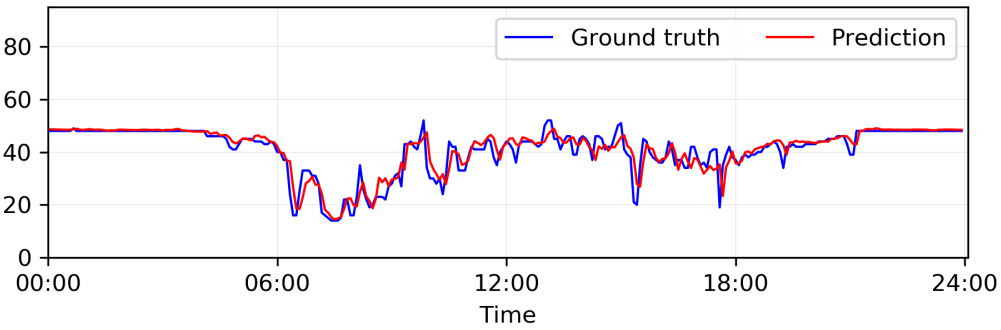
Figure 4.6 visualizes the predicted traffic speed sequences and the ground truth of two locations selected from the LOOP and INRIX dataset. Though the traffic networks of the two datasets are very different, the curves demonstrate that the trends of the traffic speed are predicted well at both peak traffic and off-peak hours.

4.8 CHAPTER SUMMARY

In this chapter, we learn the traffic network as a graph and define a traffic graph convolution operation to capture spatial features from the traffic network. The traffic graph convolution incorporates the adjacency matrix and the proposed free-flow reachable matrix to extract localized features from the graph. We propose a traffic graph convolutional LSTM neural network to forecast



(a) Sensor ID: do05es16756 in LOOP dataset on 2015-01-04



(b) Sensor ID: 114P13774 in INRIX dataset on 2012-01-03.

Figure 4.6: Traffic time series forecasting visualization for LOOP and INRIX datasets on two randomly selected days.

network-wide traffic states. We also design two regularization terms on the TGC weights and TGC features, respectively, that can be added to the model's loss function to help the learned TGC weight to be more stable and interpretable. By evaluating on two real-world traffic datasets, our approach is proved to be superior to the compared baseline models. In addition, the learned TGC weight can help to identify the most influential roadways, and thus, enhance the interpretability of the proposed model.

For future work, we will move forward to improve the model's prediction performance in terms of accuracy and robustness, and further investigate how to conduct the convolution on both spatial and temporal dimensions to make the neural network more interpretable. [3]

3 This chapter is a slightly modified version of "**Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting**" published in *IEEE Transactions on Intelligent Transportation Systems* and has been reproduced here with permission from IEEE. (doi: https://doi.org/10.1109/TITS.2019.2950416)

5

LEARNING TRAFFIC AS A GRAPH: A GATED GRAPH WAVELET RECURRENT NEURAL NETWORK FOR NETWORK-SCALE TRAFFIC PREDICTION

5.1 OVERVIEW

Traffic pattern recognition is critical for modern intelligent transportation systems (ITS) and the planning for smart cities. A comprehensive recognition of historical urban traffic patterns is not only capable of identifying the recurrent congestions and bottlenecks of urban traffic networks, but also able to largely enhance the forecasting of future traffic states. As a component of ITS, accurate network-wide traffic prediction is the prerequisite for dynamic route planning and traffic assignment optimization. The growing need for short-term prediction of traffic parameters embedded in ITS has led to a great deal of research on traffic forecasting in the last three decades (Vlahogianni et al., 2004). Before the rise of artificial intelligence, traffic forecasting methods have been gradually shifting from traditional statistical models to computational intelligence, or say machine learning methods (Vlahogianni et al., 2014). With the exponentially increase in the volume of traffic data and the computational capability, a large amount of deep learning models, including recurrent neural network (RNN) (Ma et al., 2015), convolutional neural network (CNN) (Ma et al., 2017), stacked autoencoder (Lv et al., 2015), and their combinations were adopted for short-term traffic forecasting in recent years.

5.1.1 *Challenges in Network-wide Traffic Forecasting*

Much previous research focusing on traffic prediction only studied a roadway segment or several consecutive segments on a corridor. It is proved that using the information of multiple sensors/locations can help prediction models track short-term trends and enhance prediction performance (Li et al., 2015). Thus, in order to enhance traffic prediction performance and bring the power of artificial intelligence into real applications in the transportation field, it is inevitable to take large-scale traffic networks as the study areas. Due to the complex structure of traffic networks, there are several obvious hurdles in the traffic forecasting process.

The first question is how to represent the complex structure of a roadway network accurately? An example of a complex traffic network is shown in Figure 5.1 (a). Previous studies attempted to convert the traffic states of sensing locations in a roadway network into a 2D spatial-temporal matrix (Ma et al., 2017) or convert the geometrical structure of urban roadway networks as colored images (Yu et al., 2017a) for learning traffic states' features, where an example is shown in Figure 5.1 (b). However, in these ways, the topology of a roadway network cannot be adequately represented and the relationship of the adjacent roadway segments can hardly be comprehensively learned. To address this issue, many studies (Cui et al., 2019; Li et al., 2018; Yu et al., 2018; Zhang et al., 2018) have proposed a more elegant way to consider the traffic network as a graph consisting of vertices and edges denoting roadway segments and intersections, respectively, as shown in Figure 5.1 (c). In this way, the topology of a traffic network can be comprehensively represented by a graph.

Second, given considering the traffic network as a graph, designing an effective feature extraction process for the non-linear structured data is also challenging. Some powerful mathematical tools, such as graph convolution operator (Estrach et al., 2014; Henaff et al., 2015), have been adopted to integrate graph features into the traffic forecasting problems (Cui et al., 2019; Li et al., 2018). However, the original form of graph convolution is not well-localized, which means that, with respect to a centered node, the graph convolution cannot learn features exclusively from its neighboring nodes within a specific scale. Although Defferrard et al. (2016) proposed a fast localized spectral filter to enable the localization of the graph convolution, the receptive field

is still not flexible. The neighboring field of a centered node is strictly confined by a ball of a designated radius covering a corresponding amount of hops of neighboring nodes.

Thus, the third issue is the lack of methods to enable the flexible local feature extraction process in a traffic prediction model. As shown in (Li et al., 2015), traffic time series have long-term and short-term trends. Generally, the traffic state of a road is mainly affected by its neighboring roads. However, some key roadway segments, such as traffic bottlenecks or the ones with critical incidents, can highly affect the operational performance of the entire traffic network. Thus, much attention should be discriminatively paid to these segments in the feature extraction process. The classical wavelet transform inherently with the localization property can capture the sudden changes and detect peaks in a signal. Considering the spectral graph convolution is defined based on Fourier transform (Shuman et al., 2013), analogously, wavelet transform can be extended to the spectral domain as the graph wavelet (Hammond et al., 2011) to overcome the localization problem in the graph convolution. Graph wavelets are localized in the vertex domain, reflecting the information diffusion centered at each node. Based on the graph wavelet theory (Hammond et al., 2011), Xu et al. first proposed the graph wavelet neural network to solve semi-supervised classification problems. In the traffic modeling process, the graph wavelet can also be used to flexibly extract comprehensive features and automatically concentrate more on critical segments in the traffic network. Thus, we adopt the graph wavelet as a component of the proposed neural network in this study.

In addition, a traffic network related dataset not only contains geospatial information but also forms as temporal sequences. RNN and its variants, like long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014) neural networks, have been proved to be the superior methods to deal with sequence data in multiple fields. Since the traffic prediction is based on a long stream of historical data, the vanishing gradient problem often occurs in the vanilla RNN. To overcome this problem, a gated structure should be adopted in the prediction model.

5.1.2 *Contribution and Organization of the Chapter*

To address these issues, we propose a new model for network-scale traffic learning and prediction. In this study, we learn the traffic network as a graph. To overcome the localization problem in graph convolution and make the receptive field more flexible, we adopt the graph wavelet as a key component of the proposed neural network. The graph wavelet is localized in the vertex domain. The graph wavelet operation considers both the traffic states of a road network and the underlying topology structure of the network. In order to learn the spatial-temporal features from the traffic network and fulfill traffic forecasting, we propose a graph wavelet gated recurrent neural network (GWGR). As tested on two real-world traffic datasets, the proposed GWGR model achieves better forecasting accuracy with fewer weight parameters comparing to existing benchmark models. The contributions of this study are summarized as follows:

1. We learn the traffic network as a graph and incorporate the graph wavelet as a key component to extract well-localized features from the traffic network based graph. Comparing to graph convolution, graph wavelet is pretty flexible with no need to specify the neighboring area in the topological graph structure for feature extraction. To the best of our knowledge, this is the first time that a graph wavelet based neural network is utilized for traffic forecasting.

2. We propose a graph wavelet gated recurrent neural network to learn from the spatial-temporal traffic network data, in which the graph wavelet operators act as filters in the gates of the recurrent neural network.

3. The proposed graph wavelet gated recurrent neural network can achieve superior prediction performance with fewer weight parameters and higher training efficiency, comparing to many existing models.

4. The quantifiable learned weights of the graph wavelets turn out to be sparse. This sparse property of learned weights can largely enhance the interpretability of the model and help to identify the key roadway links in the traffic network.

The rest of this chapter is organized as follows. Section 5.2 discusses the related work. Section 5.3 introduces the wavelet transform and graph wavelet transform as the building block of the

proposed method. Section 5.4 proposes the graph wavelet gate recurrent neural network in detail. Section 5.5 presents the experimental settings. Section 5.6 shows the performance and the analysis of several properties of the proposed model. In Section 5.7, we conclude this chapter and describe the future directions of learning comprehensive features for traffic prediction.

5.2 RELATED WORK

Previous traffic prediction methods can be categorized into two main groups, i.e. parametric approaches and nonparametric approaches (Lv et al., 2015). Parametric traffic prediction approaches are developed based on a predefined model structure with several certain theoretical assumptions, and parameters are calibrated using historical data (Smith et al., 2002). A variety of parametric traffic prediction approaches were proposed, including many variants of autoregressive integrated moving average (ARIMA) models (Williams, 2001), parametric Kalman filtering models (Okutani and Stephanedes, 1984), and other types of time-series models (Ghosh et al., 2009). In order to accommodate the stochastic and nonlinear nature of traffic flow, nonparametric approaches were also widely adopted for traffic prediction or traffic data imputation, including k-nearest neighbor (k-NN) methods (Chang et al., 2012), support vector regression (SVR) (Wu et al., 2004), Bayesian network approaches (Sun et al., 2006), and tensor decomposition approach (Chen et al., 2019b). In spite of classical traffic prediction models are well studied and applied, it is pretty difficult for these models to deal with huge amount of large-scale network-wide traffic data.

Traffic prediction is widely studied in recent years with the rise of artificial intelligence. Existing literature shows that deep neural network models achieve superior prediction performance comparing to classical traffic prediction models. Due to traffic prediction is based on historical sequence data, most of the existing deep learning models (Cui et al., 2017; Ma et al., 2015; Pu et al., 2019; Wang et al., 2019a) are built on RNN and its variants like LSTM and GRU. Because of the spatiotemporal characteristics of traffic data, a large amount of other types of deep learning models, including convolutional neural network (CNN) (Ma et al., 2017), stacked autoencoder (Lv et al., 2015)), generative adversarial network (GAN) (Liang et al., 2018), and capsule network (Ma et al., 2020), and multiple combinations (Cui et al., 2017; Ke et al., 2020; Liao et al., 2018; Lv et al., 2018; Wu et al., 2018) of these models were utilized to capture spatial features and estimate traffic

states. However, there is still much room for the aforementioned models to improve their feature extraction when taking the complicated nature of the traffic network structure into consideration.

Due to many real-world data are non-linear structured data, graph based deep learning models attracted much attention in recent years (Zhou and Li, 2017). Roadway network based traffic data as a representative graph structured data can be effectively analyzed by taking many advantages of graph based models. Graph embedding methods (Yao et al., 2018) and graph attention networks (Zhang et al., 2018) have been adopted as components of neural network structures to deal with spatiotemporal traffic prediction problems. Based on the spectral graph theory (Shuman et al., 2013), spatiotemporal graph convolution neural network (Yu et al., 2018), diffusion convolutional recurrent neural network (Li et al., 2018), graph convolutional neural network with data-driven graph filter (Lin et al., 2018), etc. are proposed for traffic forecasting. A traffic graph convolutional recurrent neural network (Cui et al., 2019) incorporating the physical properties of roadways is also proposed for improving forecasting accuracy and enhancing the interpretability of the model. However, the interpretation and flexible localization of the feature extraction process in these graph convolution based models are not well addressed and discussed. A recent work (Xu et al., 2019) proposed the graph wavelet neural network to implement efficient convolution on graph data to solve semi-supervised classification problems. In this study, inspired by the graph wavelet neural network, we attempt to address those issues by adopting the graph wavelet as a component of the proposed gated recurrent model.

5.3 PRELIMINARIES AND NOTIONS

The spatial-temporal traffic data collected by sensors are normally denoted as $X = [x_0,\ x_1, \ldots, x_t, \ldots, x_{T-1}] \in \mathbb{R}^{T\times N}$, in which T is the length of time steps and N is the number of traffic sensing locations. Each element x_t^n in X denotes the traffic status at t-th observation on the n-th location. Due to most of traffic sensing locations are connected by bidirectional roadways, the traffic network can be considered as an undirected graph consisting of vertices and edges representing sensing locations and connecting links, respectively. The graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$ consisting of N vertices or nodes $v_j \in \mathcal{V}$ and their linking edges $(v_i, v_j) \in \mathcal{E}$. For clearance, graph nodes refer to vertices and they might be used interchangeably in this chapter. The adjacency matrix

$A \in \mathbb{R}^{N\times N}$ is used to describe the connectedness of vertices, in which element $A_{i,j} = A_{j,i} = 1$ if vertices i and j are connected, otherwise $A_{i,j} = 0$ ($A_{i,i} = 0$). Then a diagonal graph degree matrix $D \in \mathbb{R}^{N\times N}$ describing how many edges are attached to each vertex can be obtained by $D_{i,i} = \sum_{j=1}^{N} A_{i,j}$. The connectivity of the graph vertices can also be encoded in the graph Laplacian matrix $\mathcal{L}$, which is essential for spectral graph analysis. The combinatorial definition is $\mathcal{L} = D - A$ and the normalized Laplacian matrix is defined as $\mathcal{L} = I_N - D^{-1/2}AD^{-1/2}$, where $I_N \in \mathbb{R}^{N\times N}$ is the identity matrix. Due to $\mathcal{L}$ is a symmetric positive semidefinite matrix, it can be diagonlaized as $\mathcal{L} = U\Lambda U^T$ by eigenvector matrix $U = [u_0,\ u_1, \ldots, u_{N-1}] \in \mathbb{R}^{N\times N}$ and its corresponding diagonal eigenvalue matrix $\Lambda = \text{diag}(\lambda_0, \lambda_1,\ \ldots,\ \lambda_{N-1}) \in \mathbb{R}^{N\times N}$ satisfying $\mathcal{L}u_i = \lambda_i u_i$.

5.3.1 *Graph Fourier Transform and Graph Convolution*

The Fourier transform of a signal in the temporal domain can be regarded as the linear combination over a set of Fourier basis with different frequencies in the frequency domain. The larger the weight of a Fourier basis is, the more contribution the specific basis makes. Similarly, a graph signal in the vertex domain can also be transformed into a combination of a set of base in the frequency/spectral domain. The spectrum of graph Laplacian $\mathcal{L}$ carries a notion of frequency represented by the eigenvectors that the eigenvectors associated with higher eigenvalues generally have more zero crossings (Shuman et al., 2013). Taking the eigenvectors in U as the Fourier basis, the graph Fourier transform on graph signal vector $x \in \mathbb{R}^N$ can be defined as $\hat{x} = U^T x$ and its inverse as $x = U\hat{x}$, where $\hat{x}$ is the graph signal in the frequency domain. According to the convolution theorem (Bracewell and Bracewell, 1986), the Fourier transform of a convolution is the element-wise product of Fourier transform, and thus, the convolution operator $\otimes_{\mathcal{G}}$ on graph $\mathcal{G}$ can be generalized as:

$$f \otimes_{\mathcal{G}} x = U\left(\hat{f} \bigodot \hat{x}\right) = U\left(\left(U^T f\right) \bigodot \left(U^T x\right)\right) \tag{5.1}$$

where f is a filter function that can be considered as the convolutional kernel and $\bigodot$ is the element-wise multiplication operator. According to matrix multiplication rule, the $U^T f$ can be replaced by a diagonal matrix $\hat{f}(\Lambda_\theta)$ and Equation 5.1 can be represented by

$$f \otimes_{\mathcal{G}} x = U\hat{f}\left(\Lambda_\theta\right) U^T x \tag{5.2}$$

where $\hat{f}$ is the filter function in the frequency domain and Λ_θ is a diagonal parameter matrix. When applying to deep learning models, the graph convolution of x is normally defined as $U\Lambda_\theta U^T x$, and here, Λ_θ is the diagonal weight matrix that should be learnt during the training process.

The spectral graph convolution operator has been adopted in many neural network structures (Estrach et al., 2014; Henaff et al., 2015). In those graph convolution layers, the diagonal matrix Λ_θ acts as a learnable filter function and its diagonal elements are all learnable weight parameters. However, due to the graph convolution operation is defined based on Fourier basis U, the convolution result on one vertex of a graph signal comes from all vertices, i.e. the receptive field of the graph convolution operator covers the whole graph structure. Thus, the graph convolution is not well-localized in the vertex domain. To overcome this limitation, a polynomial filter is proposed (Defferrard et al., 2016) to conduct graph convolution on one vertex from the signals of its exactly k hops of neighboring vertices, functioning as a localized receptive field in conventional CNNs. The filter is defined as

$$g_\theta = \sum_{k=0}^{K-1} \theta_k \Lambda^k \tag{5.3}$$

where $\theta \in \mathbb{R}^K$ is the learnable polynomial coefficients and Λ^k is the k-order Laplacian diagonal eigenvalue matrix. In this case, the graph convolution on the Laplacian is K-localized. To apply the graph convolution in the transportation network learning and take the roadway physical properties into consideration, a free flow reachability matrix is incorporated in the traffic graph convolution (Cui et al., 2019).

However, in those existing methods, the hyperparameter K is still required to be specified to decide the size of graph convolution receptive field, i.e. the hops of neighborhood nodes that influence a centering node. Further, the receptive field confined by K is not flexible enough considering that key vertices in a graph are normally not evenly distributed.

5.3.2 *Wavelet Transform*

Fourier transform can decompose a signal in the temporal domain into a combination of frequencies, while wavelet transform can indicate when and where the frequencies contribute more, if a

signal varies with time. The wavelet transform adopts a wavelet prototype function, i.e. mother wavelet, to cut up signal data into different frequency components and studies each component with a resolution matched to its scale (Bruna et al., 2013). A classical wavelet $\psi_{s,a}$ of a function at scale s and location a is constructed from the translated and scaled mother wavelet ψ:

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right) \tag{5.4}$$

And the wavelet coefficients $W_f(s,a)$ is acquired by convolving the input $f(x)$ with wavelet:

$$W_f(s,a) = \int_{-\infty}^{\infty} \frac{1}{s}\,\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx \tag{5.5}$$

in which $(\cdot)^*$ denotes complex conjugate and ψ^* equals to ψ for real-valued and even the mother wavelet. Then the classical wavelet transform operator T^s for a scale s at location a can be denoted by $T^s f(a) = W_f(s,a)$. Letting $\overline{\psi}_s(x) = \frac{1}{s}\psi^*\left(\frac{-x}{s}\right)$, the operator T^s can be written as:

$$T^s f(a) = \int_{-\infty}^{\infty} \frac{1}{s}\,\psi^*\left(\frac{x-a}{s}\right) f(x)\,dx = \int_{-\infty}^{\infty} \frac{1}{s}\,\overline{\psi}_s(a-x)\,f(x)\,dx = \left(\overline{\psi}_s \otimes f\right)(a) \tag{5.6}$$

where $\otimes$ is the convolution theorem. Taking the Fourier transform and applying the convolution theorem, we can get:

$$\widehat{T^s f}(\omega) = \widehat{\overline{\psi}}_s(\omega)\,\hat{f}(\omega) \tag{5.7}$$

where ω is a frequency component after the Fourier transform. Based on the scaling properties of the Fourier transform, $\widehat{\overline{\psi}}_s(\omega) = \widehat{\psi}^*(s\omega)$. Then by inverting the Fourier transform we can the classical wavelet transform as:

$$\left(T^s f\right)(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\omega x}\,\widehat{\psi}^*(s\omega)\,\hat{f}(\omega)\,d\omega \tag{5.8}$$

By applying an impulse function, $\delta_a(x) = \delta(x-a)$, on the wavelet operator, we can localize the real-valued wavelet as

$$\left(T^s \delta_a\right)(x) = \frac{1}{s}\psi^*\left(\frac{a-x}{s}\right) = \psi_{s,a}(x) \tag{5.9}$$

The fundamental idea of wavelet transform is to analyze the data/signal according to a scale s at whatever location a. A vivid interpretation of wavelet analysis can be described as following: the mother wavelet acts as a window. If we look at a signal with a large "window", we would notice the gross feature, while if we take a small "window", we would notice small features. The wavelet transform can help to see both the forest (gross features) and the trees (small features) (Bruna et al., 2013).

5.3.3 *Graph Wavelet Transform*

Similar to graph convolution transform, graph wavelet transform also converts graph signal from vertex domain to spectral domain. The graph wavelet transform conducted on the graph Laplacian matrix is represented by a wavelet operator defined as $T_g^s = g(s\mathcal{L})$ with a kernel g at scale s, where the kernel g acts the mother wavelet in classical wavelet transform. Although the vertex domain of the graph is discrete, this kernel g is defined in the continuous domain, and thus, the scaling parameter s can be assigned as any positive real value.

The wavelet operator acts on a given function f in the spectral domain is fulfilled by modulating each Fourier mode of f as

$$\widehat{T_g^s f}\,(i) = g\,(s\lambda_i)\,\hat{f}\,(i) \tag{5.10}$$

where λ_i is the i-th eigenvalue of $\mathcal{L}$. Applying inverse Fourier transform, the wavelet operator acts on f in the vertex domain can be defined as

$$\left(T_g^s f\right)(m) = \sum_{i=0}^{N-1} g\,(s\lambda_i)\,\hat{f}\,(i)\,u_i \tag{5.11}$$

where $m \in \{0, \ldots,\ N-1\}$ is an index of the m-th element/vertex in the graph. Equally, the spectral graph wavelet transform on a single vertex n can be calculated by applying the wavelet operator to a delta impulse function δ_n as

$$\psi_{s,n} = T_g^s \delta_n \tag{5.12}$$

where $\psi_{s,n} \in \mathbb{R}^N$. Similar to the graph wavelet transform performed on vertex n from any vertex m is obtained as:

$$\psi_{s,n}\,(m) = \sum_{i=0}^{N-1} g\,(s\lambda_i) u_i^*\,(n)\,u_i\,(m) \tag{5.13}$$

where u_i^* should be the conjugate eigenvector. Mathematically, based on Equation 5.13, the graph wavelet coefficients of all the vertices can be written as

$$\Psi_s = U G_s U^T \tag{5.14}$$

where U is the matrix formed by Laplacian eigenvectors and $G_s = \text{diag}\,(g\,(s\lambda_0)\,,\ \ldots, g\,(s\lambda_{N-1}))$ is a diagonal kernel matrix. The graph wavelet set $\Psi_s = (\psi_{s,0}, \ldots,\ \psi_{s,N-1}) \in \mathbb{R}^{N\times N}$ can be considered

as the graph wavelet basis. Hence, according to (Xu et al., 2019), the graph wavelet transform of a graph signal x is defined as $\hat{x} = \Psi_s^{-1} x$ and the inverse graph wavelet transform is defined as $x = \Psi_s \hat{x}$.

As stated by (Pérez-Rendón and Robles, 2004), the convolution theorem holds for continuous admissible wavelet transform that $\widehat{(\Psi_1 \otimes \Psi_2)}(\omega) = \widehat{\Psi_1}(\omega)\widehat{\Psi_2}(\omega)$, where Ψ_1 and Ψ_2 are two wavelet functions, $\otimes$ is the convolution operator, and ω is the signal in Fourier domain. Analogously, the graph convolution based on graph wavelet transform is defined as

$$f \otimes_{\mathcal{G}} x = \Psi_s \left(\left(\Psi_s{}^{-1} f \right) \odot \left(\Psi_s{}^{-1} x \right) \right) \tag{5.15}$$

which is similar to the graph Fourier transform based graph convolution. Compare to graph Fourier transform, the graph wavelet transform based graph convolution has multiple good properties:

1. Since the structure of traffic network usually does not change, the graph wavelet matrix Ψ_s can be calculated in advance to make the training and testing process more efficient. The Chebyshev polynomial approximation of Ψ_s can make the calculation even more efficient (Xu et al., 2019).

2. The graph wavelet is well-localized in vertex domain, due to each wavelet coefficient measures the significance of the signals on a centered vertex to neighboring vertices with respect to a certain scale in the graph.

3. The wavelet coefficient matrix Ψ_s and $\Psi_s{}^{-1}$ are normally highly sparse, especially for the graphs extracted from real-world traffic networks. Then, the computation can be more efficient when the sparse matrix operation is incorporated in the neural network.

5.4 GRAPH WAVELET GATED RECURRENT NEURAL NETWORK

To extract the spatial-temporal features on each vertex of the graph constructed from a roadway network, we propose a graph wavelet gated recurrent (GWGR) neural network. Similar to LSTM, the GWGR has several gate units to filter out or add information to the cell state. The difference is that the gate units in GWGR are defined based on a graph wavelet matrix. The framework of

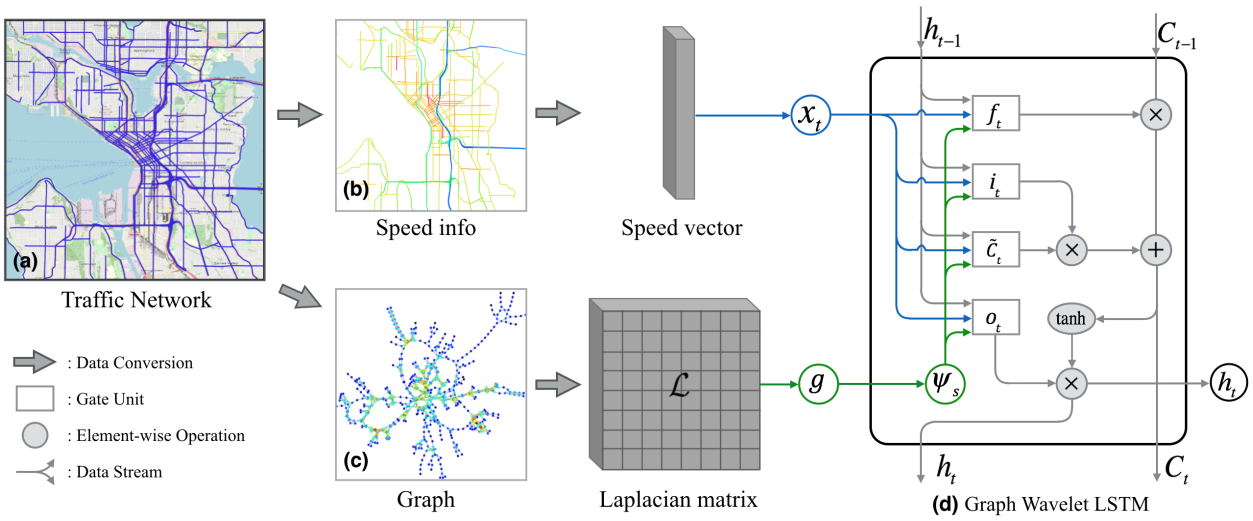


Figure 5.1: Demonstration of model framework. (a) Urban traffic network in downtown Seattle. (b) Speed information of roadway segments illustrated by various colors. (c) Graph structure converted from the traffic network. (d) Structure of a graph wavelet LSTM unit at time t, in which g is the kernel function and Ψ_s is the graph wavelet matrix.

the proposed model is shown in Figure 5.1 (d). At time t, the network-wide speed information shown in Figure 5.1 (b) is converted into a vector x_t as the input of the model. The traffic network structure is converted into a graph, as shown in Figure 5.1 (c). The graph wavelet is fixed for all time steps and it is designed based the Laplacian matrix and a kernel function.

In the proposed model, the graph wavelet coefficient matrix Ψ_s is defined in Equation 5.14. We adopt the heat kernel $g(s\lambda_i) = e^{-s\lambda_i}$ and the Ψ_s^{-1} is easily obtained by replacing $g(s\lambda_i)$ with $g(-s\lambda_i)$. Then the GWGR is defined by the following equations:

$$f_t = \sigma_g\left(\Psi_s \Lambda_f^x \Psi_s^{-1} x_{t-1} + \Psi_s \Lambda_f^h \Psi_s^{-1} h_{t-1} + b_f\right) \tag{5.16}$$

$$i_t = \sigma_g\left(\Psi_s \Lambda_i^x \Psi_s^{-1} x_{t-1} + \Psi_s \Lambda_i^h \Psi_s^{-1} h_{t-1} + b_i\right) \tag{5.17}$$

$$o_t = \sigma_g\left(\Psi_s \Lambda_o^x \Psi_s^{-1} x_{t-1} + \Psi_s \Lambda_o^h \Psi_s^{-1} h_{t-1} + b_o\right) \tag{5.18}$$

$$\tilde{C}_t = \tanh\left(\Psi_s \Lambda_C^x \Psi_s^{-1} x_{t-1} + \Psi_s \Lambda_C^h \Psi_s^{-1} h_{t-1} + b_C\right) \tag{5.19}$$

where f_t, i_t, o_t and $\tilde{C}_t \in \mathbb{R}^N$ are the outputs of the forget gate, input gate, output gate and the input memory cell. Λ_f^x, Λ_i^x, Λ_o^x, and $\Lambda_C^x \in \mathbb{R}^{N\times N}$ are diagonal weight matrices that filter the input x_t to the three gates and the memory cell with the help of graph wavelet matrix. Similarly Λ_f^h, Λ_i^h, Λ_o^h, and $\Lambda_C^h \in \mathbb{R}^{N\times N}$ are also diagonal weight matrices for the preceding hidden state h_t. b_f, b_i, b_o, and $b_C \in \mathbb{R}^N$ are four bias weight vectors. In this way, we call those matrices with the form like $\Psi_s \Lambda \Psi_s^{-1}$ as graph wavelet weight matrices in GWGR. The σ_g is the sigmoid activation function and tanh is the hyperbolic tangent function. Then the cell state C_t and the hidden state h_t at time t are calculated as follows

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{5.20}$$

$$h_t = o_t \odot \tanh(C_t) \tag{5.21}$$

The $h_t \in \mathbb{R}^N$ is also the output of the GWGR unit at time t. Given the input sequence $X = [x_0, x_1, \ldots, x_{T-1}] \in \mathbb{R}^{T\times N}$, the predicted value of the future step is $\hat{x}_T = h_T$. If we only need to predict traffic data for one future step, the loss function of the model can be defined as

$$Loss = Loss(\hat{x}_T - x_T) = Loss(h_T - x_T) \tag{5.22}$$

where Loss $(\cdot)$ is the loss function, normally adopting the mean square error function of the traffic prediction problem.

5.5 EXPERIMENTAL SETTING

5.5.1 *Dataset Description*

The proposed model is tested on two real-world datasets covering a freeway network and a more complicated urban roadway network, respectively.

Freeway Traffic Dataset (Cui et al., 2017): The data in this dataset is collected by inductive loop detectors deployed on the Seattle freeway system covering four connected freeways in the Great Seattle areas, including I-5, I-90, I-405, and SR-520, as shown in Figure 5.2 (b). The

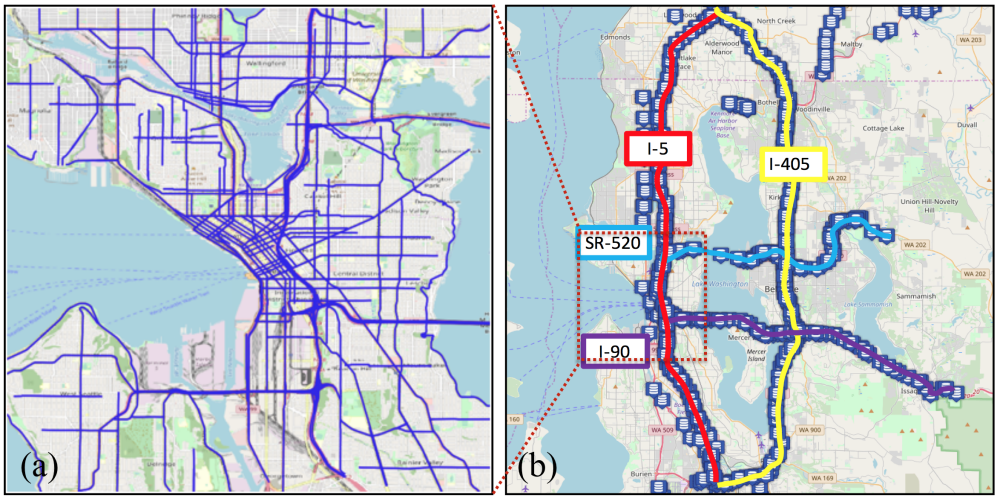


Figure 5.2: (a) Urban traffic dataset covering the downtown Seattle urban corridors. (b) Freeway traffic data covering freeway system in Seattle area.

raw data contains three basic traffic flow characteristics, including traffic speed, volume, and density. After the dataset was comprehensively checked and cleaned (Wang et al., 2016), only the high-quality speed information in 2015 is used in the experiment. The traffic network contains 323 traffic sensing locations, i.e. $N = 323$. The time interval is 5-minute. The speed data is well-formatted and there is no missing value. This dataset is also published via an accessible link: https://github.com/zhiyongc/Seattle-Loop-Data.

Urban Traffic Dataset: This dataset originated from the National Performance Management Research Data Set (NPMRDS) data (FHWA, 2019). This dataset contains the speed data of roadway links in the Seattle downtown area, which is mostly collected by probe vehicles. In this area, the road network is very complex that it contains principal arterials, minor arterials, one-way streets, freeways, ramps, express lanes, etc. This dataset covers the year of 2012 and the time interval is also 5-minute. The roadway network contains more than 1000 roadway links, but we select the largest connected roadway network containing 745 segments in the experiment, i.e. N= 745, as shown in Figure 5.2 (a). For confidentiality reasons, this dataset is not allowed to be published at this stage.

It should be noted that the speed values of each dataset are normalized to [0,1] in the training and testing process using the following equation:

$$X = \frac{X - min\,(X)}{\max\,(X)\ - min\,(X)} \tag{5.23}$$

Due to this study mainly focuses on developing a new neural network structure to extract spatiotemporal features and forecast network-scale traffic status, some other traffic-related datasets, like weather data and incident data, are not incorporated in this study.

5.5.2 *Baseline Models and Metrics*

Baseline Models: We compare the proposed model with the following baseline models:

1. ARIMA (Hamed et al., 1995): the auto-regressive integrated moving average model. The model is tested by using the StatsModels 0.9.0 package (Seabold and Perktold, 2010) and the model parameters are set as default;

2. SVR (Wu et al., 2004): the support vector regression with the radial basis function kernel. The model is tested by using the scikit-learn package (Pedregosa et al., 2011) and the model parameters are set as default;

3. FNN: a feed-forward neural network with two hidden layers, i.e. the multi-layer perceptron model. The dimension of the hidden layers equals the number of roadway locations/links, i.e. N;

4. LSTM (Ma et al., 2015): a long short-term memory neural network;

5. SGC+LSTM: a spectral graph convolution neural network stacked with an LSTM;

6. LSGC+LSTM: a localized graph convolution neural network (Defferrard et al., 2016) stacked with an LSTM, in which the hop of graph convolution is set as 3;

7. STGCN: Spatio-Temporal Graph Convolutional Networks, which is implemented used the source code [1]. The parameters of this model are kept the same as the source code in this experiment.

8. TGCLSTM (Cui et al., 2019): a traffic graph convolution LSTM incorporating the roadway physical properties. The hop of traffic graph convolution is also set as 3.

All the baseline models are implemented or imported from existing packages using Python 3.6.8. The deep learning based models are all implemented from scratch by the authors based on Pytorch 1.0.1. These baseline models are trained and tested on a Windows 10 computer with 32GB random-access memory (RAM) and one NVIDIA GTX 1080 Ti GPU with 11GB memory.

Hyper-parameters: The spatial dimension N is set according to the tested datasets mentioned in Section 4.1. The temporal dimension of the input sequence is set as 10, i.e. $T = 10$. Based on the empirical tuning, the graph wavelet kernel scale s is set as 0.08 for both datasets. For both datasets, the samples are randomized and divided into training, validation, and testing sets with a ratio of 7:2:1. The batch size is set as 40 and the training loss is based on mean square error (MSE). Since the RMSProp (Tieleman and Hinton, 2012) works well for RNNs, it is used as the gradient descent optimizer whose alpha (smoothing constant) is set as 0.99 and epsilon (the term added to the denominator to improve numerical stability) is set as 10^{-8}.

Due to the tested models have different amounts of weight parameters, the best learning rates for these models are also different. In the training process, for the GWGR model, the initial learning rate for the first 10 epochs is set as 0.01 and it decreases one order of magnitude every 10 epochs after the 10^{th} epoch. For other LSTM based models, based on the hyper-parameter tuning, the initial learning rate is set as 10^{-5}. In addition, the early stopping strategy is applied to the validation set to avoid overfitting. The training process will stop if the validation error cannot decrease 10^{-5} MSE within 10 patience steps.

Evaluation metrics: The performances of all tested models are evaluated by three metrics, including mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i| \#(24) \tag{5.24}$$

1 https://github.com/FelixOpolka/STGCN-PyTorch

Model	Urban Traffic Dataset			Freeway Traffic Dataset			Order of Magnitude of Weight Size
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	
ARIMA	4.8	0.32	13.51%	6.1	1.09	13.85%	
SVR	4.78	0.37	13.37%	6.85	1.17	14.39%	
FNN	2.31	0.17	8.35%	4.45	0.81	10.19%	N^2
LSTM	1.14	0.09	3.88%	2.7	0.18	6.83%	N^2
SGC+LSTM	1.07	0.08	3.74%	2.64	0.12	6.52%	N^2
LSGC+LSTM	1.38	0.12	4.54%	3.16	0.23	7.51%	N^2
TGCLSTM	1.02	0.07	3.28%	2.57	0.1	6.01%	N^2
STGCN	1.34	0.09	4.33%	2.64	0.1	6.12%	N
GWGR	0.93	0.07	2.67%	2.48	0.11	5.44%	N

Table 5.1: Prediction performance comparison for both datasets. (The weight matrices in LSTM and GWGR has N^2 and N weight parameters, respectively)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - \hat{x}_i}{x_i}\right| \tag{5.25}$$

$$RMSE = \left(\frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i|^2\right)^{1/2} \tag{5.26}$$

where x_i is the observed value and the $\hat{x}_i$ is the predicted value.

5.6 EXPERIMENTAL RESULTS

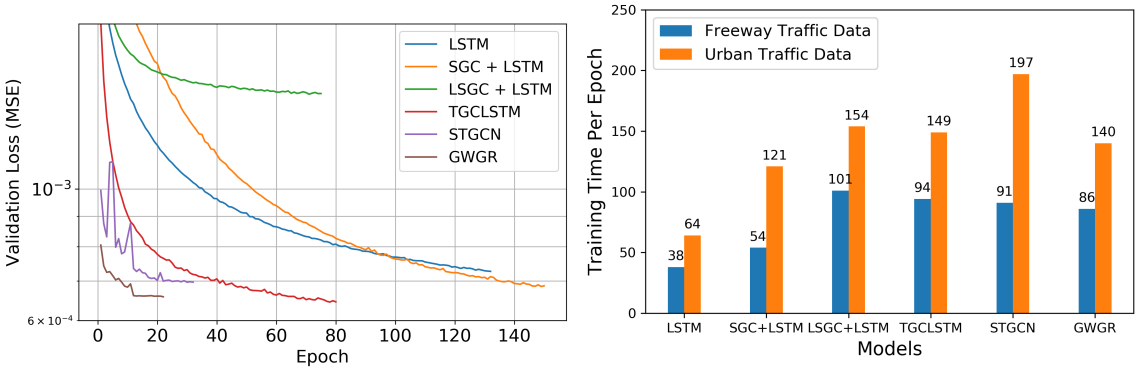
The tested prediction metrics of the urban traffic dataset and the freeway traffic dataset are shown in Table 5.1. The last column also shows the orders of magnitude of weight parameters in the tested neural networks with respect to the spatial dimension N. If the model is not based on neural network structures, such as ARIMA and SVR, the corresponding cell is intentionally left blank with a diagonal line.

The proposed GWGR outperforms all other compared models in terms of the prediction accuracy. In addition, the GWGR model requires one fewer order of magnitude of weight parameters than other models. We can notice that the deep learning based models apparently work better than ARIMA and SVR for both datasets, which means the classical prediction methods do not have enough power to handle the large-scale traffic prediction problem. Among the deep learning models, the FNN does not perform as good as the other LSTM-based models. The prediction MAEs of the LSTM drop to 1.14 and 2.7 miles per hour (mph) on the urban and freeway traffic datasets, respectively, which is a pretty good performance. The SGC+LSTM and TGCLSTM have much prediction improvement on the urban traffic dataset with respect to LSTM. But the LSGC+LSTM has worse results comparing to LSTM, which may be caused by the small number of weight parameters in the localized graph convolution. The GWGR obviously achieves superior prediction results in terms of MAE and MAPE.

Although the GWGR, SGC+LSTM, and TGCLSTM all have the gated recurrent structure and take the graph adjacency matrix A as a model component, Table 5.1 reveals that their capabilities of capturing spatiotemporal features from a road network based graph are totally different. The GWGR apparently works better and needs fewer weight parameters. The fewer weight parameters definitely reduce the complexity of the model, and thus, greatly enhance the interpretability of the model. Further, considering the weight matrices, such as $\Psi_s \Lambda_f^x \Psi_s^{-1}$, of the GWGR are sparse, the GWGR has the great opportunity to speed up the prediction by conducting sparse matrix multiplication and to find out the key links in the traffic network by analyzing the influentialness of each graph vertex.

5.6.1 *Training Efficiency*

The training efficiency of all LSTM-based models is demonstrated in this section. Figure 5.3 (a) shows the validation loss versus training epoch tested on the freeway traffic dataset. All the compared LSTM-based models, including LSTM, SGC+LSTM, and TGCLSTM, have N^2 magnitude of weight parameters. Comparatively, TGCLSTM converges much faster than LSTM and SGC+LSTM. The STGCN and GWGR contain less weight parameters and converge faster. During the training process, the validation loss of GWGR decrease smoother than that of STGCN.



(a) (b)

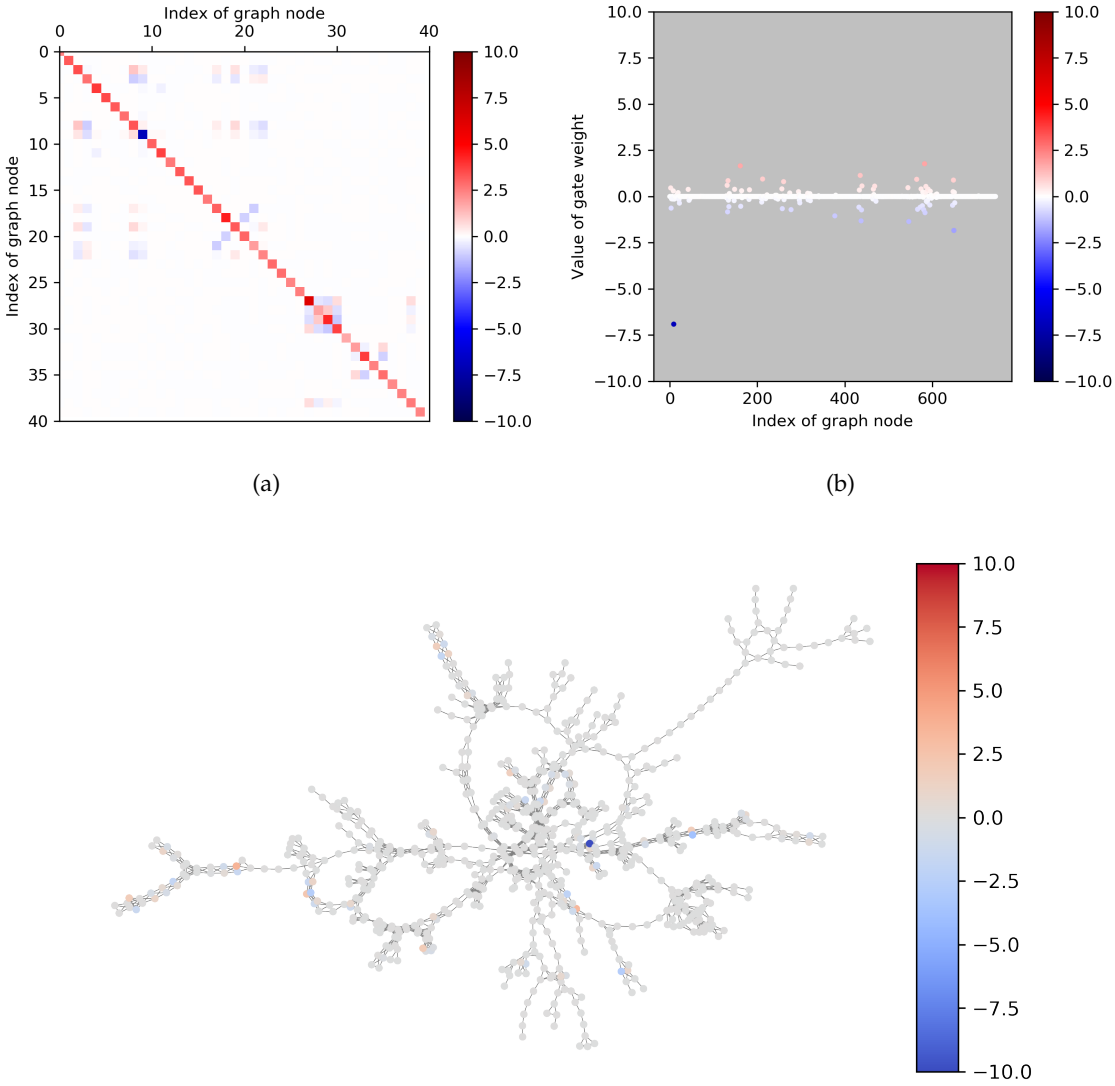
Figure 5.3: (a) Validation loss versus training epoch, tested on the freeway traffic dataset. (b) Training time per epoch, tested on both datasets.

In this study, the initial learning rate of GWGR is 10^{-2} and the GWGR nearly converges after 10 training epochs, which is a really fast training process.

Figure 5.3 (b) illustrates the training time per epoch on both datasets. Since the GWGR contains more matrix multiplication operations, such as $\Psi_s \Lambda_f^x \Psi_s^{-1}$, in the gate units, the running time of GWGR definitely is more than the vanilla LSTM. SGC+LSTM's training time per epoch is less than that of GWGR, while the training times per epoch of TGCLSTM and LSGC+LSTM are even more. Besides, STGCN also take more time per training epoch than GWGR. The training time per epoch of is also Considering that GWGR needs far less training epochs, the proposed GWGR model is still the most efficient one, comparing to the LSTM-based baseline models.

5.6.2 *Graph Wavelet Weight Analysis*

The model parameters of the GWGR consist of eight weight vectors and four bias vectors. The eight weight vectors are converted into diagonal weight matrices, such as Λ_f^x, and multiplied by graph wavelet matrices (Ψ_s and Ψ_s^{-1}) to form the graph wavelet weight matrices, such as $\Psi_s \Lambda_f^x \Psi_s^{-1}$ in the forget gate units, as shown in Equation (5.16-5.19). Due to the graph wavelet weight matrices are sparse and the size of the weight parameter is only $8N$ which is far fewer than N^2, the proposed GWGR is more interpretable than other LSTM based models.



(a) (b)

(c)

Figure 5.4: Graph wavelet weight matrix interpretation and visualization, taking the urban traffic dataset as an example. (a) Visualization of the left top part of forget gate input weight matrix $\Psi_s \Lambda_f^x \Psi^{-1}$, which is sparse and symmetric. (b) Scatter plot of the 9-th row of $\Psi_s \Lambda_f^x \Psi^{-1}$. (c) Visualization of weight on the 9-th row of $\Psi_s \Lambda_f^x \Psi^{-1}$ on the graph with a Kamada-Kawai layout.

In this section, the graph wavelet weight matrix of the input in the forget gate, $\Psi_s \Lambda_f^x \Psi_s^{-1}$, is selected from the GWGR tested on the urban traffic dataset and visualized to show the interpretability of the GWGR. Figure 5.4 (a) shows a squared section on the left top side of $\Psi_s \Lambda_f^x \Psi_s^{-1}$ containing 40 rows and 40 columns, which is sparse and symmetric. Since the $\Psi_s \Lambda_f^x \Psi_s^{-1}$

multiplies the input x_t in the forget gate, the colored pixels are the weights measuring the interactions between the speed values on different graph vertices, namely different roadway links.

When taking the 9-th row of $\Psi_s \Lambda_f^x \Psi_s^{-1}$ as an example, which contains an obvious blue dot at the 9-th column in Figure 5.4 (a), the columns with weight values away from zero are the ones contribute more to generate the filtered values of the 9-th graph vertex in the forget gate. Figure 5.4 (b) illustrates all the weight values of the 9-th row of $\Psi_s \Lambda_f^x \Psi_s^{-1}$. It can be found that only a small portion of points are away from zero. The 9-th vertex contributes the most to itself in this forget gate, since the absolute value of the weight value (the dark blue dot) which is around 7.1, is the highest. Further, from Figure 5.4 (b), it can be distinguished that some points with higher absolute weight values are far away from the 9-th points in terms of the index of the graph vertex. However, the distance between indices of vertices might not be the actual distance between vertices in the graph.

To measure whether those graph vertices with higher absolute weight values are away from the 9-th vertex in the graph, the graph structure with a Kamada-Kawai layout is visualized in Figure 5.4 (c). It is easy to find a dark blue point in Figure 5.4 (c), which is the 9-th vertex. Then, some other light red and light blue vertices are found to be far away from the 9-th vertex in the graph. This implies the graph wavelet transform operation works well that a graph vertex is not only influenced by its neighbors but also influenced by some other vertices, which might be the key vertices in the graph or the hotspot links in the traffic network. This sparse property of the graph wavelet is quite different from the localized graph convolution, whose vertices are only influenced by fixed hops of neighboring vertices in the graph.

5.6.3 *Graph Wavelet Weight Matrix Sparsity Analysis and Traffic Hotspot Detection*

Section 4.5 presents the analysis of weight sparsity of the graph wavelet in GWGR by visualizing one of the weight matrices ($\Psi_s \Lambda_f^x \Psi_s^{-1}$) based on the urban traffic dataset. However, in this section, we quantify the sparsity of all the eight graph wavelet matrices simultaneously in order to find the most influential vertices in the graph, i.e. the most influential roadway links in the traffic network. Since the urban traffic dataset has a more complicated traffic network, it is still used for an example in this section.

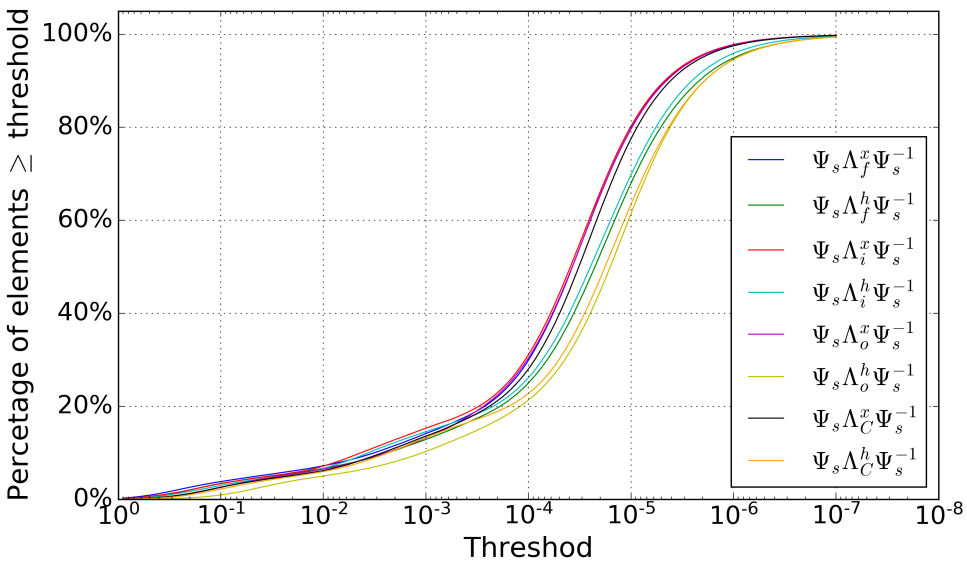
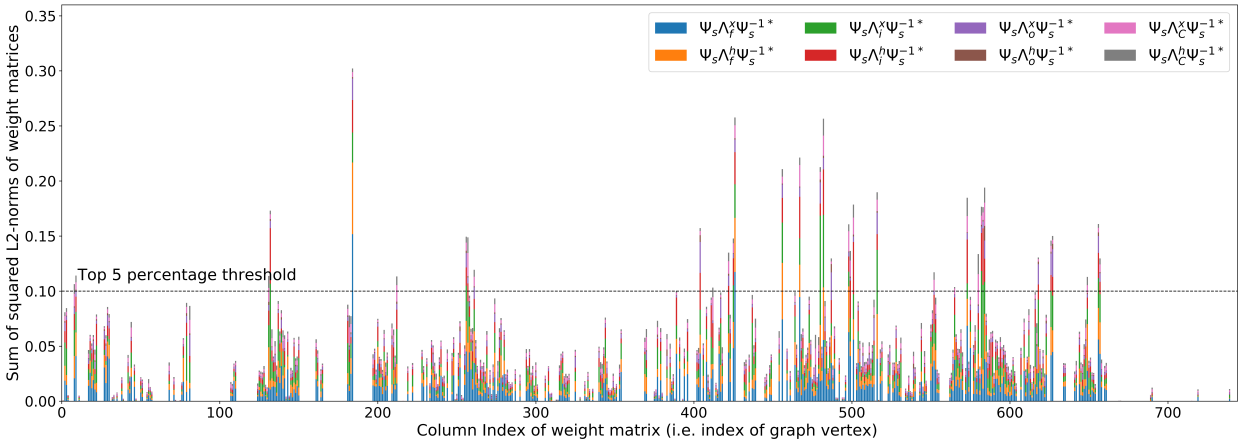


Figure 5.5: Percentage of elements in the graph wavelet weight matrices that is larger than the threshold

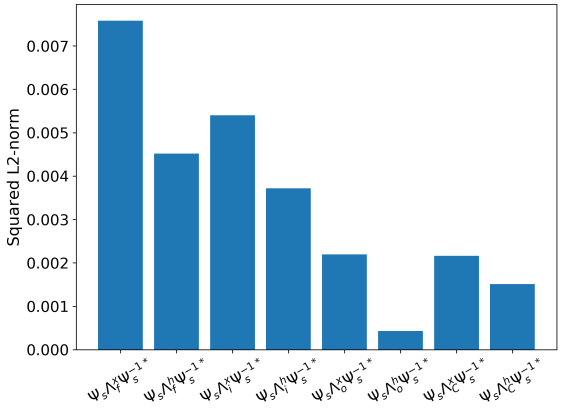
Figure 5.5 shows eight curves depicting how many element values in a graph wavelet weight matrix are larger than the thresholds represented by the x-axis. It can be noticed that, for all the eight matrices, only around 8% of the matrix elements are larger than 10^{-2} and only around 20% of the matrix elements are larger than 10^{-4}. Thus, it is proved that the graph wavelet is very sparse.

As tested, the values on the diagonal line of the graph wavelet weight matrices are normally larger than the non-diagonal values, which makes sense that the states of vertices tend to contribute more to themselves to generate future states. To measure the interactions between vertices and strip the vertices' self-influence out, we set all the diagonal values of the eight graph wavelet weight matrices to zero and these formatted matrices are denoted as $\Psi_s\Lambda\Psi_s^{-1^*}$, where $(\Psi_s\Lambda\Psi_s^{-1^*})_{ij} = 0$ if $i = j$, otherwise $(\Psi_s\Lambda\Psi_s^{-1^*})_{ij} = (\Psi_s\Lambda\Psi_s^{-1})_{ij}$. Here, the diagonal weight matrix is denoted as $\Lambda \in \{\Lambda_f^x, \Lambda_i^x, \Lambda_o^x, \Lambda_C^x, \Lambda_f^h, \Lambda_i^h, \Lambda_o^h, \Lambda_C^h\}$ for simplicity.

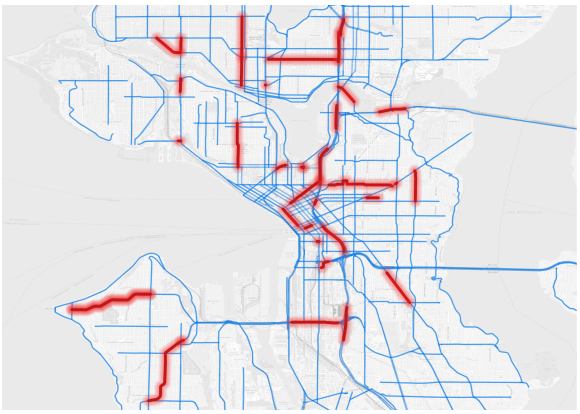
Since these formatted matrices are all sparse, the norm functions are good indicators to show the scale/magnitude of a matrix. Here, we choose to calculate the squared $\ell 2$-norm of each formatted weight matrices, i.e. $\left\|\Psi_s\Lambda\Psi_s^{-1^*}\right\|_2^2$, to measure the interactions between vertices. The $\ell 2$-norm of matrix $M \in \mathbb{R}^{P\times Q}$ is defined as $\sqrt{\sum_{i=1}^{P}\sum_{j=1}^{Q} M_{ij}^2}$. Intuitively, if a weight matrix has a larger squared ℓ^2-norm, it has larger absolute element values and contributes more to generate outputs.



(a)



(b)



(c)

Figure 5.6: (a) Bar chart of column-wise squared $\ell 2$-norms of each graph wavelet matrix. Eight column-wise squared $\ell 2$-norms calculated from the eight graph wavelet weight matrices are stacked. A horizontal dashed line shows the threshold of the top 5 percentage of sums of squared $\ell 2$-norms. (b) Squared $\ell 2$-norms of formatted graph wavelet weight matrices, i.e. $||\Psi_s \Lambda_f^x \Psi^{-1}||_2^2$. (c) Visualization of traffic network based on the urban traffic dataset. The roadway links having more impacts on other links (with top 5 percentage of largest column-wise squared $\ell 2$-norms) are highlighted with red color.

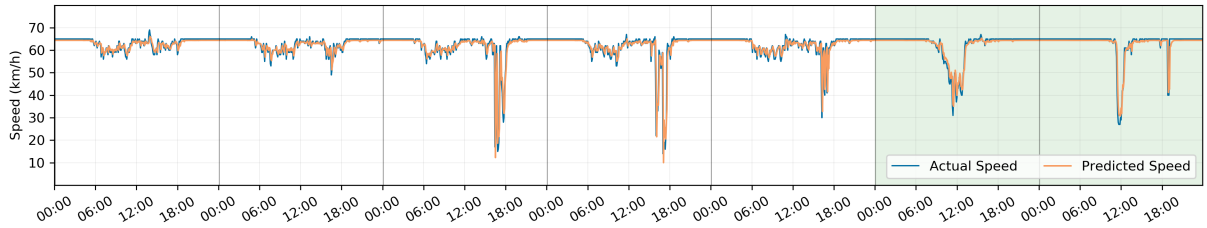
Figure 5.6 (b) shows that the squared $\ell 2$-norms in the forget gate and the input gate is larger than that of the output gate and the memory cell in GWGR. This implies the forget and input gates play more important roles during the prediction process, which is mutually corroborated by Greff et al. (2017) that the forget gate makes the LSTM-based structures more effective. Since the

vertices' self-influence has been stripped out, these norm values also reveal that the graph vertices have more interactions in the forget and input gates during the prediction process.

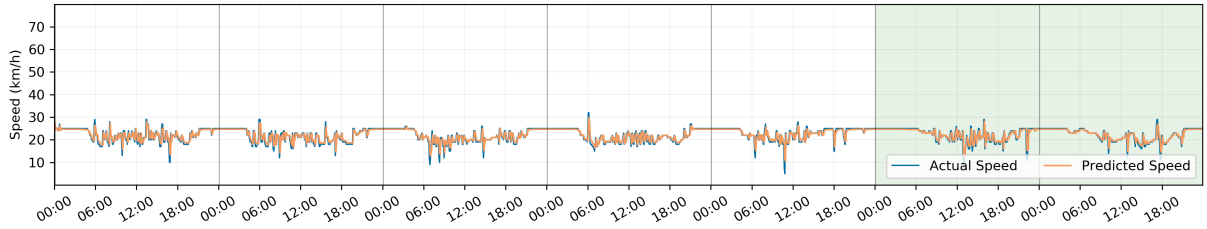
Further, to identify the most influential vertices in the graph, we analyze the column-wise squared $\ell 2$-norms of the eight formatted weight matrices. Actually, the column-wise squared $\ell 2$-norms equals to the column-wise squared $\ell 2$-norms, since the formatted weight matrix $\Psi_s \Lambda \Psi_s^{-1^*}$ is still symmetric. Mathematically, the column-wise squared $\ell 2$-norms of the k-th column of $\Psi_s \Lambda \Psi_s^{-1^*}$ can be written as $\sum_{i=1}^{N} (\Psi_s \Lambda \Psi_s^{-1^*})_{ik}^2$. Figure 5.6 (a) shows a bar chart, in which the stacked bars demonstrates the column-wise squared $\ell 2$-norms of the eight formatted weight matrices. The height of each bar indicates the sum of the squared $\ell 2$-norms of the eight formatted weight matrices at each corresponding column. In this way, the larger the total height of a bar is, the more impacts the graph vertex with the corresponding column index has on other vertices. It can be noticed that this bar chart is pretty sparse and a large portion of the sums of squared $\ell 2$-norms are close to zero. To identify the most influential links in the traffic network, we select the columns with top 5 percentage of the sums of squared $\ell 2$-norms and visualize these corresponding links on the real map, as shown in Figure 5.6 (c). There are 37 selected roadway links in total, which are highlighted with red color. Based on empirical investigation, these selected links are mostly located on principal arterials, intersections, freeways and freeway ramps, which are highly possible to be the hotspots in the urban traffic networks.

5.6.4 *Case Analysis on Two Tested Datasets*

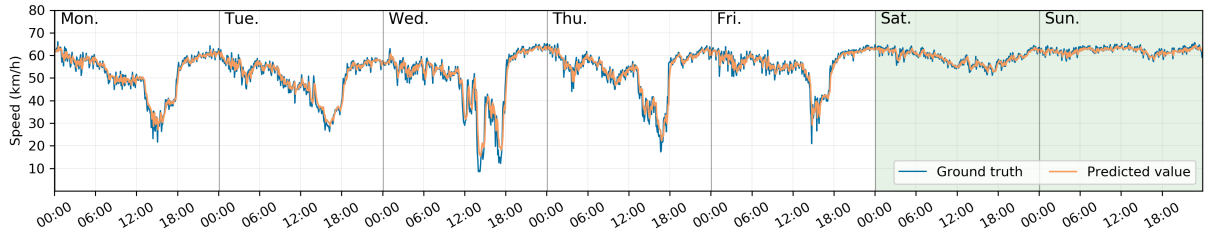
In this section, we compare the predicted speed values with the ground truth at roadway segments selected from both the urban traffic dataset and the freeway traffic dataset, as shown in Figure 5.7. All the predicted value curves and the ground truth covers fit well. Figure 5.7 (a) and (b) display the speed values on two different roadway links selected from the urban traffic dataset. Due to the urban area contains various types of roadways and the traffic flows are controlled by a large number of traffic lights, the urban traffic pattern is more complicated. Figure 5.7 (b) shows the speed values extracted from an urban corridor which are apparently smaller than the speed values extracted from a highway link, as shown in Figure 5.7 (a). Figure 5.7 (c) and (d) depict



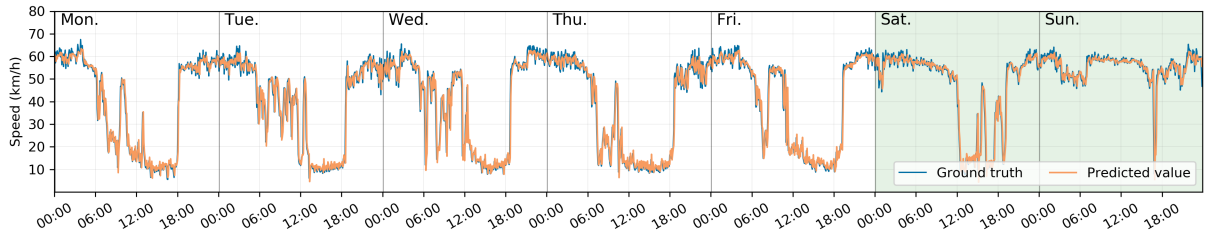
(a) Urban traffic dataset. Highway link. Road link ID: 114+04554. December 03-09, 2012.



(b) Urban traffic dataset. Urban corridor. Road link ID: 114+08028. December 03-09, 2012.



(c) Freeway traffic dataset. The sensor locates at milepost 150 of I-5, southbound. December 07-13, 2015.



(d) Freeway traffic dataset. The sensor locates at milepost 167of I-5, southbound. December 07-13, 2015.

Figure 5.7: Comparison of ground truth and predicted speed values tested on both datasets.

the speed values of two sensing locations selected from the freeway traffic dataset, which have different peak hours.

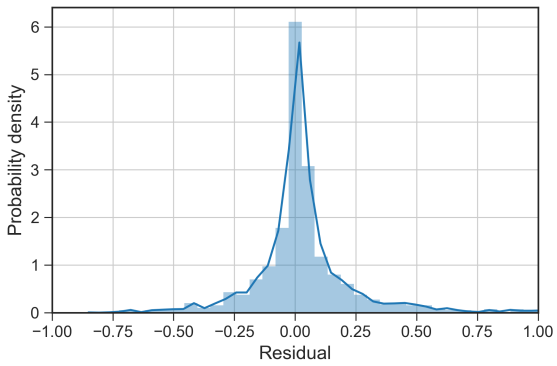
The time spans of the four sub-figures all last for one week. The difference in traffic patterns between weekdays and the weekend is also very obvious that traffic speed values fluctuate less during the weekend. In addition, we can observe that the variation of ground truth values of the urban traffic data is smaller than that of the freeway traffic data, especially at midnight, which leads

the prediction error of urban traffic data is relatively smaller. In summary, as demonstrated by these figures, the proposed GWGR model has the ability to make reliable predictions simultaneously for various roadways in the traffic network.

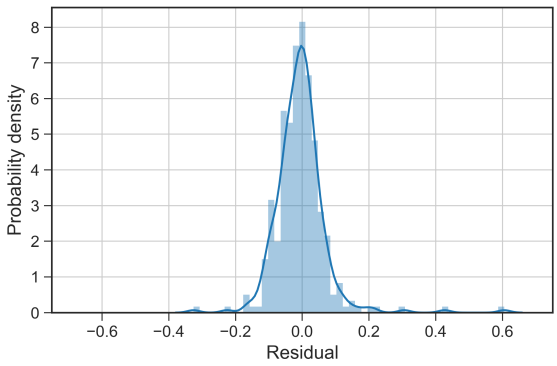
5.6.5 *Residual Analysis*

Due to residual is an important indicator to evaluate whether a model is systematically correct, the residuals of the predictions are analyzed in this section. The residual r equals the ground truth value subtracts the predicted value, i.e. $r = x - \hat{x}$. Figure 5.8 (a) and (b) shows the residual distributions of the testing sets of both urban and freeway traffic datasets. Basically, the residuals of both tested datasets follow normal distributions with zero means. Although the proposed GWGR model is far more complicated than a regression model, the residual distributions with zero mean indicate the trained GWGR model has acquired sufficient predictive information.

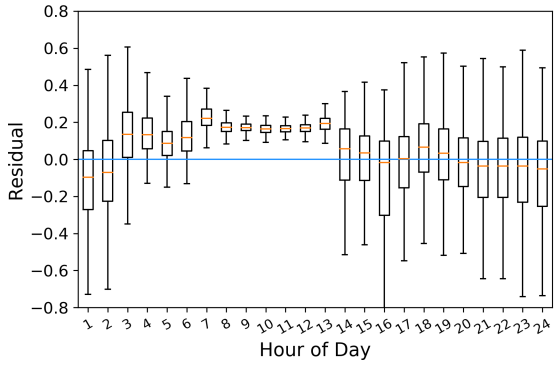
Moreover, due to traffic prediction accuracy is highly influenced by temporal information, such as the time of day, the residuals with regard to the hour of day are also evaluated by drawing boxplots, as presented in Figure 5.8 (c) and (d). Figure 5.7 (c) shows that the residuals of urban traffic data tend to be positive in the day time. It reveals that the traffic speed of roads in the urban area might have more chances to suddenly increase in the day time, taking the case shown in Figure 5.7 (b) as an example, leading that the predicted values are smaller than the ground truth. In the opposite, Figure 5.8 (d) shows that, at peak hours, the residuals of freeway traffic data tend to be negative. It also makes sense that the sudden drops of the traffic speed at rush hours will possibly result in the predicted values are larger than the ground truth. Overall, the residuals presented in Figure 5.8 (c) and (d) are distributed around the zero lines indicating the proposed model is capable of fitting datasets with different complexities and traffic patterns and achieving good prediction performance.



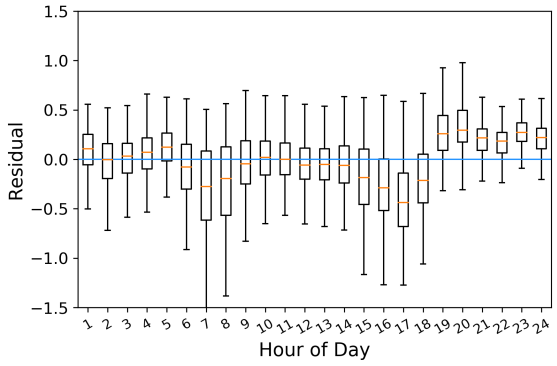
(a) Residual distribution of the testing set (urban traffic dataset)



(b) Residual distribution of the testing set (freeway traffic dataset)



(c) Boxplot of residual w.r.t. hour of day (urban traffic dataset)



(d) Boxplot of residual w.r.t. hour of day (freeway traffic dataset)

Figure 5.8: Residual analysis plots generated based on both the urban traffic dataset and the freeway traffic dataset.

5.7 CHAPTER SUMMARY

In this chapter, we propose a graph wavelet gated recurrent neural network to predict network-scale traffic speed information. Since that traffic status on a road segment is highly influenced by the upstream/downstream segments and nearby bottlenecks in the traffic network, we consider the traffic network as a graph and learn flexible neighboring features from each graph node. The graph wavelet is incorporated as a key component responsible for extracting localized spatial feature. Comparing to graph convolution based model, the proposed GWGR model is more flexible, because it does not need to specify the order of hops for the vertices in the graph. To

the best of our knowledge, this is the first time that a graph wavelet based neural network is utilized for traffic forecasting. The GWGR model is tested on two real-world traffic datasets and the experimental results show that it can achieve superior prediction performance comparing to multiple classical and deep learning based models. Moreover, the GWGR contains less weight parameters than other LSTM-based models and achieves higher training efficiency. In addition, the model's sparse weight matrices are comprehensively analyzed and visualized. The sparsity of the weight matrices can help interpret the model and identify the most influential vertices in the traffic network based graph.

In the future, we will investigate more on the interpretation of the graph wavelet and apply graph based theories for the reasoning of traffic congestions. [2]

2 This chapter is a slightly modified version of "**Learning Traffic as a Graph: A Gated Graph Wavelet Recurrent Neural Network for Network-scale Traffic Prediction**" published in *Transportation Research Part C: Emerging Technologies* and has been reproduced here with permission from the copyright holder. (doi: https://doi.org/10.1016/j.trc.2020.102620)

Part II

PREDICTION WITH MISSING VALUES

6

LSTM RECURRENT NEURAL NETWORK FOR FORECASTING NETWORK-WIDE TRAFFIC STATE WITH MISSING VALUES

6.1 OVERVIEW

Short-term traffic forecasting based on data-driven models for ITS applications has great influence on the overall performance of modern transportation systems (Vlahogianni et al., 2014). In the last three decades, a large number of methods have been proposed for traffic forecasting in terms of predicting speed, volume, density and travel time. Studies in this area normally focus on the methodology components, aiming at developing different models to improve prediction accuracy, efficiency, or robustness. Previous literature (Ma et al., 2015; Vlahogianni et al., 2014) indicates that the existing models can be roughly divided into two categories, i.e. classical statistical methods and computational intelligence (CI) approaches. Classical statistical models, such as Autoregressive Integrated Moving Average (ARIMA) and its variants (Chandra and Al-Deek, 2009; Williams and Hoel, 2003), have made great contributions to address the traffic prediction problem. With the ability to deal with high dimensional data and the capability of capturing non-linear relationship, CI approaches, especially novel machine learning methods, tend to outperform the statistical methods with respect to handling complex traffic forecasting problems (Karlaftis and Vlahogianni, 2011). The representative machine learning methods include support vector regression (Asif et al., 2013), K-nearest neighbor (Cai et al., 2016), etc. Besides, nonparametric approaches, such Kalman filter and its variants (Chien et al., 2003; Van Lint, 2008), and matrix/tensor factorization methods (Tan et al., 2016) are also widely used in traffic prediction problems.

Deep learning models, as a branch of machine learning models, become popular and rapidly be adopted in the traffic forecasting area. Most of the newly proposed traffic forecasting models (Chen et al., 2016; Duan et al., 2016b; Ma et al., 2015; Song et al., 2016; Wu and Tan, 2016; Zhao et al., 2017) are based on recurrent neural networks (RNNs), which mainly process sequence data by maintaining a chain-like structure and internal memory with loops (Jozefowicz et al., 2015). To address RNN's exploding gradient problem, Long Short-Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997) and the Gated Recurrent Unit network (GRU) (Cho et al., 2014) were designed to learn long-term dependencies of sequence data via gate and memory units. Many recent studies (Chen et al., 2016; Duan et al., 2016b; Zhao et al., 2017) adopted the LSTM as a baseline or building blocks in their proposed models for traffic forecasting. Although RNN and its variants have been adopted as building blocks of traffic prediction models, few studies reformulated their model structure to improve traffic prediction accuracy and robustness. In this study, we focus on RNN-based models and attempt to design a better structure to solve the traffic prediction problem. Three primary limitations of existing RNN-based models for traffic forecasting can be summarized as follows: 1) Few existing models are capable of dealing with missing data. 2) Although time series of traffic states are normally processed in a chronological order to capture the forward dependencies, backward dependencies in traffic state sequences, which can be learned in a reverse-chronological order, has not been explored. 3) Few studies evaluate the trade-off between model capacity and complexity.

Firstly, missing data is a common problem in the traffic data collection process due to sensor or communication failure. Various data imputation methods for time series have been developed and applied to estimate missing data. However, solving the imputation and prediction tasks at the same time often leads to a two-step process where imputation and prediction models are separated (Che et al., 2018). In this way, the missing patterns of the data cannot be effectively explored in prediction models, and thus may result in biased prediction results (Wells et al., 2013). In some real-time traffic forecasting scenarios, the assumption of data imputation methods may not be satisfied, and thus, missing data cannot be imputed in real-time. Further, it is usually computationally expensive for training and applying these imputation methods. Due to RNNs for times series with missing values have been explored and applied (Che et al., 2018; Lipton et al., 2016), RNN-based models have the potential to combine imputation methods with prediction

models. Considering the ability of LSTM to capture and maintain long-term dependencies, LSTM is even more suitable for time series imputation. From another perspective, the ability to impute missing values in time series can be regarded as a capability of processing unevenly spaced time series, which is unachievable for most of the LSTM-based traffic prediction models (Chen et al., 2016; Duan et al., 2016b; Ma et al., 2015; Song et al., 2016; Yu et al., 2017a). Hence, exploiting the power of customized LSTM to predict traffic states with missing values, as one of our main motivations, is promising and attainable. In this study, we propose a customized LSTM structure with an imputation unit (LSTM-I) to fulfill this goal.

The second improvable aspect of previous work is the learning order of the traffic state time series in RNN-based models. Normally, the dataset fed to an LSTM model is chronologically arranged and the model's chain-like structure makes use of the forward dependencies. But in this process, it is possible that useful information does not efficiently pass through the chain-like gated structure. Therefore, it may be informative to consider backward dependencies into consideration by processing series data in a negative direction. Another reason for including backward dependencies in our study is the periodicity of the traffic states. Traffic conditions have strong periodicity and regularity, and even short-term periodicity can be observed (Jiang and Adeli, 2004). According to (Box et al., 2015), analyzing the periodicity of time series data from both forward and backward temporal perspectives will enhance the predictive performance. Besides, the impact of upstream and downstream traffic states with respect to a road segment in the traffic network should not be neglected. Previous studies (Chandra and Al-Deek, 2009; Kamarianakis et al., 2010) found that past speed values of upstream and downstream have an influence on the future speed values of a location along a corridor. For complicated traffic networks with intersections and loops, upstream and downstream both refer to relative positions and two arbitrary locations can be upstream and downstream of each other. Upstream and downstream are defined with respect to space, while forward and backward dependencies are defined with respect to time. With the help of the forward and backward dependencies of spatial-temporal data, the learned features will be more comprehensive. Based on our review of the literature, few studies on traffic analysis utilized the backward dependency. To fill this gap, a bidirectional LSTM (BDLSTM) with the ability to deal with both forward and backward dependencies is adopted as a component of the proposed network framework in this study.

The third limitation of previous work is the lack of trade-off evaluation between model capacity and complexity. Some newly proposed LSTM-based prediction models (Ma et al., 2015) have only one LSTM layer to deal with time series. Existing studies (LeCun et al., 2015) have shown that deep LSTM architectures with several hidden layers can build up progressively higher levels of representations of sequence data. Although some studies (Chen et al., 2016; Wu and Tan, 2016; Yu et al., 2017b) utilized more than one LSTM layers, the influence of the number of LSTM layers needs to be further evaluated. Furthermore, the impact of the number of other layers, the size of model weights, and the spatial dimension size of the network-wide traffic data should also be evaluated as influential factors of prediction performance.

6.1.1 *Contribution and Organization of the Chapter*

In this chapter, we focus on RNN-based models and attempt to reformulate the way to incorporate RNNs into traffic prediction models, even when the input traffic data contains missing values. We propose a stacked bidirectional and unidirectional LSTM network architecture (SBU-LSTM) for network-wide traffic state prediction to address the aforementioned shortcomings. The evaluation of the prediction capability of stacked LSTM- or BDLSTM-based models has the potential to facilitate the further research on the deep learning model design for traffic prediction problems. Experiments based on two real-world datasets with different missing value patterns indicate that the proposed architecture can achieve outstanding prediction results. In summary, our contributions can be summarized as follows:

1. We propose an LSTM structure with an imputation unit, i.e. LSTM-I, to infer and fill the missing values in the spatial-temporal input data and in return to help improve prediction accuracy.

2. We propose a stacked bidirectional and unidirectional LSTM architecture. i.e. SBU-LSTM, for network-wide traffic forecasting. This stacked architecture with multiple layers is flexible. The evaluation of the prediction capability of stacked LSTM- or BDLSTM-based models has great potential to facilitate the design of neural network models for traffic prediction.

3. The trade-off between model capacity and complexity is evaluated and discussed.

4. Two real-world traffic state data is tested in this study and the LOOP-SEA dataset is published via Github (Cui et al., 2016) and Zenodo (Wang et al., 2019b).

The remainder of the chapter is organized as follows: In Section 6.2 presents existing and representative related work. We introduce several existing neural networks and the proposed LSTM-I in Section 6.3notions that will be used for building the traffic graph convolution module in Section 4.3. Then, we present the experiment settings, experimental results, and analysis, including prediction accuracy, imputation performance, and training efficiency, in Section 6.4. In the end, Section 6.5 concludes the chapter by showing the contribution of the proposed model and identifying future directions.

6.2 RELATED WORK

6.2.1 *Deep Learning based Traffic Prediction*

Deep learning-based models generated the state-of-the-art performance for traffic forecasting. Ever since the precursory study of utilizing NN into the traffic prediction problem was proposed (Hua and Faghri, 1994), many NN-based methods, like feed-forward NN (Gers et al., 1999), fuzzy NN (Yin et al., 2002), and recurrent NN (RNN) (Van Lint et al., 2002) are adopted for traffic forecasting problems more than ten years ago. Ma et al. (2015) firstly adopt the LSTM to forecast traffic speed. Several other studies utilize the LSTM to forecast travel time (Duan et al., 2016b), congestion (Chen et al., 2016), and traffic flow (Zhao et al., 2017). Song et al. (2016) utilized shared hidden LSTM layers to help predict human mobility and transportation mode. Cui et al. (2017) adopted bidirectional LSTM in the traffic prediction problem. Yu et al. (2017a) combine the convolutional neural network with RNN to predict transportation states. There are many studies forecasting traffic from other perspectives, like learning traffic as images (Ma et al., 2017) or graphs (Cui et al., 2019) by combining convolutional based neural networks.

6.2.2 *Combining Imputation and Prediction*

One option to deal with missing values is the skipping mechanism, which is usually used in the dropout process of RNNs (Gal and Ghahramani, 2016). The other one is data imputation. Interpolation (Kreindler and Lumsden, 2012), and spline (De Boor et al., 1978) methods are simple and efficient for data imputation, but they cannot capture variable correlations and complex patterns to perform imputation. Various data imputation methods for time series, including regression (Chen et al., 2003), spectral analysis (Mondal and Percival, 2010), EM algorithm (Garcia-Laencina et al., 2010), and matrix factorization (Koren et al., 2009) (Chen et al., 2019b) have been developed and applied to estimate missing data. Among these non-deep learning-based models, matrix factorization methods normally can achieve state-of-the-art prediction accuracy. A new Bayesian temporal matrix factorization method is proposed by Sun and Chen (2019) to solve spatiotemporal data prediction problems when there is missing values in the input data. This method can deal with missing values and achieve good prediction accuracy. However, the mechanisms of these methods and the sizes of the dataset used to train models are greatly different from those of deep learning-based models. Besides, combining these data imputation models with traffic prediction models often leads to a two-step process. To overcome this weakness, Che et al. (2018) firstly exploit to use a GRU-based model, GRU-D, to combine imputation and prediction models by designing a decay mechanism for data imputation.

6.3 METHODOLOGY

6.3.1 *Notations*

A time series of network-wide traffic states with D sensor stations can be denoted as $X = \{x_1, x_2, ..., x_T\}^T \in \mathbb{R}^{T \times D}$ with T time steps. Each vector $x_t \in \mathbb{R}^D$ denotes the D sensors' traffic states at time t, whose elements x_t^d represents the traffic state of d-th sensor station. It should be noted that the traffic state can refer to traffic speed, travel time, traffic volume, etc. In this

chapter,the traffic state specifically refers to traffic speed, which is consistent with the tested datasets in the experiment section.

In reality, traffic sensors, like inductive loop detectors, may fail due to the breakdown of wire insulation or damage caused by construction activities or electronics unit failure. The sensor failure further will lead to missing values in the collected data. To deal with missing values, a *masking* vector $m_t \in \{0,1\}^D$ is adopted to denote whether traffic states are missing at time step t. The masking vector for x_t is defined as

$$m_t^d = \begin{cases} 1, & \text{if } x_t^d \text{ is observed} \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

Accordingly, for a traffic state data sample $X \in \mathbb{R}^{T \times D}$, we can get a masking data sample, $M = \{m_1, m_2, ..., m_T\}^T \in \mathbb{R}^{T \times D}$.

In this study, the traffic state prediction problem aims to learn a function $F(\cdot)$ to map T steps of historical traffic state data to the next subsequent step of traffic state data, which can be described as:

$$F([x_1, x_2, ..., x_T]; [m_1, m_2, ..., m_T]) = [x_{T+1}] \tag{6.2}$$

6.3.2 *Long Short-Term Memory*

It has been showed that LSTMs work well on sequence-based tasks with long-term dependencies (Chen et al., 2016; Duan et al., 2016b; Song et al., 2016; Wu and Tan, 2016; Yu et al., 2017a; Zhao et al., 2017). Although a variety of LSTM variants were proposed in recent years, a comprehensive analysis of LSTM variants shows that none of the variants can improve upon the standard LSTM architecture significantly (Greff et al., 2017). Thus, we adopt the LSTM as the base model in this study.

The structure of LSTM has been introduced in Chapter 3. A clearer structure of LSTM is shown in Figure 6.1 (a). At time step t, the LSTM layer maintains a hidden memory cell $\tilde{C}_t$ and three gate units, which are input gate i_t, forget gate f_t, and output gate o_t. The LSTM cell takes the current variable vector x_t, the preceding output h_{t-1}, and the preceding cell state C_{t-1} as inputs. With the memory cell and gate units, LSTM can learn long-term dependencies to allow useful information

to pass along the LSTM network. Gate structures, especially the forget gate, help LSTM to be an effective and scalable model for sequential data learning problems (Greff et al., 2017). The input gate, forget gate, output gate, and memory cell in an LSTM cell are represented by blue boxes in Figure 6.1 (a). They can be calculated using the following equations:

$$f_t = \sigma_g(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \tag{6.3}$$

$$i_t = \sigma_g(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \tag{6.4}$$

$$o_t = \sigma_g(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \tag{6.5}$$

$$\tilde{C}_t = \tanh(W_C \cdot x_t + U_C \cdot h_{t-1} + b_C) \tag{6.6}$$

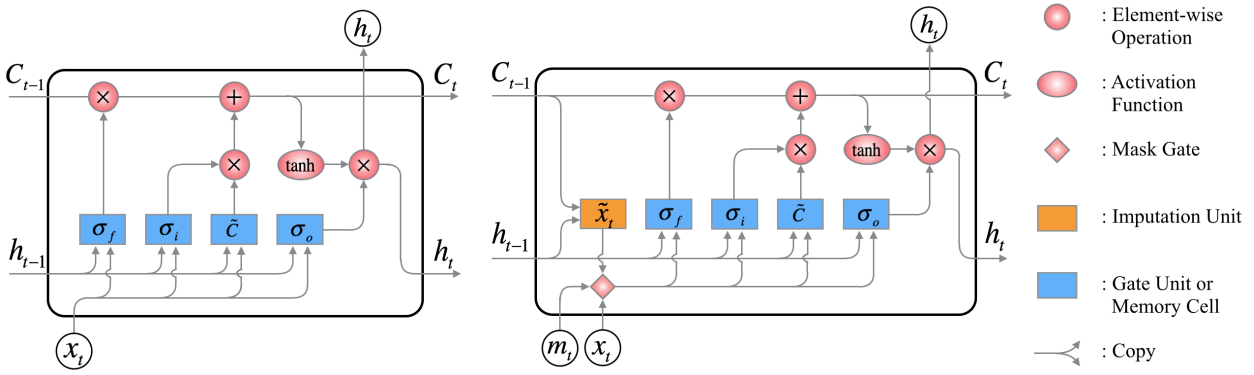
where $\cdot$ is the matrix multiplication operator. W_f, W_i, W_o, and W_C are the weight matrices mapping the hidden layer input to the three gate units and the memory cell. U_f, U_i, U_o, and U_C are the weight matrices connecting the preceding output to the three gates and the memory cell. b_f, b_i, b_o, and b_C are four bias vectors. $\sigma_g(\cdot)$ is the gate activation function, which is a Sigmoid function here, and $\tanh(\cdot)$ is the hyperbolic tangent function. Then, the cell output state C_t and the layer output h_t can be calculated as follows:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{6.7}$$

$$h_t = o_t \odot \tanh(C_t) \tag{6.8}$$

where $\odot$ is the element-wise vector/matrix multiplication operator.

The output of an LSTM layer can be a set of outputs from all T steps, represented by $H_T = [h_1, h_2, ..., h_T]$. Here, when taking the traffic prediction problem (Equation 6.2) as an example, only the last element of the output vector h_T is what we want to predict. Hence, the predicted value for



(a) LSTM (b) LSTM-Imputation

Figure 6.1: (a) Structure of the vanilla LSTM. (b) Structure of the LSTM-I. The mask gate determines the positions of the missing values. The missing input values can be imputed via the imputation unit and the inferred/imputed values can assist the training process by adding a regularization term to the loss function.

the subsequent time step $T+1$ is $\hat{x}_{T+1} = h_T$. In the training process, the model's total loss $\mathcal{L}$ at each iteration can be calculated by

$$\mathcal{L} = \text{Loss}(\hat{x}_{T+1} - x_{T+1}) = \text{Loss}(h_T - x_{T+1}) \tag{6.9}$$

where $\text{Loss}(\cdot)$ is the loss function, which is normally a mean square error function for traffic prediction problems.

6.3.3 *LSTM with Imputation Unit*

For the LSTM-based prediction problem, if the input time series contains missing/null values, the model will fail due to null values cannot be computed during the training process. If the missing values are set as some pre-defined values, like zeroes, mean of historical observations, or last observed values, these biased model inputs will result in biased parameter estimation in the training processing (Che et al., 2018). Further, solving the imputation and prediction tasks at the same time often results in separated imputation and prediction models.

To fulfill data imputation and traffic prediction in one model, we propose an LSTM-based model with an imputation unit, called **LSTM-I**. Unlike the GRU-D (Che et al., 2018) targeting on inferring missing values based on the historical mean and the last observation with a learnable decay rate,

the proposed LSTM-I aims to infer missing values at current time step from preceding LSTM cell states and hidden states. The weight parameters in the imputation unit are learnable. Further, the values inferred from the imputation values can contribute in the training process. In this way, the LSTM-I can complete the data imputation and prediction tasks at the same time. Thus, it is particularly suitable for online traffic prediction problems, which may frequently encounter missing values issues. Please note that the inferred values may not be the "actual" missing values, since the proposed imputation unit is only designed for generating appropriate values to help the calculation process in the LSTM structure work properly and generate accurate predictions.

In LSTM-I, we design an imputation unit σ_p, which is fed with the preceding cell state C_{t-1} and the preceding output h_{t-1}, to infer the values of the subsequent observation, as shown in Figure 6.1 (b). The inferred observation $\tilde{x}_t \in \mathbb{R}^D$ is denoted as

$$\tilde{x}_t = \sigma_g(W_I \cdot C_{t-1} + U_I \cdot h_{t-1} + b_I) \tag{6.10}$$

where W_I and U_I are the weights and b_I is the bias in the imputation unit. Then, each missing element of the input vector is updated by the inferred element

$$x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d)\tilde{x}_t^d \tag{6.11}$$

where $\tilde{x}_t^d$ is the d-th element of $\tilde{x}_t$. According to Equation 6.11, if x_t^d is missing, m_t^d is zero and x_t^d is imputed by $\tilde{x}_t^d$.

Besides, since each masking vector m_t contains the positions/indices of missing values at time step t, the masking vector is also fed into the model and the LSTM-I structure can be characterized as

$$x_t = m_t \odot x_t + (1 - m_t) \odot \tilde{x}_t \tag{6.12}$$

$$f_t = \sigma_g(W_f \cdot x_t + U_f \cdot h_{t-1} + V_f \cdot m_t + b_f) \tag{6.13}$$

$$i_t = \sigma_g(W_i \cdot x_t + U_i \cdot h_{t-1} + V_i \cdot m_t + b_i) \tag{6.14}$$

$$o_t = \sigma_g(W_o \cdot x_t + U_o \cdot h_{t-1} + V_o \cdot m_t + b_o) \tag{6.15}$$

$$\tilde{C}_t = \tanh(W_C \cdot x_t + U_C \cdot h_{t-1} + V_C \cdot m_t + b_C) \tag{6.16}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{6.17}$$

$$h_t = o_t \odot \tanh(C_t) \tag{6.18}$$

where V_f, V_i, V_o, and V_C are weight parameters for the masking vector m_t in different gates. If there are no missing values in the input data, all elements in m_t are zeros and the structure of LSTM-I is identical to that of LSTM.

Furthermore, the imputation unit can contribute to the training process. According to Equation 6.10, at each time step $t-1$, the imputation unit infers the input x_t and generates $\tilde{x}_t$ no matter x_t contains missing values or not. When x_t^d is not missing, x_t^d can help LSTM-I evaluate the correctness of the inferred value $\tilde{x}_t^d$ by quantifying the difference between x_t^d and $\tilde{x}_t^d$. Thus, we can add a regularization term to the model's total loss (Equation 6.9) at each iteration as follows

$$\mathcal{L} = \text{Loss}(h_T - x_{T+1}) + \lambda \sum_{t=1}^{T} \sum_{m_t^d \neq 0} |x_t^d - \tilde{x}_t^d| \tag{6.19}$$

where λ is the penalty term and the regularization term $\sum_{t=1}^{T} \sum_{m_t^d \neq 0} |x_t^d - \tilde{x}_t^d|$ measures the total absolute imputation error during a training iteration. By adding the regularization term to the loss, the imputation performance can be enhanced, and it has the potential to improve the model's overall prediction accuracy.

6.3.4 *Bidirectional LSTMs*

The idea of the BDLSTM comes from the bidirectional RNN (Schuster and Paliwal, 1997), which processes sequence data in both forward and backward directions with two separate LSTM

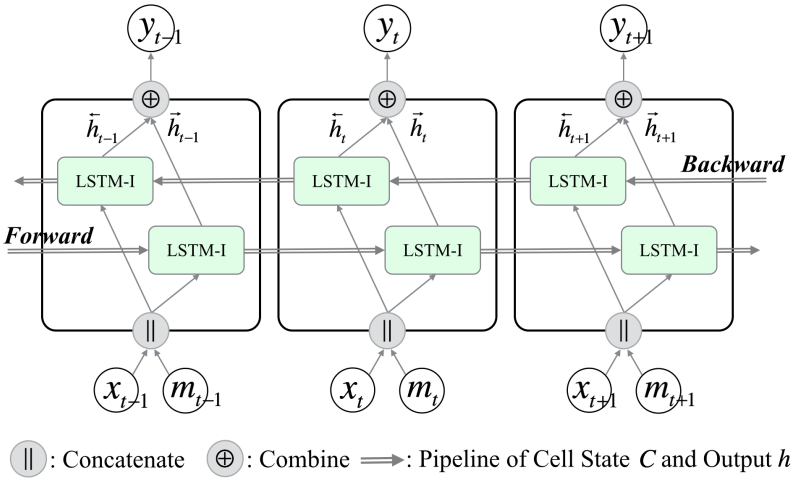


Figure 6.2: Structure of Bi-Directional LSTM-I.

hidden layers. It has been proved that the bidirectional networks are substantially better than unidirectional ones in many fields, like phoneme classification (Graves and Schmidhuber, 2005) and speech recognition (Graves et al., 2013). But, based on our review of the literature (Chen et al., 2016; Duan et al., 2016b; Ma et al., 2015; Wu and Tan, 2016; Yu et al., 2017b), BDLSTMs have not been utilized in traffic prediction problems. Due to the several aforementioned reasons in the introduction section, BDLSTMs are fit for handling network-wide traffic prediction problems. Thus, we adopt the BDLSTM as one component of our proposed framework.

To let BDLSTMs be able to deal with time series with missing data, we propose a BDLSTM with Imputation unit, called **BDLSTM-I**, in which LSTM components are replaced with LSTM-I components, as shown in Figure 6.2. The imputation mechanism of the LSTM-I is to infer the current missing input from the preceding cell state and output. By adopting the BDLSTM-I, missing values can be imputed from both the forward and the backward LSTM-Is. It is equivalent to infer the missing value at the current time step twice from both the preceding and the subsequent time steps, respectively. Thus, the advantage of the BDLSTM-I is that missing values are imputed based on both forward and backward temporal dependencies. Considering the periodic traffic patterns and the interaction between downstream and upstream in traffic networks, BDLSTM-I has the potential to reduce bias in the imputation process and further enhance traffic prediction.

The structure of an unfolded BDLSTM-I layer contains a forward LSTM-I layer and a backward LSTM-I layer, which is illustrated in Figure 6.2. The forward layer output, $\overrightarrow{h}_t$, is iteratively

calculated based on positive ordered inputs $[x_1, x_2, ..., x_T]$ and masks $[m_1, m_2, ..., m_T]$. The backward layer output, $\overleftarrow{h}_t$, is iteratively calculated using the reversed ordered inputs and masks from time step T to time step 1. Both forward and backward outputs are calculated based on the LSTM-I model equations (Equations 6.10 - 6.18). The BDLSTM-I layer generates output element y_t at each step t based on the combination of $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ by using the following equation:

$$y_t = \oplus(\overrightarrow{h}_t, \overleftarrow{h}_t) \tag{6.20}$$

where $\oplus$ is an average function. It should be noted that other functions, such as summation, multiply, or concatenate functions, can be used instead. Similar to the LSTM-I layer, the final output of a BDLSTM layer can be represented by a vector $Y = [y_1, y_2, ..., y_T]$.

If solely using one-layer BDLSTM-I for the prediction task, the loss function of BDLSTM-I should be defined based on that of LSTM-I. However, due to BDLSTM-I has two LSTM-I arranged in two directions, the regularization term can be slightly adjusted as

$$\mathcal{L} = \text{Loss}(y_T - x_{T+1}) + \lambda \sum_{t=1}^{T} \sum_{m_t^d \neq 0} \frac{1}{2}(|x_t^d - \overrightarrow{\tilde{x}_t^d}| + |x_t^d - \overleftarrow{\tilde{x}_t^d}|) \tag{6.21}$$

where $\overrightarrow{\tilde{x}_t^d}$ and $\overleftarrow{\tilde{x}_t^d}$ denote the inferred values from forward and backward LSTM-Is, respectively. In this way, the imputation errors of the two LSTM-Is are averaged.

6.3.5 *Stacked Bidirectional and Unidirectional LSTM Network Architecture*

Existing studies (Graves et al., 2013; LeCun et al., 2015) have shown that LSTM architectures with several hidden layers can progressively build up a higher level of representations of sequence data, and thus, work more effectively. In a stacked multi-layer LSTM architecture, the output of a hidden layer will be fed as the input into the subsequent hidden layer. This stacking layer mechanism, which can enhance the power of neural networks, is adopted by our proposed architecture. In this study, we propose a deep architecture named stacked bidirectional and unidirectional LSTM network architecture (SBULSTM) to predict the network-wide traffic speed values. The proposed architecture does not have a fixed number of layers or use fixed types of RNNs. Instead, this architecture, possibly containing multiple layers of LSTM or BDLSTM components, can be flexible for solving different tasks.

As mentioned in previous sections, BDLSTMs can make use of both forward and backward dependencies. When feeding the spatial-temporal information of the traffic network to the BDLSTMs, both the spatial correlation of the speeds in different locations and the temporal dependencies among the traffic state sequences can be captured during the feature learning process. In this regard, a BDLSTM layer is suitable for being the first feature learning layer of a stacked architecture for network-wide traffic prediction. Meanwhile, if the input data contains missing values, a BDLSTM-I layer with the ability to deal with missing values will be used instead.

For the proposed stacked architecture, the stacking/following layers could be LSTM instead of BDLSTM. Since BDLSTM contains more learnable parameters, the architecture of stacked BDLSTMs has the potential to perform better. Hence, the proposed SBU-LSTM contains a BDLSTM-I layer as the first feature-learning layer with the capability of imputing missing values. For the sake of making full use of the input data and learning complex and comprehensive features, in a SBU-LSTM architecture, the BDLSTM-I layer can be optionally stacked with one or more LSTM/BDLSTM layers.

The SBU-LSTM takes the sequence data as the input. The output $\hat{x}_{T+1}$ is generated by the last layer of SBU-LSTM. If the dataset contains missing values, the first layer of SBU-LSTM should be a BDLSTM-I. Then, the SBU-LSTM can bring the imputation errors from the BDLSTM-I layer into its loss function as a regularization term:

$$\mathcal{L} = \text{Loss}(\hat{x}_{T+1} - x_{T+1}) + \lambda \sum_{t=1}^{T} \sum_{m_t^d \neq 0} \frac{1}{2}(|x_t^d - \overrightarrow{\tilde{x}_t^d}| + |x_t^d - \overleftarrow{\tilde{x}_t^d}|) \tag{6.22}$$

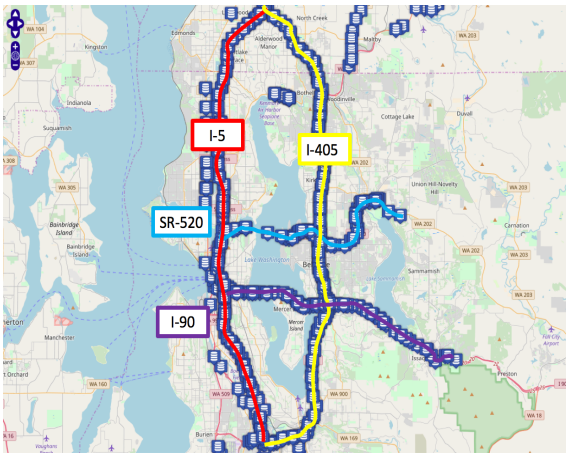
6.4 EXPERIMENTS

6.4.1 *Dataset*

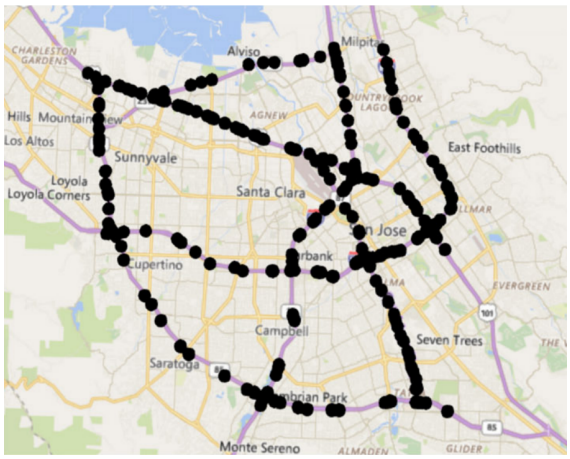
In this study, two real-world network-scale traffic state datasets are used for testing models.

6.4.1.1 *LOOP-SEA Dataset*

The dataset named as LOOP-SEA is collected by inductive loop detectors deployed on 4 connected freeways (I-5, I-405, I-90, and SR-520) in the greater Seattle area, as shown in Figure 6.3a. This



(a) LOOP-SEA dataset



(b) PEMS-BAY dataset

Figure 6.3: Datasets. (1) Loop detector data in Seattle area. (2) Loop detector data in Bay area.

dataset contains traffic speed data of 323 sensor stations ($D = 323$). This dataset covers the whole year of 2015 and the time interval is 5 minutes. Thus, this dataset has $12(5 - minutes) \times 24(hour) \times 365(days) = 105120$ time steps in total. For this traffic forecasting problem, if we suppose the length of the input sequence is 10 (T=10), the dataset contains $(105120 - 10)$ samples in total. The dataset is published on GitHub [1] (Cui et al., 2016) and Zenodo (Wang et al., 2019b).

6.4.1.2 *PEMS-BAY Dataset*

This dataset named as PEMS-BAY is collected by California Transportation Agencies (CalTrans) Performance Measurement System (PeMS). This dataset contains the speed information of 325 sensor stations in the Bay Area, as shown in Figure 6.3b. The dataset covers six months ranging from Jan 1st, 2017 to Jun 30th, 2017. The interval of time steps is 5-minutes. The total number of observed traffic data points is 16,941,600. The dataset is published by (Li et al., 2018) on the Github [2].

1 https://github.com/zhiyongc/Seattle-Loop-Data
2 https://github.com/liyaguang/DCRNN

6.4.2 *Experimental Settings*

6.4.2.1 *Hardware*

In this study, the experiments were conducted on a computer with an Intel i7-7700 CPU @ 4.2GHz processor and 32GB of memory. All the neural network-based models are trained and evaluated on a single NVIDIA GeForece GTX 1080 Ti with 11GB Memory.

6.4.2.2 *Baselines*

As indicated by multiple existing studies, the classical statistical models and machine learning models cannot outperform the LSTM model for traffic forecasting. Thus, those classical statistical models, such as ARIMA (Williams and Hoel, 2003) and machine learning models, such as support vector regression (Wu et al., 2004) and random forest (Zarei et al., 2013), are not compared in this study.

The compared models tested on datasets without missing values include LSTM, BDLSTM, and multiple combinations of LSTM and BDLSTM. The baseline models tested on datasets with missing values include Bayesian Gaussian CANDECOMP/PARAFAC decomposition(BGCP) (Chen et al., 2019b), gated recurrent unit RNN with a decay mechanism (GRU-D) (Che et al., 2018), LSTM, and several combinations of LSTM-I and BDLSTM-I.

6.4.2.3 *Parameters*

The neural network models are implemented by PyTorch 1.0.1. In the training process, we use the mini-batch training strategy. The input of the forecasting models is a 3-D vector $\mathbf{X} \in \mathbb{R}^{b \times T \times D}$. The batch size b is set as 64 and D is the number of sensors depending on the specific dataset. The length of input sequence T is set as 10, which is within a reasonable range according to (Lv et al., 2015; Yu et al., 2017b). The samples are randomized and divided into the training, validation, and test set with the ratio 6:2:2. All the RNN-based models are trained by minimizing the mean square error (MSE) using the Adam optimization method (Kingma and Ba, 2014). The early stopping mechanism is used to avoid over-fitting. If the model improvement, i.e. the decrease of the validation loss, cannot exceed a threshold, set as 0.00001 (MSE), in 5 consecutive epochs,

the training process will be terminated. We also design a learning rate decay mechanism for the training process to speed up the models' convergence. The initial learning rate of all models is set as 10^{-3}. If the model improvement cannot surpass the threshold, the learning rate will reduce an order of magnitude until it reaches 10^{-5}.

6.4.2.4 *Missing Scenarios*

When forecasting models are evaluated based on traffic state data with missing values, both the amount and the distribution of missing values will affect the prediction performance. Hence, we create a **random** scenario and a **non-random** scenario to generate datasets with different missing patterns according to (Chen et al., 2019a). The random scenario is created by randomly setting a specific proportion of values in the input as zeroes. The non-random scenario is created by randomly setting the values at a specific proportion of time steps as zeroes. The masking vectors for the two scenarios can be generated accordingly. For generating datasets with different amounts of missing values, datasets with 10%, 20%, 40%, and 80% missing values are created and tested in this study. When generating datasets with missing values, we use the identical random seed to ensure all models are evaluated on the identical datasets.

6.4.2.5 *Evaluation*

To measure the effectiveness of different traffic state prediction algorithms, widely used traffic prediction metrics (Li and Shahabi, 2018), including Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE), are computed using the following equations:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i| \tag{6.23}$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}|\frac{x_i - \hat{x}_i}{x_i}| \tag{6.24}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}|x_i - \hat{x}_i|^2}{n}} \tag{6.25}$$

where x_i is the observed traffic speed, and $\hat{x}_i$ is the predicted speed.

Table 6.1: Performance of RNN-Based models for network-wide traffic speed prediction on LOOP-SEA dataset

Models	Performance of Models on LOOP-SEA Dataset											
	N=0			N=1			N=2			N=3		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
N+1 LSTM	3.769	11.021	6.106	2.389	5.681	3.562	2.417	5.800	3.634	2.643	6.606	4.052
N+1 BDLSTM	3.027	6.815	5.265	2.336	5.475	3.507	2.405	3.631	5.723	2.472	5.947	3.750
N BDLSTM + LSTM	-	-	-	2.523	6.153	3.809	2.464	5.954	3.707	2.579	6.344	3.911
N LSTM + BDLSTM	-	-	-	2.362	5.552	3.542	2.448	5.875	3.707	2.580	6.317	3.941

Table 6.2: Performance of RNN-Based models for network-wide traffic speed prediction on PEMS-BAY dataset

Models	Performance of Models on PEMS-BAY Dataset											
	N=0			N=1			N=2			N=3		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
N+1 LSTM	3.286	6.530	4.914	2.315	3.989	3.085	2.363	4.131	3.198	5.444	13.656	9.185
N+1 BDLSTM	1.659	3.003	4.295	1.186	2.251	1.927	1.337	2.583	2.252	1.569	3.142	2.822
N BDLSTM + LSTM	-	-	-	2.509	4.520	3.476	2.398	4.223	3.252	5.618	13.831	9.192
N LSTM + BDLSTM	-	-	-	1.333	2.545	2.161	1.526	3.023	2.671	2.532	4.732	6.223

6.4.3 *Experimental Results*

In this section, the evaluation results of stacked and bidirectional LSTM-based models tested on the LOOP-SEA and PEMS-BAY datasets are shown in Table 6.1 and 6.2, respectively. The "N DBLSTM + LSTM" refers to a n-layer BDLSTMs with an LSTM layer as the last layer. The "N LSTM + BDLSTM" is named in the similar way.

From the experimental results shown in the two tables, we can observe at least three main similar patterns. Firstly, compared with multi-layer LSTMs and BDLSTMs, the one-layer models performs worst, which indicates the stacking mechanism can improve the prediction performance. Among the multi-layer models, the two-layer models outperforms the models with more layers. The prediction performance decreases along with the increase of amount of layers. The two-layer BDLSTM achieves the minimum MAEs of 2.336 and 1.186 on the LOOP-SEA and PEMS-BAY datasets, respectively. The second main finding is that the BDLSTM model with a specific number of layers performs better than the LSTM model with the same number of layers. This phenomenon is particularly evident on the results of the PEMS-BAY data. The third finding is that the BDLSTM is more suitable than the LSTM for being the last layer of a model. Although the "N BDLSTM + LSTM" has more parameters than the "N LSTM + BDLSTM", when these models has same amount of layers, the "N LSTM + BDLSTM" achieves better prediction performance.

The differences between the experimental results on two datasets are also obvious. The BDLSTM and the "N LSTM + BDLSTM" can achieve much better performance than other types of models on the PEMS-BAY dataset. However, the superiority of BDLSTM-based models tested on the LOOP-SEA dataset is not as evident as that on the PEMS-BAY dataset. This phenomenon may be lead by that the traffic state sequences in the LOOP-SEA dataset contain more Irregular variations.

6.4.4 *Training Time*

Figure 6.4a shows the training time per epoch of the compared models tested on the LOOP-SEA dataset. Considering the length of the input time series is fixed, the training time of a model is mostly related to the amount of parameters. Since the BDLSTM contains two LSTMs, the training

Table 6.3: Performance comparison for the SBU-LSTM with different numbers of model parameters

Dimensions of weight matrices in a two-layer BDLSTM	**LOOP-SEA**			**PEMS-BAY**		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
$\lceil 1/4D \rceil$	2.457	5.896	3.698	1.438	2.788	2.358
$\lceil 1/2D \rceil$	2.383	5.643	3.578	1.250	2.375	2.028
D	2.336	3.507	3.507	1.186	2.251	1.927
2 D	2.324	5.443	3.483	1.124	2.127	1.824
4 D	2.324	5.436	3.489	1.099	2.076	1.791

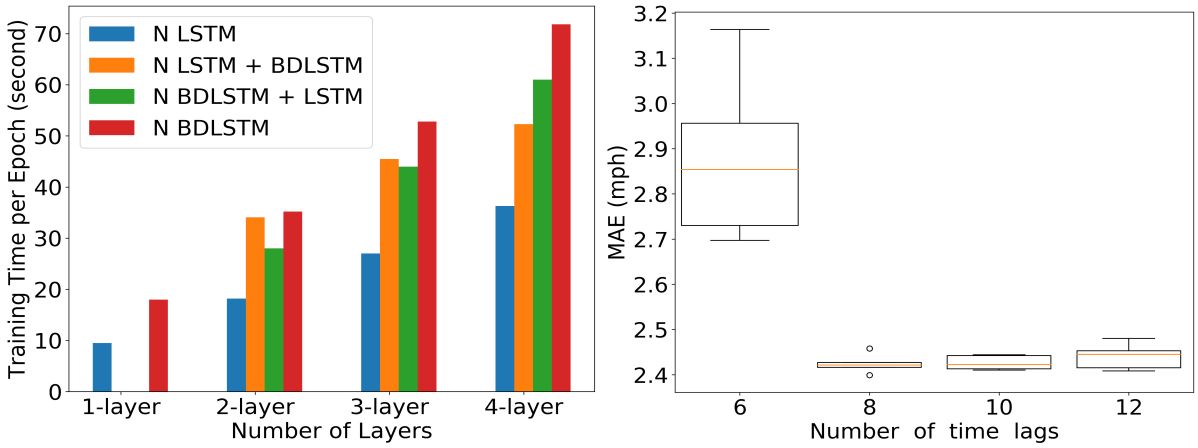
time of BDLSTM is nearly double of that of LSTM. The training times of those multi-layer models are nearly linearly related to the number of layers. Since the "N BDLSTM + LSTM" and the "N LSTM + BDLSTM" are the combinations of the BDLSTM and the LSTM, their training times are between those of multi-layer BDLSTMs and LSTMs.

6.4.5 *Influential Factors of the RNN-based Model*

Two factors that influence the prediction performance, including the size of the model weights and the amount of time lags of the input sequences, are measured in this section.

6.4.5.1 *Influence of the size of model weights*

Since the spatial dimension of prediction output should be same as the spatial dimension of the input sequence's (D), the weight matrices in one-layer LSTM or BDLSTM, i.e. the W and U in Equations 6.3 to 6.6, should be with the dimension of $D \times D$. For a multi-layer LSTM or BDLSTM, the dimension of the weight matrices in each layer can be customized, except for the first dimension of the weight matrices in the first layer and the second dimension of the weight matrices in the last layer that should be D. Hence, in this section, we change the dimensions of the customizable weight matrices in a two-layer BDLSTM to measure the influence of the size of model weights on the prediction performance. The customized dimensions include $\lceil 1/4D \rceil$,



(a) (b)

Figure 6.4: (a) Training time per epoch of the compared models tested on the LOOP-SEA dataset. (b) Boxplot of MAE versus number of time lags. The MAEs are generated by the BDLSTM+LSTM model tested on the LOOP-SEA dataset. The unit of one time lag (time step) is 5 minutes.

$\lceil 1/2D \rceil$, D, $2D$, and $4D$. The experimental results generated by the two-layer BDLSTM tested on both LOOP-SEA and PEMS-BAY datasets are shown in Table 6.3. The results indicate that the more parameters the model contains, the more accurate the prediction results are. When the customized dimension reduces to $\frac{1}{4}D$, the prediction performance on both datasets obviously decreases. When the customized dimension increase to $4D$, the model achieves best prediction accuracy on the PEMS-BAY dataset. However, the prediction performance cannot improve much on the LOOP-SEA dataset. This phenomenon shows that for a specific type of models, increasing the amount of parameters can improve the model's prediction capability to some extent. However, the prediction accuracy will not keep improving along with the increase of the amount of model parameters.

6.4.5.2 *Influence of the length of the input sequences*

The length of the input sequences has an influence on the short-term traffic forecasting performance. Figure 6.4b shows the boxplot of the MAE of a BDLSTM+LSTM model versus the length of the input sequence tested on the LOOP-SEA dataset. When the length equals 8, 10, and 12, the MAEs of the predictions are very close and the deviations of these MAEs are relatively small. When the number of time lags is set as 6, the MAE is much higher, and the deviation is much

larger than that of other cases. That means, given the 5-minutes time interval and the studied traffic network, 6 steps of traffic sequence data are not long enough for modeling and predicting network-wide traffic states accurately. In summary, the number of time lags tends to influence the predictive performance, especially when the input sequences is relatively short.

6.4.6 *Dealing with Missing Values*

In this section, to evaluate the effectiveness of the proposed imputation unit in the LSTM-based structures, we compared the prediction performance of GRU-D (Che et al., 2018), LSTM-I, and BDLSTM-I. Since the previous sections show that two-layer LSTM- or BDLSTM-based two-layer models have better prediction performance, the LSTM-I and BDLSTM-I stacked with LSTM or BDLSTM is also compared. Besides, although the proposed imputation unit in LSTM-I is designed for doing data imputation tasks, we still attempt to compare the data imputation performance of the proposed models with a state-of-the-art data imputation model, i.e. the Bayesian Gaussian CANDECOMP/PARAFAC (BGCP) tensor decomposition model (Chen et al., 2019b). The parameters of the BGCP is identical to the parameter settings in (Chen et al., 2019b).

6.4.6.1 *Comparison with Forecasting Methods*

In this section, the compared models are tested on the LOOP-SEA and PEMS-BAY datasets with different missing scenarios, including random and non-random scenarios, and different missing rates ranging from 10% to 80%. The experimental results tested on the two datasets with random missing value are shown in Table 6.4 and 6.5, and the results tested on datasets with the non-random missing value scenario are shown in Table 6.6 and 6.7, respectively.

When the missing rate is relatively small (10% and 20%), the experimental results indicate that the GRU-D model cannot outperform other compared models in the random missing scenarios. The LSTM-I also cannot deal with the missing values very well, especially in the non-random missing scenarios. However, the bidirectional and stacked LSTM-models achieve better prediction accuracy on the LOOP-SEA datasets. Further, experimental results on the PEMS-BAY datasets show that the models with a BDLSTM as the last layer perform better no matter in the random

or non-random missing scenario. Among the two-layer models, the prediction results of the BDLSTM-I + BDLSTM obviously outperforms those of other compared models.

When the missing rate is relatively large (40% and 80%), the one-layer models, including the GRU-D, LSTM-I, and BDLSTM-I, cannot compete with the two-layer models. In all cases, the two-layer models with a BDLSTM second layer perform better than those with an LSTM second layer. The BDLSTM-I + BDLSTM model achieves the smallest prediction errors.

Overall, the prediction results in the non-random scenario are close to or slightly better than those in the random scenario. The prediction results on the LOOP-SEA dataset have larger deviations than those on the PEMS-BAY dataset. The models containing a BDLSTM-I layer slightly outperform the models with an LSTM-I layer. The experimental results also indicate that the BDLSTM is appropriate to be the last layer of a model, compared with LSTM. Hence, the BDLSTM-I + BDLSTM model outperforms other models, which is particularly obvious on the PEMS-BAY dataset.

Table 6.4: Prediction results on LOOP-SEA dataset with Random missing values

Multi-layer Models	**LOOP-SEA** dataset with **Random** missing values											
	Missing Rate = 10 %			Missing Rate = 20 %			Missing Rate = 40 %			Missing Rate = 80 %		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
GRU-D	3.498	10.354	5.411	3.676	10.880	5.881	3.973	11.760	6.032	4.539	13.480	7.292
LSTM-I	3.989	11.830	6.443	4.060	12.053	6.548	4.226	12.715	6.812	4.791	14.89	7.697
LSTM-I + LSTM	2.574	6.290	3.862	2.675	6.642	4.037	7.709	30.657	12.339	8.836	30.178	12.341
LSTM-I + BDLSTM	2.639	6.494	4.035	2.927	7.355	4.541	3.539	9.913	5.864	4.516	13.963	7.481
BDLSTM-I	2.605	6.358	3.887	2.687	6.642	4.024	2.846	7.178	4.307	6.813	17.31	13.183
BDLSTM-I + LSTM	2.888	7.471	4.391	3.010	7.969	4.618	3.415	9.632	5.442	4.546	13.917	7.368
BDLSTM-I + BDLSTM	2.768	6.920	4.208	2.843	7.200	4.384	3.078	8.149	4.878	3.675	10.655	6.073

6.4.6.2 *Comparison with Data Imputation Methods*

As mentioned before, values generated by the proposed imputation unit ($\tilde{x}_t$) may not be the "actual" missing values. However, comparing data imputation performance of the proposed model with other data imputation models can still be informative. In this section, we compare the imputation

Table 6.5: Prediction results on PEMS-BAY dataset with Random missing values

Multi-layer Models	**PEMS-BAY** dataset with **Random** missing values											
	Missing Rate = 10 %			Missing Rate = 20 %			Missing Rate = 40 %			Missing Rate = 80 %		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
GRU-D	5.320	13.584	9.163	5.347	13.609	9.160	5.387	13.67	9.180	5.739	13.767	9.193
LSTM-I	3.486	7.137	5.330	3.550	7.321	5.460	3.704	7.737	5.746	4.141	8.959	6.544
LSTM-I + LSTM	2.511	4.536	3.435	2.746	5.178	3.877	5.628	14.027	9.303	5.641	14.056	9.300
LSTM-I + BDLSTM	1.566	3.054	2.599	1.621	3.155	2.653	2.349	5.122	4.365	3.111	7.189	5.943
BDLSTM-I	1.446	2.820	2.341	1.582	3.126	2.596	2.378	6.377	4.654	8.939	16.202	17.746
BDLSTM-I + LSTM	2.664	4.963	3.724	2.954	5.729	4.322	3.229	6.584	4.990	3.804	8.217	6.111
BDLSTM-I + BDLSTM	1.697	3.391	2.830	1.810	3.710	3.186	2.066	4.385	3.772	2.451	5.365	4.566

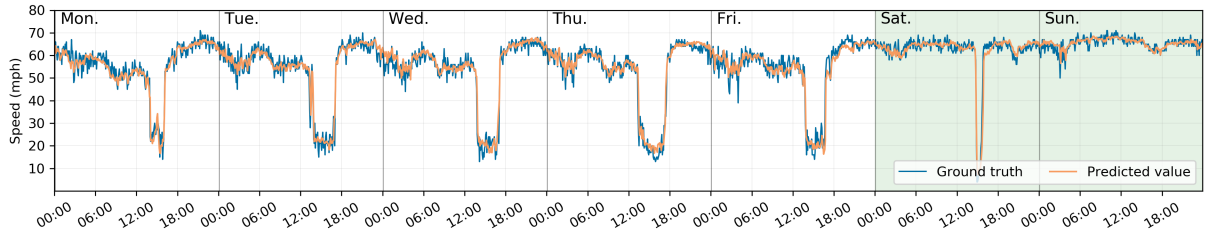
Table 6.6: Prediction results on LOOP-SEA dataset with non-random missing values

Multi-layer Models	**LOOP-SEA** dataset with **Non-Random** missing values											
	Missing Rate = 10 %			Missing Rate = 20 %			Missing Rate = 40 %			Missing Rate = 80 %		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
GRU-D	3.732	10.876	5.757	3.870	11.023	6.124	4.123	11.643	6.459	4.903	14.194	8.149
LSTM-I	4.374	13.620	7.177	4.810	15.410	7.881	5.712	19.412	9.212	7.349	26.585	11.263
LSTM-I + LSTM	2.653	6.643	4.062	2.788	7.120	4.310	3.048	8.143	4.785	4.327	13.573	7.112
LSTM-I + BDLSTM	2.665	6.674	4.041	2.728	6.921	4.151	2.898	7.650	4.467	3.864	11.829	6.458
BDLSTM-I	3.136	8.446	5.328	3.612	10.391	6.202	4.772	15.167	8.144	8.005	26.965	12.296
BDLSTM-I + LSTM	2.828	7.277	4.342	2.981	7.855	4.618	3.277	9.123	5.207	4.742	15.193	7.631
BDLSTM-I + BDLSTM	2.504	6.085	3.770	2.577	6.328	3.903	2.764	6.995	4.248	3.712	11.034	6.244

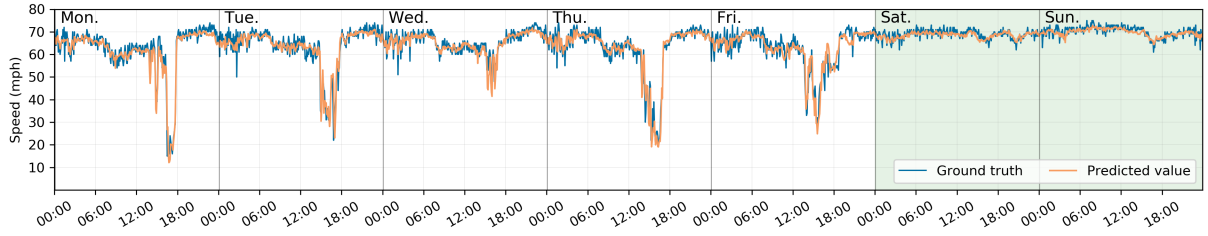
Table 6.7: Prediction results on PEMS-BAY dataset with non-random missing values

Multi-layer Models	**PEMS-BAY** dataset with **Non-Random** missing values											
	Missing Rate = 10 %			Missing Rate = 20 %			Missing Rate = 40 %			Missing Rate = 80 %		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
GRU-D	5.103	13.049	9.162	5.657	13.470	9.568	5.357	13.918	9.028	5.124	13.145	9.881
LSTM-I	3.676	7.920	6.010	3.887	8.603	6.469	4.350	10.067	7.362	5.192	12.846	8.799
LSTM-I + LSTM	2.971	5.783	4.393	2.847	5.441	4.137	2.891	5.569	4.241	5.261	12.784	8.677
LSTM-I + BDLSTM	1.787	3.703	3.184	1.921	4.082	3.504	2.058	4.520	3.834	3.097	7.244	5.698
BDLSTM-I	1.742	3.674	3.667	2.134	4.803	4.601	3.010	7.178	6.140	6.730	14.829	12.646
BDLSTM-I + LSTM	3.064	6.091	4.636	2.948	5.751	4.387	3.187	6.384	4.807	3.470	7.287	5.547
BDLSTM-I + BDLSTM	1.560	3.119	2.665	1.716	3.505	3.004	1.951	4.102	3.530	2.764	6.271	5.241

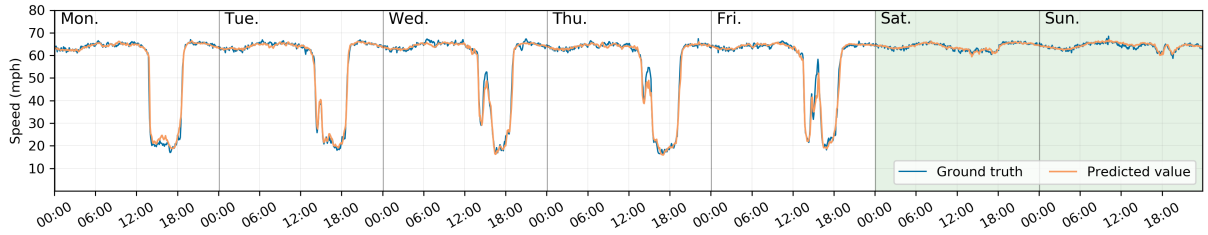
performance of BDLSTM-I+BDLSTM and the BGCP model. The experimental results tested on both LOOP-SEA and PEMS-BAY datasets are presented in Table 6.8. The proposed method outperforms the BGCP when the missing rate is relatively small. The imputation performance of the BDLSTM-based model decreases along with the increase of the missing rate. However, the imputation errors of the BGCP model nearly keep the same when the missing rate varies. This is reasonable because the data imputation mechanism of BGCP, which is a tensor decomposition-based model, greatly differs from that of deep learning models. The data imputation mechanism of a trained LSTM-I only uses a short-term sequence data as input data. However, tensor decomposition-based methods use the whole datasets to impute missing data. Besides, tensor decomposition-based models use different types of optimization methods and have more optimization iterations. In summary, although the proposed method is not designed for solving data imputation tasks and uses much fewer data, it can achieve similar imputation performance as the BGCP model. This experimental results imply that the proposed imputation unit in LSTM-I/BDLSTM-I can indirectly contribute to the traffic prediction task.



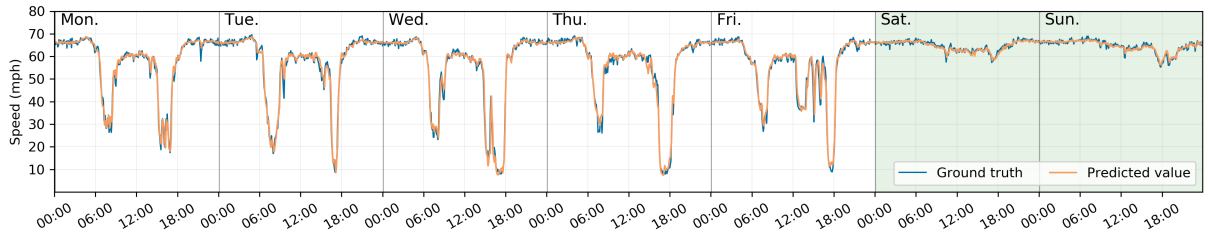
(a)



(b)



(c)



(d)

Figure 6.5: Visualized ground truth and predicted traffic states. Four sites are selected from the LOOP-SEA and PEMS-BAY datasets. The first two are from the LOOP-SEA dataset during the second week in 2015 and the last two are from the PEMS-BAY dataset during the fifth week in 2012. (a) Sensor ID: d005es15214. (b) Sensor ID: d005es15608. (c) Sensor ID: 400017. (d) Sensor ID: 400057.

Table 6.8: Data imputation performance comparison

Datasets & Models		Missing Rate = 10 %			Missing Rate = 20 %			Missing Rate = 40 %			Missing Rate = 80 %		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
LOOP-SEA	BDLSTM-I + BDLSTM	3.676	7.920	6.010	3.887	8.603	6.469	4.350	10.067	7.362	5.192	12.846	8.799
	BGCP	3.764	11.230	5.991	3.757	11.221	5.981	3.774	11.280	6.001	3.763	11.228	5.991
PEMS-BAY	BDLSTM-I + BDLSTM	1.742	3.674	3.667	2.134	4.803	4.601	3.010	7.178	6.140	6.730	14.829	12.646
	BGCP	2.131	4.692	3.969	2.140	4.704	3.971	2.139	4.710	3.988	2.131	4.698	3.981

6.4.7 *Model Interpretation and Visualization*

In this section, we take several specific regions as examples to conduct experimental validation and visualization. Figure 6.5a and 6.5b show the ground truth and predicted traffic states of two sensing locations selected from the LOOP-SEA dataset. Figure 6.5c and 6.5d show the results of two sensing locations selected from the PEMS-BAY dataset. The traffic states in the LOOP-SEA dataset have more variations and the traffic states in the PEMS-BAY dataset varies smoothly. As shown by those figures, different traffic patterns on weekdays and weekends and the various traffic states during the peak hours can be accurately predicted.

6.5 CHAPTER SUMMARY

In this Chapter, we attempt to reformulate the way to incorporate LSTM into traffic prediction models. A stacked bidirectional and unidirectional LSTM network architecture is proposed for network-wide traffic state prediction. Experimental results show that the stacked bidirectional LSTM models achieve the superior prediction performance. In addition, the evaluation of the prediction capability of multiple stacked LSTM- or BDLSTM-based models has great potential to facilitate the design of neural network models for traffic prediction problems.

We also proposed an imputation unit in the LSTM model, which is designed to handle missing values. The LSTM- or BDLSTM-based models with the imputation unit can infer and fill the missing values in the spatial-temporal input data and in return to help improve prediction accuracy. Experimental results indicate that the proposed models with the imputation unit can

outperform the state-of-the-art RNN based models and compete with the tensor decomposition based models. Further, the trade-off between model capacity and complexity and the influential factors of the proposed model are evaluated and discussed. In addition, two real-world traffic state data is tested in this study and the LOOP-SEA dataset is published on public accessible repositories to facilitate further research in this field.

In the future, we will carry out more in-depth studies using different datasets. Further improvements and extensions may focus on improving the model to better interpret spatial features and fusing traffic prediction with other applications. Potential applications, like non-recurring congestion detection, will be explored by combining other datasets. [3]

3 This chapter is a slightly modified version of "**Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Forecasting Network-wide Traffic State with Missing Values**" published in *Transportation Research Part C: Emerging Technologies* and has been reproduced here with permission from the copyright holder. (doi: https://doi.org/10.1016/j.trc.2020.102674)

7

GRAPH MARKOV NETWORK FOR TRAFFIC FORECASTING WITH MISSING VALUES

7.1 OVERVIEW

Traffic forecasting, as a challenging topic for both academia and industry, has been under active research, development, and implementation for more than 40 years (Laña et al., 2018). Traffic forecasting plays an important role in transportation management and the general planning process. With the exponential increase in the volume of traffic data and the computational capability, traffic forecasting methods have been gradually shifting from classical statistical models to data-driven machine learning-based methods (Vlahogianni et al., 2014). In recent years, the rise of artificial intelligence (AI), especially deep learning methods, has dramatically stimulated the traffic forecasting research field. By leveraging the spatial-temporal patterns contained in immense data resources, many deep neural network models, including recurrent neural network (RNN), convolutional neural network (CNN), generative adversarial network (GAN), etc., have been widely applied in traffic forecasting studies and achieved state-of-the-art prediction performance.

However, since network-wide traffic state data are mostly collected by traffic sensors or probe vehicles, sensor failures or irregular sampling from probe vehicles will result in missing values in the collected data. The missing value issue usually leads to an apparent decline in the forecasting performance, as most of the existing methods for traffic forecasting are not capable of dealing with missing values. Thus, forecasting performance of the models that only accept valid data as input will be significantly affected and limited.

The regular solution for the missing data issue is to conduct data imputation, which targets to estimate the corrupted or missing traffic data. Due to the complex spatial-temporal patterns of traffic data,existing novel and effective data imputation methods, such as the tensor decomposition-based approaches (Chen et al., 2019b; Tan et al., 2016), usually need large datasets covering a long period of time to achieve good imputation performance. However, a large dataset covering a long period of time is not always available. Further, in the connected vehicle environments or under the edge computing scenarios, online forecasting computations need to be completed in devices with limited storage and computational capabilities. However, those tensor decomposition-based model might require hundreds of iterations to converge, which might take tens of minutes, and achieve their best imputation performance (Chen et al., 2019b; Tan et al., 2016). Thus, when processing large datasets, these solutions might not be feasible for real-time traffic forecasting tasks.

To solve the missing value issue and fulfill traffic forecasting at the same time, traffic forecasting models with the capability of dealing with missing values have also been proposed. However, most of the existing models (Lee and Fambro, 1999; Zhang and Zhang, 2016), take the spatial-temporal traffic data as multivariate time series, and thus, they neglect the important spatial influence between the road links in the traffic network. There are several deep learning-based methods (Duan et al., 2016a; Tian et al., 2018) taking spatial factors into consideration. However, they still cannot incorporate the intrinsic structure of the traffic network into the traffic forecasting process.

In this study, to overcome the problems mentioned above, we propose a graph Markov network (GMN), which is a new neural network architecture for spatial-temporal data forecasting with missing values. The traffic network is converted into a graph with topological properties. We consider the variations of the traffic states in the traffic network has Markov property and graph localization property. Based on the two properties, the traffic state transition process can be considered as a graph Markov process. The GMN is designed based on the graph Markov process, which inherently incorporates the spatial-temporal relationships among the links in the traffic network. By incorporating the spectral graph convolution operation, we also propose a spectral graph Markov network (SGMN). Experimental results indicate that the proposed SGMN and GMN can achieve superior prediction performance with greater efficiency.

7.1.1 *Contribution and Organization of the Chapter*

The contributions of this study can be summarized as follows:

1. We consider the traffic network as a graph and define the transition between network-wide traffic states at consecutive time steps as a graph Markov process.
2. We propose a new neural network structure, i.e. the graph Markov network, based on the proposed graph Markov process for dealing with missing values and forecasting traffic state simultaneously.
3. By incorporating the spectral graph convolution operation, we also propose a spectral graph Markov network.
4. Experiment results tested on three real-world network-wide traffic state datasets show that the proposed models can achieve superior prediction performance in terms of both accuracy and efficiency.

The rest of this paper is organized as follows: Section 7.2 describes the related studies on deep learning-based traffic forecasting and related work with the capability of dealing with missing values. Before introducing the models, we present several related notions in Section 7.3. Section 7.4 introduces the proposed graph Markov process and the corresponding proposed Graph Markov Network model. Section 7.5 discusses the experimental results, and the concluding remarks are presented in Section 7.6.

7.2 RELATED WORK

Classical traffic forecasting models can generally be classified into two categories, traditional statistical models and computational intelligence, i.e. machine learning-based, models (Vlahogianni et al., 2014). The statistical methods are mostly parametric approaches, including variants of auto-regressive integrated moving average (ARIMA) models (Williams, 2001), parametric Kalman filtering models (Okutani and Stephanedes, 1984), and other types of time-series models (Ghosh et al., 2009), that are developed based on a predefined model structure with theoretical assumptions

and the parameters are calibrated using historical data (Smith et al., 2002). With the ability to accommodate the stochastic and non-linear nature of traffic patterns, classical machine learning methods are widely adopted for the traffic forecasting task, such as support vector regression (Wu et al., 2004), Bayesian network approaches (Sun et al., 2006). In recent years, with the rise of AI, the performance of emerging deep learning-based traffic forecasting methods outperform that of classical methods.

7.2.1 *Deep learning-based traffic forecasting methods*

Since the traffic data contain both spatial and temporal attributes, the deep learning-based methods can be grouped by the ways to deal with spatial-temporal traffic data. One type of studies convert the spatial-temporal data into a 2-dimensional (2D) matrix and use long short-term memory (LSTM) recurrent neural network (Ma et al., 2015), bi-directional LSTM (Cui et al., 2017), CNN (Ma et al., 2017), GAN (Liang et al., 2018), or a combination of multiple models (Yang et al., 2019), to extract feature and forecast traffic states. However, a traffic network's spatial features cannot be completely represented by a 2D matrix. Thus, another type of methods (Ma et al., 2020; Yu et al., 2017a) is proposed to convert the physical roadway networks as images according to roads' geospatial properties. Although the traffic network images demonstrate the true traffic network structure, those images contain too many noisy pixels and blank pixels without traffic state information. To analyze the traffic network in an efficient way, many studies consider the traffic network as a graph and predict traffic state by incorporating the graph convolutional network (Cui et al., 2019; Li et al., 2018; Yu et al., 2018).

7.2.2 *Deep learning-based traffic forecasting with missing values*

Traffic forecasting performance will be influenced by the missing values (Cui et al., 2017). A bunch of data imputation methods has been developed to solve the missing values issues, including the probabilistic principal component analysis (PCA) (Li et al., 2014), tensor decomposition-based methods (Chen et al., 2019b, 2018b; Ran et al., 2016; Tan et al., 2016), clustering approaches (Ku

et al., 2016; Tang et al., 2015). There are also some deep learning-based data imputation methods proposed in the most recent years, such as denoising stacked auto-encoder (Duan et al., 2016a) and generative adversarial imputation network (Yoon et al., 2018). However, the PCA-based (Li et al., 2014) and tensor decomposition-based (Chen et al., 2019b, 2018b; Ran et al., 2016; Tan et al., 2016) models normally need hundreds or thousands of iterations to converge and achieve their best data imputation performance. Further, the aforementioned deep learning models for data imputation are not originally designed for solving the traffic forecasting with missing values issue.

To combine the data imputation and traffic forecasting together, a few RNN-based approaches, such as the LSTM-M (Tian et al., 2018), have been proposed based the GRU-D (Che et al., 2018) for processing multivariate time series with missing values. Even though these RNN-based methods can recurrently fill missing values in each time step and forecasting the future traffic state, they cannot capture spatial interactions between road links in the traffic network. Further, although recent research (Barredo-Arrieta et al., 2019; Du et al., 2019) is trying to interpret RNN-based models, for the traffic forecasting problem, it is still hard for RNN-based models to interpret the spatial relationship between neighboring links and the links' temporal dependencies between different time steps.

To solve these problems, in this study, we consider the traffic state transition process as a graph Markov process and propose the graph Markov network for the traffic forecasting with missing values. The design of the graph Markov network inherently incorporates the spatial-temporal relationships among the links in the traffic network. The graph Markov network making use of the topological structure of the traffic network can achieve accurate prediction results with efficient training and testing process. The proposed graph Markov process and graph Markov network are introduced in detail in the following section.

7.3 PRELIMINARIES

7.3.1 *Traffic Forecasting*

A traffic network normally consists of multiple roadway links. The traffic forecasting task targets to predict future traffic states of all (road) links or sensor stations in the traffic network based on

historical traffic state data. The collected spatial-temporal traffic state data of a traffic network with S links/sensor-stations can be characterized as a T-step sequence $[x_1, x_2, ..., x_t, ..., x_T] \in \mathbb{R}^{T \times S}$, in which $x_t \in \mathbb{R}^S$ demonstrates the traffic states of all S links at the t-th step. The traffic state of the s-th link at time t is represented by x_t^s. In this study, the superscript of a traffic state represents the spatial dimension and the subscript denotes the temporal dimension. The short-term traffic forecasting problem can be formulated as, based on T-step historical traffic state data, learning a function $F(\cdot)$ to generate the traffic state at next time step as follows:

$$F([x_1, x_2..., x_T]) = [x_{T+1}] \tag{7.1}$$

7.3.2 *Graph Representations of Traffic Network*

Since the traffic network is composed of road links and intersections, it is intuitive to consider the traffic network as an undirected graph consisting of vertices and edges. The graph can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{D})$ with a set of vertices $\mathcal{V} = \{v_1, ..., v_S\}$ and a set of edges $\mathcal{E}$ between vertices. $\mathcal{A} \in \mathbb{R}^{S \times S}$ is a symmetric (typically sparse) adjacency matrix with binary elements, where $\mathcal{A}_{i,j}$ denotes the connectedness between nodes v_i and v_j. The existence of an edge is represented through $\mathcal{A}_{i,j} = \mathcal{A}_{j,i} = 1$, otherwise $\mathcal{A}_{i,j} = 0$ ($\mathcal{A}_{i,i} = 0$). Based on $\mathcal{A}$, a diagonal graph degree matrix $\mathcal{D} \in \mathbb{R}^{S \times S}$ describing the number of edges attached to each vertex can be obtained by $\mathcal{D}_{i,i} = \sum_{j=1}^{S} \mathcal{A}_{i,j}$.

The $\mathcal{A}$ can only indicate the relationship between different vertices. In some cases, the vertices' relationship with themselves also needs to be characterized. Thus, we define the self-connection adjacency matrix $\mathbf{A} = \mathcal{A} + I$, i.e. $\mathbf{A}_{i,i} = 1$, which implies each vertex in the graph is self-connected. Here, $I \in \mathbb{R}^{S \times S}$ is an identity matrix.

In addition, the connectedness of the graph vertices can also be encoded by the Laplacian matrix, which is essential for spectral graph analysis. The combinatorial Laplacian matrix is defined as $\mathcal{L} = \mathcal{D} - \mathcal{A}$ and the normalized definition is $\mathcal{L} = I - \mathcal{D}^{-1/2}\mathcal{A}\mathcal{D}^{-1/2}$. Since $\mathcal{L}$ is a symmetric positive semi-definite matrix, it can be diagonalized as $\mathcal{L} = U\Lambda U^T$ by its eigenvector matrix U (Defferrard et al., 2016), where $U = [u_0, u_1, ..., u_{S-1}] \in \mathbb{R}^{S \times S}$ and $\Lambda = \text{diag}(\lambda_0, \lambda_1, ..., \lambda_{S-1}) \in \mathbb{R}^{S \times S}$ is the corresponding diagonal eigenvalue matrix satisfying $\mathcal{L}u_i = \lambda_i u_i$.

In this study, under the traffic forecasting scenario, the attribute on vertex v_s (road link s) at time t is denoted as x_t^s. Given the graph representation of the traffic network, the Equation 7.1 can be extended as

$$F(\mathcal{G}, [x_1, x_2..., x_T]) = [x_{T+1}] \tag{7.2}$$

7.3.3 *Traffic Forecasting with Missing Values*

Traffic state data can be collected by multi-types of traffic sensors or probe vehicles. When traffic sensors fail or no probe vehicles go through road links, the collected traffic state data may have missing values. We use a sequence of masking vectors $[m_1, m_2, ..., m_T] \in \mathbb{R}^{T \times S}$, where $m_t \in \mathbb{R}^S$, to indicate the position of the missing values in traffic state sequence $[x_1, x_2, ..., x_T]$. The masking vector can be obtained by

$$m_t^s = \begin{cases} 1, \text{ if } x_t^s \text{ is observed} \\ 0, \text{ otherwise} \end{cases} \tag{7.3}$$

where x_t^s is the traffic state of s-th link at step t.

Missing values in traffic data can be handled by many existing data imputation methods. Most state-of-the-art data imputation methods, such as the Bayesian tensor decomposition approach (Chen et al., 2019b) and the Generative Adversarial Imputation Nets (GAIN) (Yoon et al., 2018), need long-term historical data to capture complicated traffic patterns and fill missing values. However, in real-time environments, especially under the connected autonomous vehicle (CAV) and edge computing scenarios, it may not be possible to conduct data imputation on historical data and forecast future traffic states sequentially, because the volume of traffic state data is huge and the computing capability of devices is limited. In these cases, the traffic forecasting models should be able to handle missing values. Taking the missing values into consideration, we can formulate the traffic forecasting as follows

$$F(\mathcal{G}, [x_1, x_2..., x_T], [m_1, m_2..., m_T]) = [x_{T+1}] \tag{7.4}$$

In this section, we first describe several properties of traffic states. Based on that, we propose a graph Markov process to characterize the variations of traffic states. Then, we introduce the proposed graph Markov Network for traffic forecasting with the capability of dealing with missing values.

7.4.1 *Properties*

A traffic network is a dynamic system and the states on all links keep varying resulted by the movements of vehicles in the system. Thus, we assume the traffic network's dynamic process satisfies the Markov property that the future state of the traffic network is conditional on the present state.

Markov property: The future state of the traffic network x_{t+1} depends only upon the present state x_t, not on the sequence of states that preceded it. Taking $X_1, X_2, ..., X_{t+1}$ as random variables with the Markov property and $x_1, x_2, ..., x_{t+1}$ as the observed traffic states. The Markov process can be formulated in a conditional probability form as

$$Pr(X_{t+1} = x_{t+1}|X_1 = x_1, X_2 = x_2, ..., X_t = x_t) = Pr(X_{t+1} = x_{t+1}|X_t = x_t) \tag{7.5}$$

where $Pr(\cdot)$ demonstrates the probability.

However, the transition matrix is temporal dependent, since at the different time of a day, the traffic state's transition pattern should be different. Based on Equation 7.5, the transition process of traffic states can be formulated in the vector form as

$$x_{t+1} = P_t x_t \tag{7.6}$$

where $P_t \in \mathbb{R}^{S \times S}$ is the transition matrix and $(P_t)_{i,j}$ measures how much influence x_t^j has on forming the state x_{t+1}^i.

The transition process defined in Equation 7.6 does not take the time interval between x_{t+1} and x_t into consideration. We denote the time interval between two consecutive time steps of traffic states by Δt. If Δt is small enough ($\Delta t \to 0$), the traffic network's dynamic process can be

measured as a continuous process and the difference between consecutive traffic states are close to zero, i.e. $|x_{t+\Delta t} - x_t| \to 0$. However, a long time interval may result in more variations between the present and future traffic states, leading to a more complicated transition process. Since the traffic state data are normally processed into discrete data and the size of transition matrix P_t is fixed, we consider that the longer the Δt is, the lower capability of measuring the actual transition process P_t has. Thus, we multiply a decay parameter $\gamma \in (0, 1)$ in Equation 7.6 to represent the temporal impact on transition process as

$$x_{t+1} = \gamma P_t x_t \tag{7.7}$$

The transition matrix can measure the contributions made by all roadway links on a specific link, which assumes that the state of a roadway link is influenced by all links in the traffic network. However, since vehicles in the traffic network traverse connected road links one by one and traffic states of connected links are transmitted by those vehicles, the traffic state of a link will only be affected by its neighboring links during a short period of time.

Graph localization property: The traffic state of a specific link s in a traffic network is mostly influenced by localized links, i.e. the link s itself and its neighboring links, during a short period of time. The neighboring links refer to the links in the graph within a specific order of hops with respect to the link s. With the help of the graph's topological structure, the localization property in the graph can be measured based the adjacency matrix in two ways: (1) The self-connection adjacency matrix $\mathbf{A}$, describing the connectedness of vertices, can inherently indicate the localization property of all vertices in the graph. Then, the impacts of localized links can be easily measured by a weighted self-connection adjacency matrix. (2) The other way is to conduct the spectral graph convolution operation on the traffic state x_t to measure the localized impacts in the graph. The spectral graph convolution on x_t can be defined as $U\Lambda_\theta U^T x_t$ (Defferrard et al., 2016), where U is the eigenvector matrix of the Laplacian matrix $\mathcal{L}$ and Λ_θ is a learnable diagonal weight matrix.

Graph Markov Process: With the aforementioned two properties, we define the traffic state transition process as a graph Markov process (GMP). The graph Markov process can be formulated in a conditional probability form as

$$Pr(X_{t+1} = x_{t+1}^u | X_t = x_t) = Pr(X_{t+1} = x_{t+1}^u | X_t = x_t^v, v \in \mathcal{N}(u)) \tag{7.8}$$

where the superscripts u and v are the indices of graph links (road links). The $\mathcal{N}(u)$ denotes a set of one-hop neighboring links of link u and link u itself. The properties of this graph Markov process is similar to the properties of the Markov random field (Rue and Held, 2005) with temporal information. Since the influence of a road link on its neighbors is gradually spread by the vehicles traveling on this road link, a road link's one-hop neighbors are the ones directly influence it. Thus, we assume that road links are only influenced by their one-hop neighbors in the graph. Based on Equation 7.7, we can easily incorporate the graph localization properties into the traffic states' transition process by element-wise multiplying the transition matrix P_t with the self-connection adjacency matrix $\mathbf{A}$. Then, the GMP can be formulated in the vector form as

$$x_{t+1} = \gamma(\mathbf{A} \odot P_t)x_t \tag{7.9}$$

where $\odot$ is the Hadamard (element-wise) product operator that $(\mathbf{A} \odot P_t)_{ij} = \mathbf{A}_{ij} \times (P_t)_{ij}$.

The graph localization property can also be incorporated in the transition process by replacing the transition weight matrix P_t with the spectral graph convolution operation. Then, we define the spectral version of the graph Markov process (SGMP) as

$$x_{t+1} = \gamma U \Lambda_{\theta_t} U^T x_t \tag{7.10}$$

where $\Lambda_{\theta_t} \in \mathbb{R}^{S \times S}$ is a diagonal weight matrix.

7.4.2 *Handling Missing Values in the Graph Markov Process*

In this section, we theoretically introduce how to deal with the missing values in the graph Markov process.

As we assume the traffic state transition process follows the graph Markov process, the future traffic state can be inferred by the present state. If there are missing values in the present state, we can infer the missing values from previous states. We consider x_t is the observed traffic state at time t and a mask vector m_t can be acquired according to Equation 7.3. We denote the completed state by $\tilde{x}_t$, in which all missing values are filled based on historical data. Hence, the completed

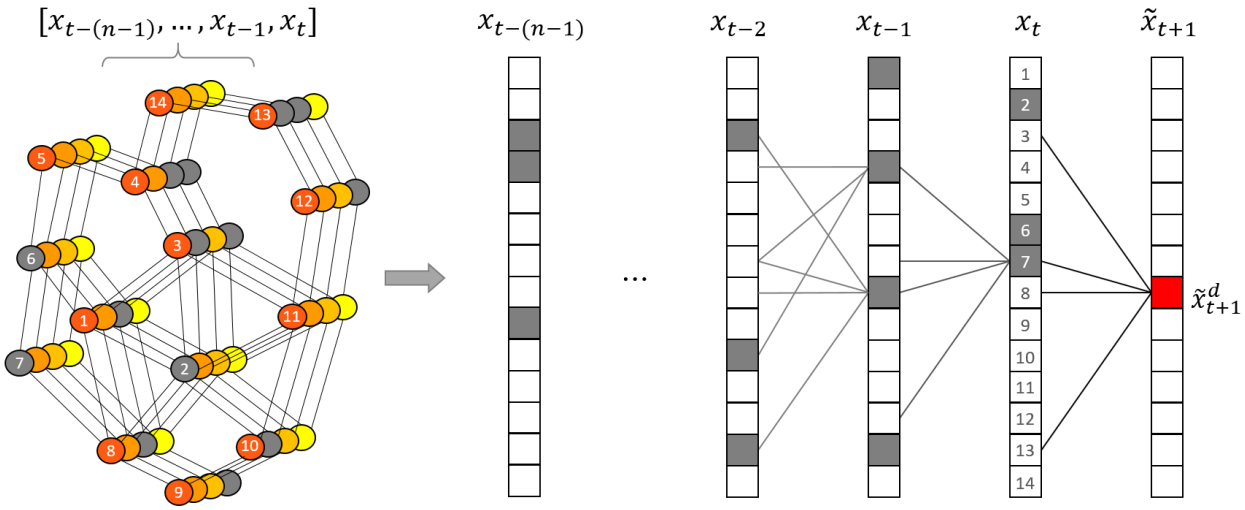


Figure 7.1: Graph Markov process. The gray-colored nodes in the left sub-figure demonstrate the nodes with missing values. Vectors on the right side represent the traffic states. The traffic states at time t are numbered to match the graph and the vector. The future state (in red color) can be inferred from their neighbors in the previous steps.

state consists of two parts, including the observed state values and the inferred state values, as follows:

$$\tilde{x}_t = x_t \odot m_t + \tilde{x}_t \odot (1 - m_t) \tag{7.11}$$

where $\tilde{x}_t \odot (1 - m_t)$ is the inferred part. Since $x_t \odot m_t = x_t$, Equation 7.11 can be written as

$$\tilde{x}_t = x_t + \tilde{x}_t \odot (1 - m_t) \tag{7.12}$$

Since the transition of completed states follows the the graph Markov process, the GMP and SGMP with respect to the completed state can be described as $\tilde{x}_{t+1} = \gamma(\mathbf{A} \odot P_t)\tilde{x}_t$ and $\tilde{x}_{t+1} = \gamma U \Lambda_{\theta_t} U^T \tilde{x}_t$, respectively. In this section, for simplicity, we denote the (spectral) graph Markov transition matrix by H_t, i.e. $H_t = \mathbf{A} \odot P_t$ or $H_t = U \Lambda_{\theta_t} U^T$. Hence, the transition process of completed states can be represented as

$$\tilde{x}_{t+1} = \gamma H_t \tilde{x}_t \tag{7.13}$$

Applying Equation 7.12, the transition process becomes

$$\tilde{x}_{t+1} = \gamma H_t (x_t + \tilde{x}_t \odot (1 - m_t)) \tag{7.14}$$

If we iteratively apply the completed state $\tilde{x}_t$, i.e. $\tilde{x}_t = \gamma H_{t-1}(x_{t-1} + \tilde{x}_{t-1} \odot (1 - m_{t-1}))$, into Equation 7.14 itself, we have

$$\begin{aligned}\tilde{x}_{t+1} &= \gamma H_t(x_t + \gamma H_{t-1}(x_{t-1} + \tilde{x}_{t-1} \odot (1 - m_{t-1})) \odot (1 - m_t)) \\ &= \gamma H_t x_t + \gamma^2 H_t H_{t-1}(x_{t-1} \odot (1 - m_t)) + \gamma^2 H_t H_{t-1}(\tilde{x}_{t-1} \odot (1 - m_{t-1}) \odot (1 - m_t))\end{aligned} \tag{7.15}$$

After iteratively applying n steps of previous states from $x_{t-(n-1)}$ to x_t, $\tilde{x}_{t+1}$ can be described as

$$\begin{aligned}\tilde{x}_{t+1} &= \gamma H_t x_t \\ &+ \gamma^2 H_t H_{t-1}(x_{t-1} \odot (1 - m_t)) \\ &+ \gamma^3 H_t H_{t-1} H_{t-2}(x_{t-2} \odot (1 - m_{t-1}) \odot (1 - m_t)) + \cdots \\ &+ \gamma^n H_t \cdots H_{t-(n-1)}(x_{t-(n-1)} \odot (1 - m_{t-(n-2)}) \odot \cdots \odot (1 - m_t)) \\ &+ \gamma^n H_t \cdots H_{t-(n-1)}(\tilde{x}_{t-(n-1)} \odot (1 - m_{t-(n-1)}) \odot \cdots \odot (1 - m_t))\end{aligned} \tag{7.16}$$

The n steps of historical steps of states can be written in a summation form as

$$\begin{aligned}\tilde{x}_{t+1} &= \sum_{i=0}^{n-1} \gamma^{i+1} (\prod_{j=0}^{i} H_{t-j})(x_{t-i} \odot \bigodot_{j=0}^{i-1} (1 - m_{t-j})) \\ &+ \gamma^n H_t \cdots H_{t-(n-1)}(\tilde{x}_{t-(n-1)} \odot (1 - m_{t-(n-1)}) \odot \cdots \odot (1 - m_t))\end{aligned} \tag{7.17}$$

where $\sum$, $\prod$, and $\odot$ are the summation, matrix product, and Hadamard product operators, respectively. For simplicity, we denote the term with the $\tilde{x}_{t-n}$ in Equation 7.17 as $\mathcal{O}(\tilde{x}_{t-n})$, and the GMP of the complected states can be represented by

$$\tilde{x}_{t+1} = \sum_{i=0}^{n-1} \gamma^{i+1} (\prod_{j=0}^{i} H_{t-j})(x_{t-i} \odot \bigodot_{j=0}^{i-1} (1 - m_{t-j})) + \mathcal{O}(\tilde{x}_{t-(n-1)}) \tag{7.18}$$

In $\mathcal{O}(\tilde{x}_{t-(n-1)})$, when $n \to \infty$, since $\gamma \in (0, 1)$, $\gamma^{n+1} \to 0$. In addition, the product of masking vectors in $\mathcal{O}(\tilde{x}_{t-(n-1)})$ will also approach to zero, i.e. $\bigodot_{j=0}^{n-1}(1 - m_{t-j}) \to 0$. The probability of each element of $\bigodot_{j=0}^{n-1}(1 - m_{t-j})$ being zero is related to the value of n and the traffic state values' missing rate (mr), which can be calculated as $1 - (1 - mr)^n$. The larger the n and the mr are, the elements of $\bigodot_{j=0}^{n-1}(1 - m_{t-j})$ have the higher probability to be zeros. When $mr = 20\%$ and $n = 10$, the probability is $1 - 0.8^{10} = 89.26\%$. Besides, the last term in Equation 7.18 contains a γ^n, which will further degrade the contribution of the last term to the traffic forecasting task. Thus, when n is large enough, we consider the $\mathcal{O}(\tilde{x}_{t-(n-1)})$ is a negligibly term.

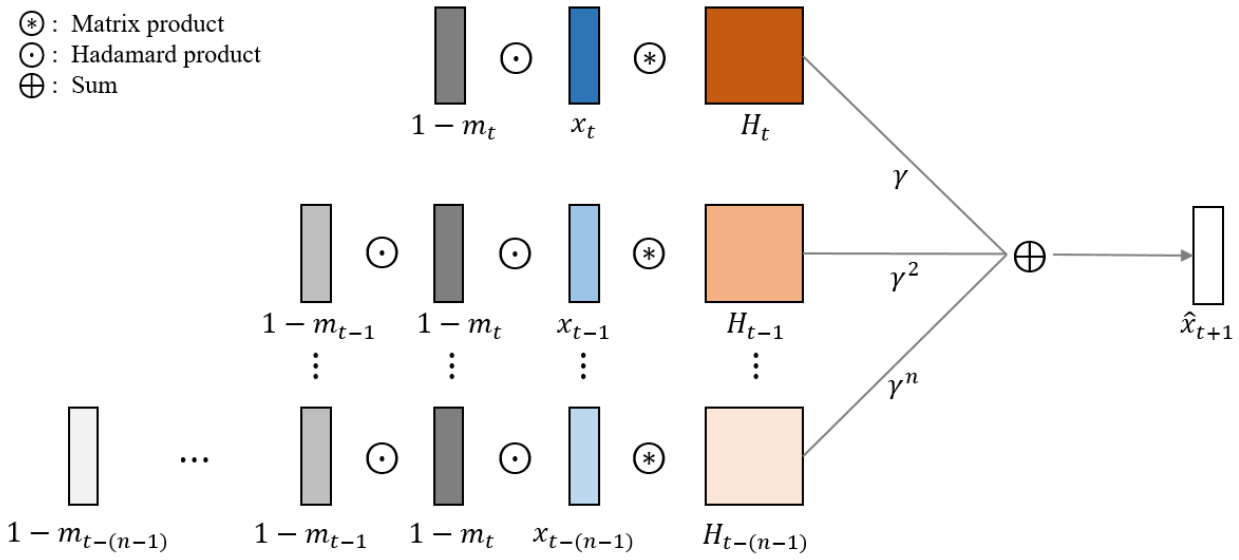


Figure 7.2: Structure of the proposed graph Markov network. Here, $H_{t-j} = \mathbf{A}^j \odot W_j$. As for the spectral version of GMN, $H_{t-j} = U\Lambda_j U^T$.

Figure 7.1 demonstrates the graph Markov process for inferring the future state. The traffic network graphs with attribute-missed nodes (in gray color) is convert into traffic state vectors. The inference of $\tilde{x}^d_{t+1}$ is based on historical traffic states by back-propagating to the $t-(n-1)$ step.

7.4.3 *Graph Markov Network*

In this section, we propose a **Graph Markov Network** (GMN) for traffic prediction with the capability of handling missing values in historical data. Suppose the historical traffic data consists of n steps of traffic states $\{x_{t-(n-1)}, ..., x_t\}$. Correspondingly, we can acquire n masking vectors $\{m_{t-(n-1)}, ..., m_t\}$. The traffic network's topological structure can be represented by the adjacency matrix.

The GMN is designed based on the proposed GMP described in the previous section. Since we consider the term $\mathcal{O}(\tilde{x}_{t-(n-1)})$ described in Equation 7.18 is small enough, the $\mathcal{O}(\tilde{x}_{t-(n-1)})$ is omitted in the proposed GMN for simplicity.

As described in Equation 7.18, the graph Markov process contains n transition weight matrices and the product of the these matrices $(\prod_{j=0}^{i} H_{t-j}) = (\prod_{j=0}^{i} \mathbf{A}^j \odot P_{t-j})$ measures the contribution of x_{t-i} for generating the $\tilde{x}_t$. To reduce matrix product operations and at the same time keep

the learning capability in the GMP, we simplify the $(\prod_{j=0}^{i} \mathbf{A}^j \odot P_{t-j})$ by $(\mathbf{A}^{i+1} \odot W_{i+1})$, where $W_{i+1} \in \mathbb{R}^{S \times S}$ is a weight matrix. In this way, $(\mathbf{A}^{i+1} \odot W_{i+1})$ can directly measure the contribution of x_{t-i} for generating the $\tilde{x}_t$ and skip the intermediate state transition processes. Further, the GMP still has n weight matrices $(\{\mathbf{A}^1 \odot W_1, ..., \mathbf{A}^n \odot W_n\})$, and thus, the learning capability in terms of the size of parameters does not change. The benefits of the simplification is that the GMP can reduce $\frac{n(n-1)}{2}$ times of multiplication between two $S \times S$ matrices in total.

Based on the GMP and the aforementioned simplification, we propose the ***graph Markov network*** for traffic forecasting with the capability of handling missing values as

$$\hat{x}_{t+1} = \sum_{i=0}^{n-1} \gamma^{i+1} (\mathbf{A}^{i+1} \odot W_{i+1})(x_{t-i} \odot \bigodot_{j=0}^{i-1} (1 - m_{t-j})) \tag{7.19}$$

where $\hat{x}_{t+1}$ is the predicted traffic state for the future time step $t+1$ and $\{W_1, ..., W_n\}$ are the model's weight matrices that can be learned and updated during the training process.

As for the spectral version of the graph Markov process, the product of the transition weight matrices $(\prod_{j=0}^{i} H_{t-j})$ can also be simplified. Because of the orthogonality of the eigenvectors of $\mathcal{L}$ (Wang and Mieghem, 2015), $U^T = U^{-1}$, and thus, $(\prod_{j=0}^{i} H_{t-j}) = (\prod_{j=0}^{i} U \Lambda_{\theta_{t-j}} U^T) = U(\prod_{j=0}^{i} \Lambda_{\theta_{t-j}}) U^T$. We further simplify the product of the transition weight matrices by replacing the $\prod_{j=0}^{i} \Lambda_{\theta_{t-j}}$ with a diagonal weight matrix Λ_{i+1}. Similar to the simplification of GMP, this simplification process will not reduce the learning capability of the SGMP because the amount of the weight parameters does not change. In this way, the SGMP can reduce at least $\frac{n(n-1)}{2}$ times of vector multiplication in total. Then, the spectral version of the graph Markov network (SGMN) can be defined as

$$\hat{x}_{t+1} = \sum_{i=0}^{n-1} \gamma^{i+1} (U \Lambda_{i+1} U^T)(x_{t-i} \odot \bigodot_{j=0}^{i-1} (1 - m_{t-j})) \tag{7.20}$$

where $\{\Lambda_1, ..., \Lambda_n\}$ are the diagonal weight matrices that can be learned and updated during the training process.

The structure of GMN for predicting traffic state x_{t+1} is demonstrated in Figure 7.2. The spectral version of GMN has the same model structure that it only need to replace the $\mathbf{A}^j \odot W_j$ with the $U \Lambda_j U^T$, as shown Figure 7.2. During the training process, the loss can be calculated by measuring the difference between the predicted value $\hat{y} = \hat{x}_{t+1}$ and the label $y = x_{t+1}$.

7.5 EXPERIMENTAL RESULTS

In this section, we compared the proposed approach with state-of-the-art traffic forecasting models with the capability of handling missing values. The graph Markov network predicts one time step ahead in this experiment, and if needed, two or more times ahead can also be predicted by using the last prediction as the input to the model. The time intervals depend on the datasets tested in the experiments. The datasets, hyper-parameters, software, and hardware used in the experiments are introduced.

7.5.1 *Datasets*

In this study, we conduct experiments on three real-world network-wide traffic state datasets. The topological structures of the traffic networks are also used in the experiments.

7.5.1.1 *PEMS-BAY*

This dataset named as PEMS-BAY is collected by California Transportation Agencies (CalTrans) Performance Measurement System (PeMS). This dataset contains the speed information of 325 sensors in the Bay Area lasting for six months ranging from Jan 1st, 2017 to Jun 30th, 2017. The interval of time steps is 5-minutes. The total number of observed traffic data points is 16,941,600. The adjacency matrix of this dataset is defined according to (Li et al., 2018). The dataset is published by (Li et al., 2018) on the Github (https://github.com/liyaguang/DCRNN).

7.5.1.2 *METR-LA*

This dataset is collected from loop detectors on the freeway of Los Angeles County (Jagadish et al., 2014). This dataset contains the speed information of 207 sensors lasting for 4 months ranging from Mar 1st, 2012 to Jun 30th, 2012. The interval of time steps is 5-minutes. The total number of observed traffic data points is 6,519,002. Similar to the PEMS-BAY dataset, the adjacency matrix of this dataset is defined according to (Li et al., 2018), and the dataset is published on the Github (https://github.com/liyaguang/DCRNN).

7.5.1.3 *INRIX-SEA*

This dataset is collected by the INRIX company from multiple data sources, including GPS probes, road sensors, and cell phone data. This dataset contains the speed information of the traffic network in the Seattle downtown area consisting of 725 road segments. The traffic network covers both freeways and urban roadways. The dataset covers a one-year period from Jan 1st, 2012 to Dec 31st, 2012. The interval of time steps is 5-minutes. The total number of observed traffic data points is 76,212,000. This dataset is provided by Washington Department of Transportation and has been used in (Cui et al., 2019). Due to privacy policies, this dataset is not published.

7.5.2 *Missing Values and Data formatting*

The dataset forms as a spatial-temporal matrix, whose spatial dimension size is the number of sensors and temporal dimension size is the number of time steps. In the experiments, the dataset is split into a training set, a validation set, and a testing set, with a size ratio of 6:2:2. In the training and testing process, the speed values of the dataset are normalized within a range of [0,1].

The PEMS-BAY and METR-LA datasets originally have missing values and their percentages of missing values are 0.003% and 8.11%, respectively. It should be noted that the original PEMS-BAY dataset should have already imputed the missing values linearly (Chen et al., 2002). But the PEMS-BAY dataset is acquired from the online link mentioned in Section 7.5.1.1. There are no missing values in the INRIX-SEA dataset. To test the model's capability of handling missing values with different missing rates, we randomly set a portion of speed values in the input sequences as zeros according to a specific missing rate and generate the masking vectors accordingly. In this study, based on each of the three datasets, three sub-datasets with artificial missing rates of 10%, 20%, and 40%, respectively, are generated. In METR-LA datasets, the original missing values are integrated with the artificial missing values. It also should be noted that, besides random missing, other missing patterns, such as consecutive missing values, are also common in the traffic domain (Boquet et al., 2019; Gondara and Wang, 2017; Laña et al., 2018). Since the proposed models dealing with the traffic network as a graph, they are more suitable for processing data with randomly missing values. Other patterns will be further studied in the future work.

7.5.3 *Hardware and Software Environments*

The experiments are conducted on a computer with 128GB memory, a Intel Core i9-7900X CPU, and a NVIDIA GTX 1080 Ti GPU. The proposed approach and all neural network-based baseline models are implemented based on PyTorch 1.0.1 using the Python language 3.6.8.

7.5.4 *Baseline Models*

- GRU (Cho et al., 2014): GRU referring to gated recurrent units is a type of RNN. GRU can be considered as a simplified LSTM.
- GRU-I : GRU-I is designed based on GRU. Since GRU is a type of a RNN with the recurrent structure, the predicted values from a previous step $\hat{x}_t$ can be used to infer the missing values in the next step. The completed traffic states with all missing values filled can be represented by $\tilde{x}_{t+1} = x_t + \hat{x}_t \odot (1 - m_t)$.
- GRU-D (Che et al., 2018): GRU-D is a neural network structure that is designed based on GRU for multivariate time series prediction. It can capture long-term temporal dependencies in time series. GRU-D incorporates the masking information and missing values' time interval as input such that it can utilize the missing patterns.
- LSTM (Hochreiter and Schmidhuber, 1997): LSTM is a powerful variant of RNN, which can overcome the gradients exploding or vanishing problem. It is suitable for being a model's basic structure for traffic forecasting.
- LSTM-I : LSTM-I is designed based on LSTM. The missing value inferring process of LSTM-I is similar to that of GRU-I.
- LSTM-M (Tian et al., 2018): LSTM-M is a neural network structure designed based on LSTM for traffic prediction with missing data. It employs multi-scale temporal smoothing methods to infer lost data.

7.5.5 *Model Parameters*

The batch size of the training samples is set as 64. The number of steps of historical data incorporated in the GMN model will have an influence on the prediction performance. Hence, the GMNs with 6-steps, 8-steps, and 10-steps of historical data are tested in the experiments, i.e. the n in Equations 7.19 and 7.20 are set as 6, 8, and 10. It should be noted that the value of n is not fixed that the larger the n is, the better performance the GMN model can achieve. In the following sections, we denoted these GMN models as GMN-6, GMN-8, and GMN-10, respectively. The corresponding SGMN models with different steps are denoted as SGMN-6, SGMN-8,and SGMN-10, respectively. As analyzed in Section 7.5.8.3, a decay rate ranging from 0.6 to 0.9 helps the model generate better performance. Thus, the decay parameter γ is set as 0.9 in the experiments. For the RNN-based baseline models, including GRU, GRU-I, GRU-D, LSTM, LSTM-I, and LSTM-M, their input sequences all have 10 time steps. The numbers of parameters of GRU-based and LSTM-based models are about $3S^2$ and $4S^2$, respectively, where S is the spatial dimension of the input data. The number of parameters of GMN and SGMN are nS^2 and nS, respectively.

7.5.6 *Training and Hyper-Parameters*

In the training process, the mean square error (MSE) between the label y_t and the predicted value $\hat{y}_t$, i.e. $\frac{1}{n}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ is used as the loss function, where N is the sample size. The Adam (Kingma and Ba, 2014) optimization method is adopted for both GMN models and baseline models to update parameters, as Adam is also used in (Che et al., 2018; Tian et al., 2018). Early stopping mechanism is used to avoid over-fitting. If there is no improvement on the MSE of the validation set in 5 consecutive epochs, the training will be stopped. The minimum improvement in MSE is set as 0.00001. We also design a learning rate decay mechanism for the training process to speed up the models' convergence. The initial learning rate of all models is set as 10^{-3}, which is identical to the learning rate in (Tian et al., 2018). If there is no improvement in 4 consecutive epochs, the learning rate is reduced an order of magnitude until it reaches 10^{-5}.

7.5.7 *Evaluation Metric*

The prediction accuracy of all tested models are evaluated by three metrics, including mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE).

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{7.21}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}|\frac{y_i - \hat{y}_i}{y_i}| \tag{7.22}$$

$$RMSE = (\frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|^2)^{\frac{1}{2}} \tag{7.23}$$

7.5.8 *Experimental Results*

7.5.8.1 *Comparing with baseline models*

The prediction results tested on the PEMS-BAY, METR-LA, and INRIX-SEA datasets with respect to different missing rates are displayed in Table 7.1, Table 7.2, and Table 7.3, respectively. All results presented these three tables were averaged with different runs of the experiment. Overall, the SGMN models are superior to other baseline models. The GMN models also perform well, especially on the PEMS-BAY dataset. However, the prediction performance of GMN models decreases faster than that of SGMN models along with the increase of the missing rate. Among the baseline models, the GRU-I model performs well that it achieve smaller RMSEs on the PEMS-BAY and INRIX-SEA datasets.

As shown in Table 7.1, the SGMN-10 achieves the smallest MAEs and MAPEs for all the three missing rates on the PEMS-BAY dataset. However, the RMSEs of GRU-I are the smallest ones for all missing rates. The test results on the INRIX-SEA dataset, as shown in Table 7.3, have the similar situation that SGMN models perform better in terms of the MAE and MAPE metrics and GRU-I achieves better RMSE results. Since RMSE takes the square root of the average squared

Table 7.1: Prediction performance on PEMS-BAY dataset

PEMS-BAY									
Model	Missing Rate = 10%			Missing Rate = 20%			Missing Rate = 40%		
	MAE (mph)	MAPE(%)	RMSE (mph)	MAE (mph)	MAPE(%)	RMSE (mph)	MAE (mph)	MAPE(%)	RMSE (mph)
GRU	1.608	3.133	2.608	1.787	3.522	2.911	2.052	4.095	3.320
GRU-I	1.108	2.133	**1.831**	1.185	2.296	**1.968**	1.385	2.729	**2.327**
GRU-D	5.320	13.584	9.163	5.347	13.609	9.160	5.387	13.67	9.180
LSTM	2.368	4.809	3.952	2.457	5.098	4.258	2.428	5.117	4.181
LSTM-I	2.218	4.001	7.472	2.373	4.278	7.742	2.058	3.989	5.863
LSTM-M	1.198	2.351	1.968	1.236	2.437	2.055	1.472	2.904	3.111
GMN-6	1.084	2.077	2.565	1.101	2.145	2.029	1.819	3.634	3.543
GMN-8	1.089	2.086	2.611	1.196	2.297	2.827	1.376	2.730	2.678
GMN-10	1.089	2.086	2.614	1.202	2.308	2.864	1.327	2.615	2.470
SGMN-6	1.009	1.930	1.877	1.064	2.048	2.067	1.930	3.671	4.375
SGMN-8	1.008	1.929	1.875	1.062	2.043	2.024	1.291	2.517	2.867
SGMN-10	**1.007**	**1.927**	1.874	**1.058**	**2.037**	2.018	**1.207**	**2.362**	2.473

Table 7.2: Prediction performance on METR-LA dataset

METR-LA									
Model	Missing Rate = 10%			Missing Rate = 20%			Missing Rate = 40%		
	MAE (mph)	MAPE(%)	RMSE (mph)	MAE (mph)	MAPE(%)	RMSE (mph)	MAE (mph)	MAPE(%)	RMSE (mph)
GRU	3.427	7.971	5.923	3.667	8.611	6.249	4.037	9.622	6.744
GRU-I	3.322	7.625	5.543	3.402	7.846	5.642	**3.389**	7.917	5.903
GRU-D	9.912	25.28	12.195	9.904	25.302	12.193	10.022	25.444	12.269
LSTM	3.477	8.050	6.015	3.652	8.559	6.263	3.899	9.300	6.663
LSTM-I	3.180	7.228	5.363	**3.267**	7.417	5.653	3.393	**7.826**	5.879
LSTM-M	3.253	7.374	5.540	3.368	7.666	5.717	3.410	7.837	**5.812**
GMN-6	3.384	7.300	5.624	3.477	7.488	5.583	3.913	8.518	6.362
GMN-8	3.565	7.684	6.126	3.653	7.852	6.001	3.864	8.365	6.083
GMN-10	3.708	7.969	6.512	3.792	8.131	6.411	3.961	8.518	6.216
SGMN-6	**3.145**	**6.836**	5.331	3.333	7.232	5.578	3.952	8.593	6.894
SGMN-8	3.174	6.889	5.362	3.318	7.203	5.552	3.699	8.053	6.186
SGMN-10	3.152	6.852	**5.321**	3.310	**7.187**	**5.525**	3.680	8.005	6.079

errors, it gives a relatively high weight to large errors. The smaller RMSEs of GRU-I indicate that GRU-I's prediction results tend to have less large errors. It may seem contradictory that

Table 7.3: Prediction performance on INRIX-SEA dataset

INRIX-SEA									
Model	Missing Rate = 10%			Missing Rate = 20%			Missing Rate = 40%		
	MAE (mph)	MAPE(%)	RMSE (mph)	MAE (mph)	MAPE(%)	RMSE (mph)	MAE (mph)	MAPE(%)	RMSE (mph)
GRU	1.097	3.964	2.158	1.146	4.143	2.257	1.256	4.530	2.443
GRU-I	0.888	3.220	**1.850**	0.939	3.419	**1.920**	1.057	3.889	**2.086**
GRU-D	3.039	11.597	5.408	2.947	11.399	5.160	2.873	11.136	5.008
LSTM	1.256	4.451	2.446	1.450	5.364	2.956	1.433	5.260	2.902
LSTM-I	0.945	3.363	2.400	0.910	3.255	2.094	1.592	5.155	5.156
LSTM-M	1.096	4.357	2.633	1.001	3.787	2.264	0.986	3.584	2.098
GMN-6	2.354	8.541	4.832	2.704	9.588	5.545	3.063	10.700	5.960
GMN-8	2.356	8.547	4.835	2.712	9.613	5.559	2.938	10.277	5.803
GMN-10	2.355	8.545	4.835	2.713	9.618	5.561	2.923	10.224	5.778
SGMN-6	**0.768**	2.715	1.922	0.829	2.940	2.038	1.355	4.949	2.983
SGMN-8	**0.768**	**2.713**	1.921	**0.826**	**2.929**	2.026	1.024	3.679	2.399
SGMN-10	**0.768**	2.716	1.921	0.827	2.934	2.026	**0.973**	**3.485**	2.283

GRU-I outperforms other models on RMSE, and the proposed model achieves best prediction performance on MAE and MAPE. By checking the prediction results, we found this is resulted by the different residual distributions of GRU-I and SGMN. Thus, although GRU-I performs better on several cases, prediction performance of SGMN at least achieves the same level of that of stat-of-the-art models.

Since traffic states in INRIX-SEA are influenced by traffic lights, the INRIX-SEA dataset is more complicated than others. Thus, SGMN containing more nonlinear functions works better than GMN for capturing complicated patterns. As for the results tested on the METR-LA dataset, shown in Table 7.2, the SGMN models outperform other models when the missing rates are 10% and 20%. When the missing rate increases to 40%, the GRU-I, LSTM-I, and LSTM-M models achieve better prediction performance in terms of all the three metrics. This phenomenon shows the proposed model is good at processing data with small missing rates. From the three tables, it can be observed that the time horizon n, ranging from 6 to 10, of the proposed model influences the performance. The model performs better when n is larger. This is more obvious when the missing rate is greater.

7.5.8.2 *Analysis on Training Time*

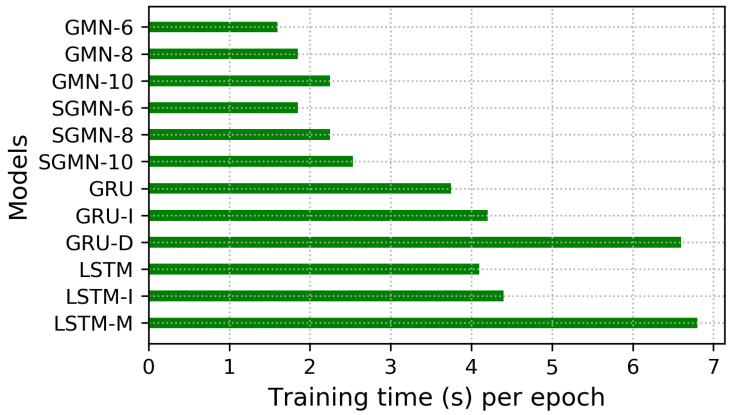
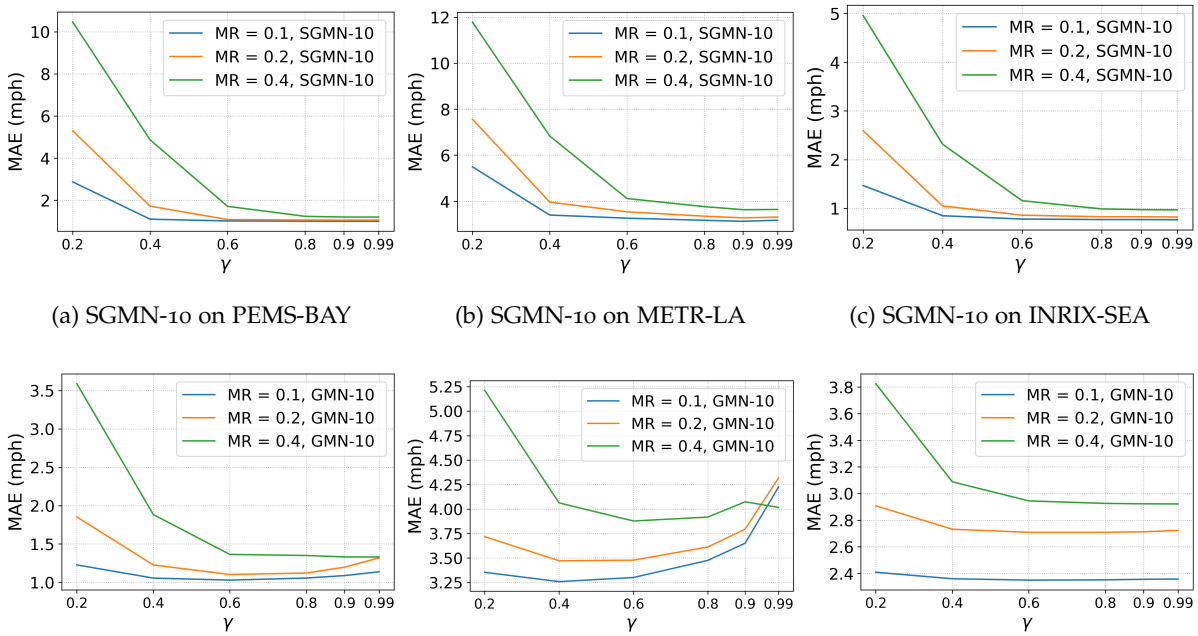


Figure 7.3: Training time of the compared models.

In this section, we analyze the training time of the proposed models and baseline models. Figure 7.3 shows the training time per epoch of the compared models tested on the PEMS-BAY datasets. The training times tested on different datasets have the same patterns. Since the GMN models have less matrix product operations than the SGMN models, GMN models take slightly less time per epoch than other models. The GMN and SGMN models apparently cost less running time than the baseline models because they get rid of the recurrent structure. The training times of GMN and SGMN increase when they incorporate more historical steps. The GRU and LSTM have similar training time per epoch. Since the GRU-I and LSTM-I have an imputation operation, their training times take a little bit more. In addition, since the GRU-D and LSTM-M both take more types of data as the input, their training time is much more than GRU and LSTM.

7.5.8.3 *Analysis on Decay Rates of GMN and SGMN*

The proposed graph Markov process adopts the decay rate $\gamma \in (0, 1)$ to represent the temporal impact of Δt on the traffic state transition process. In previous analysis sections, the Δt is 5-minutes, and the decay rates of GMN models are set as 0.9. In this section, we analyze the influence of the decay rate on the proposed models' prediction performance. The prediction performance of SGMN-10 and GMN-10 w.r.t. various decay rates are shown in Figure 7.4. The models are tested



(a) SGMN-10 on PEMS-BAY (b) SGMN-10 on METR-LA (c) SGMN-10 on INRIX-SEA

(d) GMN-10 on PEMS-BAY (e) GMN-10 on METR-LA (f) GMN-10 on INRIX-SEA

Figure 7.4: Prediction performance metric (MAE) w.r.t. the decay rate γ. The SGMN-10 and GMN-10 are tested on the PEMS-BAY, METR-LA, and INRIX-SEA datasets with different missing rates.

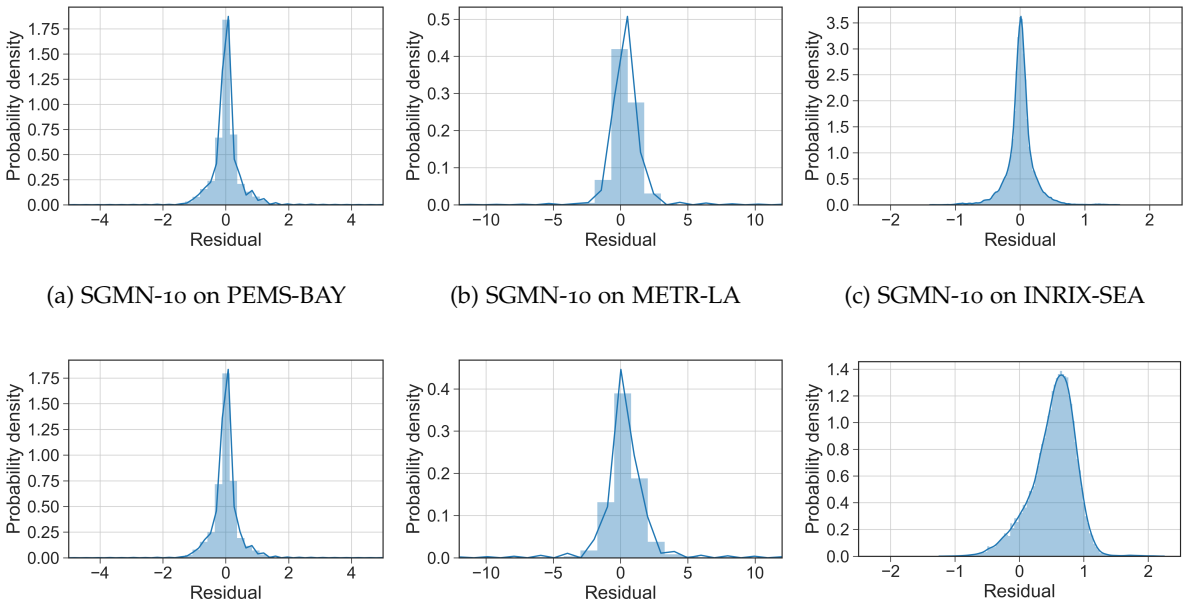
on the three datasets with different missing rates. Generally, the missing rate affects the prediction performance a lot that large missing rate results in large prediction errors.

The six sub-figures in Figure 7.4 all indicate the prediction errors (MAE) decrease along with the increase of γ. The prediction results of the SGMN-10 models tested on the three datasets have the similar curve patterns, as shown by the line-charts in Figure 7.4a, 7.4b, and 7.4c. When γ is close to zero, the prediction errors are relatively large. When γ is increasing, the prediction errors seem to be monotonically decreasing. When γ is close to one, the curves are almost flat and prediction errors nearly keep the same. However, as shown in Figure 7.4b, the MAE tested on the METR-LA dataset increases a little bit when γ increases from 0.9 to 0.99.

The MAEs of the GMN-10 models shown in Figure 7.4d, 7.4e, and 7.4f have slightly different patterns than those of SGMN-10 models. The MAE curves are not monotonically decreasing. When γ is close to one, the prediction errors start to increase. This phenomenon is particularly obvious in the results tested on the METR-LA dataset.

In addition, it should be noted that the y-axes of those sub-figures have various ranges. When the decay rate is relatively small (close to one), the prediction capability of GMN models is better than that of SGMN models. One possible reason is that GMN models contains more weight parameters than SGMN models. In summary, the selection of decay rate, which can be considered as a hyper-parameter and need to be tuned, depends on the dataset, the model, and even other hyper-parameters.

7.5.8.4 *Analysis on Forecasting Residuals*



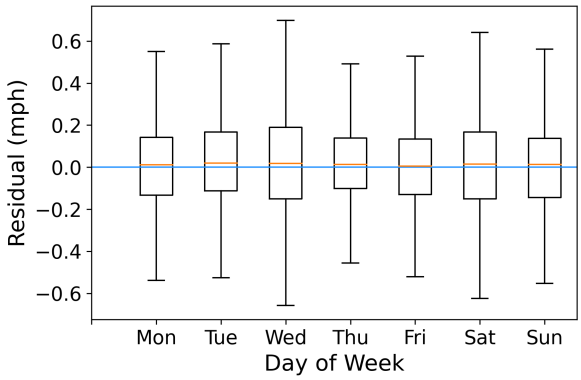
(a) SGMN-10 on PEMS-BAY (b) SGMN-10 on METR-LA (c) SGMN-10 on INRIX-SEA

(d) SGMN-10 on PEMS-BAY (e) SGMN-10 on METR-LA (f) SGMN-10 on INRIX-SEA

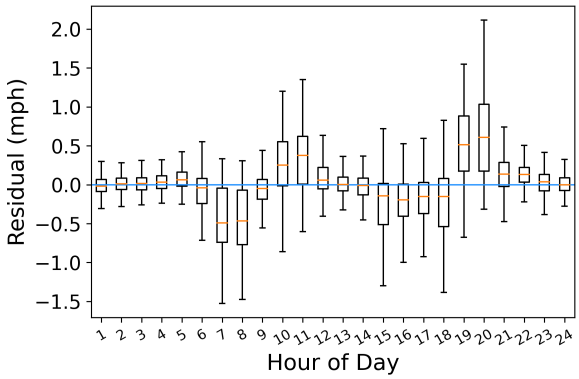
Figure 7.5: Prediction residuals of the proposed models tested on three datasets when the missing rate is 20%.

Since residual is an critical indicator for evaluating whether a model is systematically correct, the residuals of predictions are analyzed in this section. The residual equals the ground truth value subtracts the predicted value, i.e. $y - \hat{y}$. Figure 7.5 shows the residual distributions of SGMN-10 and GMN-10 tested on the three datasets. Most of the sub-figures display that the residual distributions follow normal distributions with zero means, except for the result of GMN-10 tested on the INRIX-SEA dataset. Although the proposed models are more complex than regression

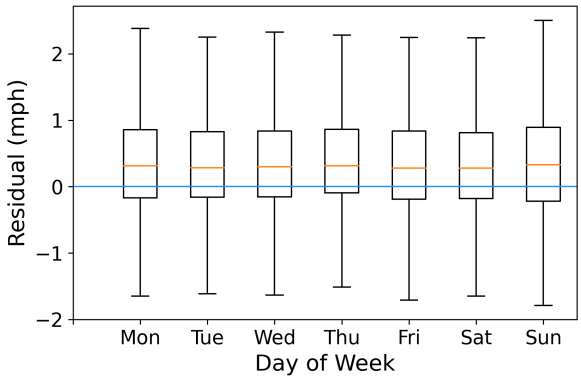
models, the residuals' normal distributions indicate that the proposed models have sufficient prediction capabilities and capture enough predictive information from the input data.



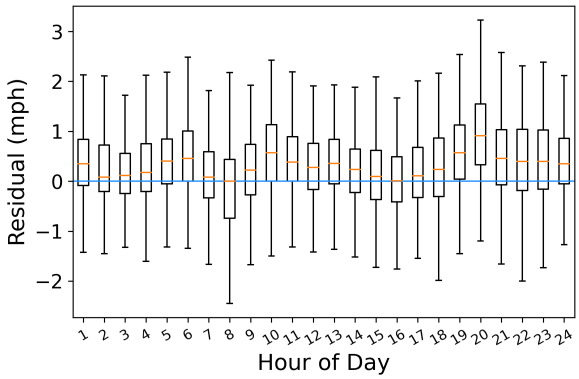
(a) PEMS-BAY



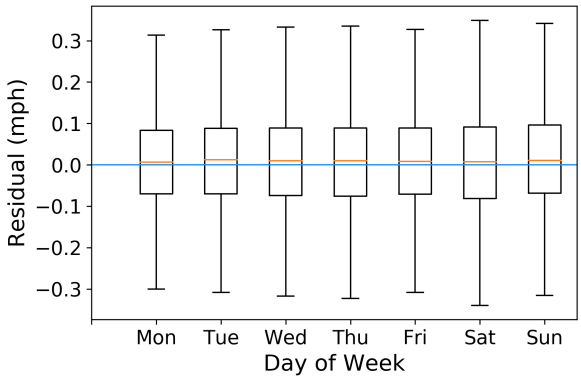
(b) PEMS-BAY



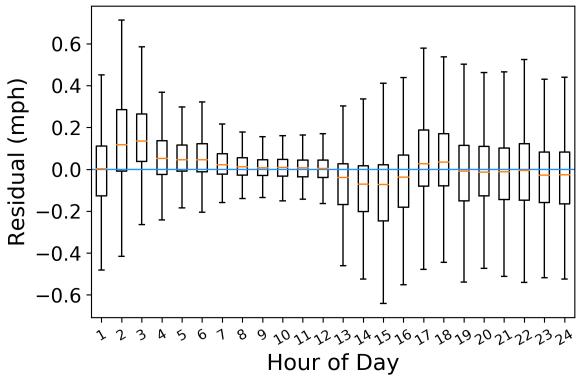
(c) METR-LA



(d) METR-LA



(e) INRIX-SEA



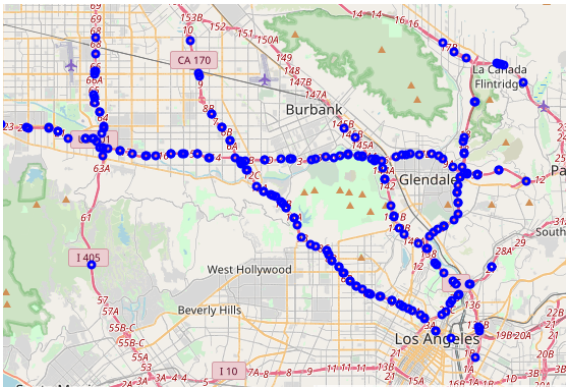
(f) INRIX-SEA

Figure 7.6: Prediction residuals of SGMN-10 with respect to day of the week and hour of day, tested on three datasets with the missing rate of 20%.

The prediction performance will also be influenced by the temporal information, such as hour of day and day of the week. Normally, during peak hours, traffic states with more variations are harder to be predicted. Thus, in this section, the influence of hour of day and day of the week is measured. The residuals of SGMN-10 tested on the three datasets with respect to day of the week and hour of day are shown in Figure 7.6. As displayed by the box-plots in Figures 7.6a, 7.6c, 7.6e, the prediction residuals on each day of the week are almost the same. That means the proposed models has the capability of forecasting traffic states on each day of the week. The residuals with respect to hour of day are displayed in Figures 7.6b, 7.6d, 7.6f. The influence of peak hours on traffic forecasting is particularly obvious on the PEMS-BAY dataset. However, the residual distributions in each hour of the day on the METR-LA dataset do not have much differences. The residual distributions on the INRIX-SEA dataset are abnormal to some extent that the the residuals are large during the afternoon and midnight. This phenomenon may be lead by the various traffic patterns of different types of roadways in different cities.

7.5.8.5 *Model Weight Analysis and Visualization*

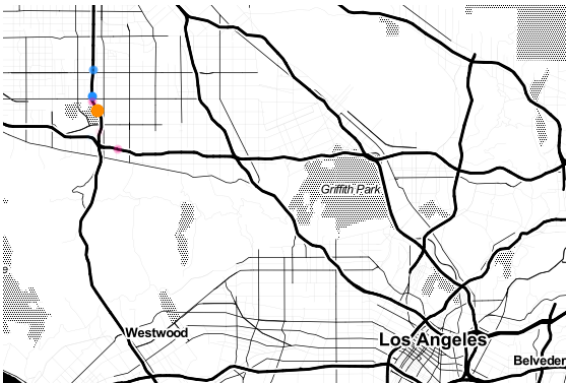
In this section, the proposed model's weights are analyzed and visualized. We take the SGMN-10 and GMN-10 trained on METR-LA dataset as an example. Figure 7.7a shows the 207 sensor locations in the METR-LA dataset denoted by blue dots, and Figure 7.7b shows the top 20 most influential sensor locations in terms of the influence on forecasting traffic states of the future $(t+1)$ step from the states of the current (t) step. The influence of a sensor of the k-th location is reflected by the sum/average of the squared element values in the k-th row/column of the model's weight matrix at the t step, i.e. the H_t described in Equation 7.18. For example, the averaged squared element values of the k-th row of H_t is calculated as $\frac{1}{n}\sum_{i=0}^{n-1}(H_t)_{k,i}^2$. Here, $H_t = \mathbf{A}^1 \odot W_1$ for the GMN case, and for the SGMN case, $H_t = U\Lambda_1 U^T$. As depicted in the map in Figure 7.7b, the selected top 20 influential sensor locations are mostly distributed near intersection areas, which has great potential to affect nearby traffic states. Figure 7.7c and Figure 7.7d display the influence of the 89-th sensor location on its neighboring sensor locations. This sensor location with the sensor ID 767351 is represented by an orange dot on the maps. The influence is reflected by the element values of the model's weight matrix at the 89-th row/column. The positive and negative weight values of other sensor locations are demonstrated by blue and pink colors, respectively.



(a) All sensor locations in the METR-LA dataset

(b) Top 20 influential sensor locations

(c) Weight ($U\Lambda_1 U^T$) visualization of SGMN-10 w.r.t. the 89^{th} sensor location, denoted by the orange dot

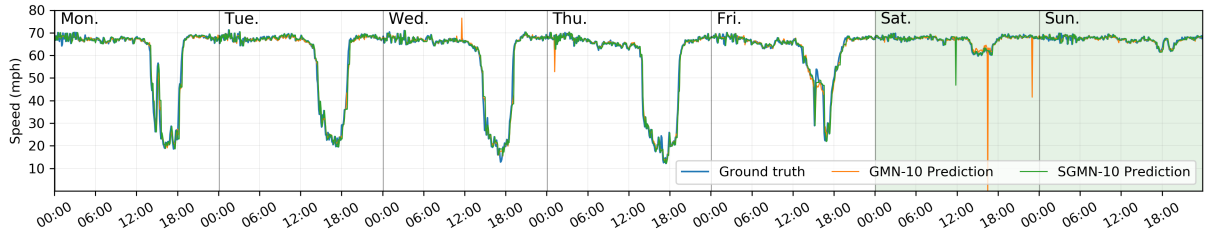


(d) Weight ($\mathbf{A}^1 \odot W_1$) visualization of GMN-10 w.r.t. the 89^{th} sensor location, denoted by the orange dot

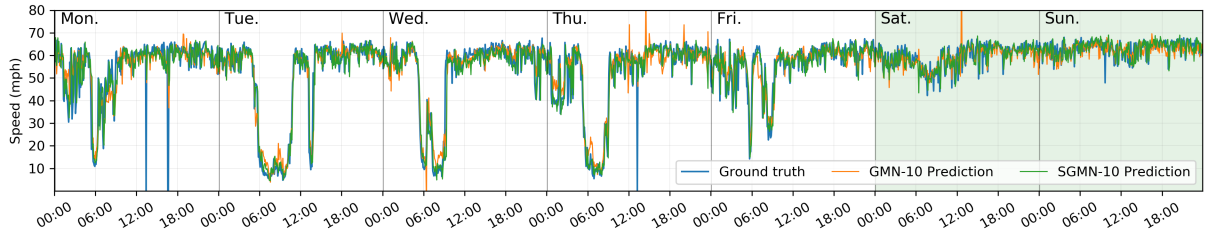
Figure 7.7: Visualization of sensor locations and models weights. The top 20 influential sensor locations in (b) are ones with the top 20 largest row-wise averaged squared element values of the weight matrix $H_1 = U\Lambda_1 U^T$ in SGMN-10. The blue and pink dots in (c) and (d) represent positive and negative weight values, respectively. The darker the color is, the larger the absolute value of the weight is.

The darker the color is, the larger the absolute value of the weight element is. The difference between these two figures is that the illustrated neighboring locations in Figure 7.7d are confined within a small one-hop neighboring area by the weight matrix of GMN-10, i.e. the $\mathbf{A}^1 \odot W_1$. However, as shown by the two figures, the surrounding sensor locations with respect to the 89-th sensor location (the orange dot) are obviously darker, which means the traffic state of a location is influenced more by the states of its neighbors. Thus, by quantitatively analyzing or visualizing the

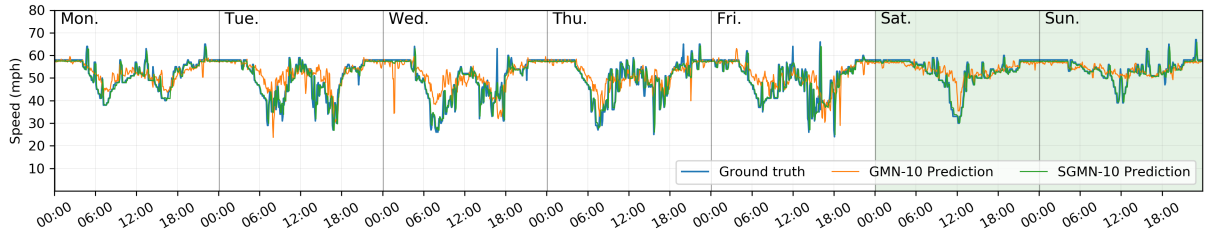
weight matrices of the proposed models, the influence of nodes/locations in a traffic network on their neighbors can be measured.



(a) Results tested on the PEMS-BAY dataset from Jan. 30, 2017 to Feb. 5, 2017. Sensor ID = 400097



(b) Results tested on the METR-LA dataset from Mar. 12, 2012 to Mar. 18, 2012. Sensor ID = 765273



(c) Results tested on the METR-LA dataset from Mar. 12, 2012 to Mar. 18, 2012. Sensor ID = 765273

Figure 7.8: Comparison of the ground truth and the speed predicted by GMN-10 and SGMN-10 tested on three datasets with the missing rate of 20% under the random missing scenario. The white and green regions in these figures demonstrate weekdays and weekends, respectively.

7.5.8.6 *Traffic Forecasting Result Visualization*

The locations covered by those datasets actually have various traffic patterns. In this section, to demonstrate the proposed model's prediction performance, we select several sensor locations/links from the three datasets and visualize the ground truth and predicted speed values. The three sub-figures in Figure 7.8 display the ground truth and speed values predicted by the GMN-10 and SGMN-10. The missing rates of the tested datasets are all set as 20%. Both GMN-10 and SGMN-10

work well on the PEMS-BAY dataset. Since the METR-LA dataset originally has missing values, there are some blue spikes reaching the bottom of Figure 7.8b demonstrating the original missing values. The prediction performance of SGMN-10 on the INRIX-SEA dataset is better than that of GMN-10. Overall, the proposed models have the capability of forecasting traffic states with missing values.

7.6 CHAPTER SUMMARY

In this chapter, we propose the GMN, which is a new neural network architecture for spatial-temporal data forecasting. We introduce two properties of the traffic state transition process and define a graph Markov process. Unlike other existing recurrent neural network (RNN)-based models dealing with traffic data as multivariate time series, the GMN handling the traffic state transition process as a graph Markov process. The proposed GMN can incorporate the spatial relationship between neighboring links and the links' temporal dependencies between different time steps. By incorporating the spectral graph convolution operation, we also propose a spectral graph Markov network (SGMN). The experimental results tested on a real-world dataset shows show that the proposed GMN and SGMN achieves superior prediction performance. Further, the proposed models' parameters, weights, and prediction residuals are discussed and visualized.

The future work will focus on enhancing the theoretical basis of the proposed graph Markov process. We will attempt to build a connection between the graph Markov process with the Markov random field to analyze the hidden factors influencing traffic states. In addition, we will conduct more experiments on multiple public accessible datasets. [1]

1 This chapter is a slightly modified version of "**Graph Markov Network for Traffic Forecasting with Missing Data**" published in *Transportation Research Part C: Emerging Technologies* and has been reproduced here with permission from the copyright holder. (doi: https://doi.org/10.1016/j.trc.2020.102671)

Part III

INTEGRATION AND PLATFORMS

8

MULTI-SOURCE TRANSPORTATION DATA INTEGRATION

8.1 OVERVIEW

In recent years, with the developments in traffic sensing, data storage, and communication technologies, the availability and diversity of traffic data have increased substantially. Transportation data is normally collected by traffic sensors, such as inductive loop detectors, monitoring cameras, and Wi-Fi/Bluetooth sensors. By means of more novel intelligent systems, transportation data can also be collected via personal mobile computing devices and probe vehicles. Moreover, transportation related data should include weather data, traffic incident data, emission data, etc. Therefore, transportation data are very beneficial to public agencies and researchers for managing transportation systems and conducting comprehensive traffic analysis.

In the context of modern traffic operations where highly accurate information is needed, single-source transportation data may not be sufficient enough. When the volume and variety of traffic data increase, fully making use of the existing multi-source data to provide accurate traffic information is becoming a challenging task (El Faouzi et al., 2011). The hurdles of integrating and utilizing multi-source data can be summarized in the following three aspects:

- The significant variability in the spatiotemporal resolution/granularity of multi-source data;
- The lack of standard geospatial representations of traffic roadways/network for multi-source data;
- There is no well-designed, widely-accepted, and off-the-shelf framework for traffic data integration.

Dealing with various spatiotemporal resolutions is one of the most challenging tasks for integrating multi-source transportation data, due to datasets might be collected and formatted by various transportation related practitioners for different purposes. The temporal resolution (the minimum time interval units) of traffic monitoring data may range from seconds to hours. The spatial granularity of traffic data also varies significantly. Some types of traffic data are collected to monitor arterials and urban corridors in downtown areas, and some others may be only applicable for freeways. Moreover, some datasets may overlap in terms of their spatial coverage, and some other datasets may have complementary monitoring areas. These differences in the spatiotemporal resolutions normally lead to big hurdles if multiple datasets have to be incorporated in real research and applications. Hence, to solve this problem and facilitate further analysis and applications, a high-resolution geospatial referencing representation of the traffic network for connecting different data sources is utilized in this study.

The second challenge is that the geospatial representations of roadways in different datasets usually differ from each other because there is no uniform geometric referencing layer for those datasets. The OpenStreetMap (OSM) (Haklay and Weber, 2008) provides a comprehensive worldwide map and the detailed information of roadway networks. Although OSM roadway layers are editable and with adequate roadway metadata, OSM roadway layers are still different from the roadway layers of the National Performance Management Research Data Set (NPMRDS) (Kaushik et al., 2015), which is used for supporting national highway system performance measurement and management activities. At the current stage, there is no standard geometric representation or GIS map layer for universal transportation analysis. If multiple datasets with different geospatial representations are used, a process of map conflation is normally needed to combine geographic information from overlapping sources so as to retain accurate data, minimize redundancy, and reconcile data conflicts (Longley et al., 2001). Therefore, in this study, a map conflation algorithm based on a uniform high-resolution map layer is proposed to integrate transportation data.

Further, modern transportation analysis and management are in great need of flexible and expandable transportation data integration frameworks. Data integration is the problem of combining data residing at different sources, and providing a unified view of these data to the users (Lenzerini, 2002). Designing data integration systems is an important procedure in a variety of real-world applications, especially in Intelligent Transportation Systems (ITS). Other conventional

problems in transportation modeling are also concerned with multi-source processing, like planning problems, demand estimation, and traffic estimation (El Faouzi et al., 2011). Transportation data integration frameworks and tools have been designed and implemented for multiple scenarios in both research and practice. However, these existing studies on traffic data integration are mainly oriented to specific traffic analysis problems and a few types of data sources are incorporated. Based on our review of the literature, there is no off-the-shelf framework designed for multi-category transportation data integration. Therefore, designing such a data integration framework is the main target of this study.

8.1.1 *Contribution and Organization of the Chapter*

In this chapter, a traffic data integration framework based on a uniform roadway referencing layer is proposed, which supports comprehensive traffic analysis based on multi-source data. According to data source characteristics, all transportation related datasets are categorized into four types in terms of the sensor's location and sensing area. To make the transportation data integration framework more efficient and extensible, an iterative map conflation algorithm as a component of the framework is proposed. In addition, the proposed traffic data integration framework is implemented on an interactive online transportation analytical platform, Digital Roadway Interactive Visualization and Evaluation Network (DRIVE Net) system (Cui et al., 2016; Ma et al., 2011). Several real-world transportation data analytics functions, including travel time analysis, freeway performance measurement, and traffic safety analysis, are implemented based on the proposed framework and discussed.

The originality and contribution of this work can be summarized as follows:

1. A new traffic data integration framework based on a uniform roadway referencing layer for multi-source traffic data is proposed.

2. An iterative map conflation (IMC) algorithm for the on-road segment-based traffic data is proposed. The experimental results show that this proposed algorithm is efficient and effective enough to be taken into practice.

3. To make the framework more extensible and flexible, a transportation data categorization strategy from the perspective of the sensor's location and sensing area is proposed.

4. The proposed framework is implemented on a data-driven transportation analytics platform. Several interactive analysis tools and case studies based on this framework are implemented and introduced.

The rest of the paper is organized as follows. Section 8.2 relevant previous data integration studies and applications. Then, the category of data and the proposed data integration framework are introduced in detail in Section 8.3 and Section 8.4, respectively. Next, in Section 8.5 the experimental results are presented, followed by introducing several real applications and case studies based on the framework. Finally, Section 8.6 provides the discussion and concludes this chapter.

8.2 RELATED WORK

Technological advances in traffic sensing, telecommunications, data storage, and data processing have facilitated the improvement in existing means of traffic data collection. Due to different types of sensors are utilized and the heterogeneous data collected by these sensors need to be combined, different traffic data integration techniques have been developed to suit specific applications. To visually manipulate and analyze urban traffic data, a framework for integrating different spatial and temporal levels of granularity was proposed (Claramunt et al., 2000). To estimate the traffic state on freeways and signalized arterial links, loop detector data and probe vehicle data were integrated (Valadkhani et al., 2017). According to a survey about data fusion in ITS, the data fusion approaches can be split into three ways: statistical, probabilistic, and artificial intelligence approaches. Multiple algorithms and architecture design of multisensory data fusion are also introduced and summarized (Fourati, 2015). Data integration methods are also proposed for some special data source. The integration problem of in-vehicle information and loop detector data are studied (Cremer and Schrieber, 1996). To transfer data between two separate geographic information database, Graettinger et al. (2009) proposed a linear referencing approach to combine information from the state route linear referencing system (LRS) with information from the local

road LRS. The U.S. DOT has also provided potential use cases for integrating emerging data sources into operational practice (Gettman et al., 2017).

Due to transportation data normally describes states of roadway segments or traffic networks, the characteristics of the roadways, like geospatial information, play an important role in the traffic data integration process. Thus, multiple data integration methods are designed based on roadway map conflation or GIS data combination. To integrate sidewalk data with transportation network data in GIS, a Split-Match-Aggregate algorithm is proposed (Kang et al., 2015). Another study integrated global positions system and geographical information systems for traffic congestions analysis (Taylor et al., 2000). Further, a conflation methodology for two roadway networks is proposed by taking advantage of GIS capabilities in the projection of transportation networks (Daneshgar et al., 2018). To create an integrated bike map, an optimized four-step geographic data conflation system is proposed to conflate data from an authoritative source (Los Angeles County Metropolitan Transportation Authority) and an open source (OpenStreetMap) (Li and Valdovinos, 2018). However, most of these existing studies integrated a few types of transportation data and they did not propose an extensible transportation data framework. The framework proposed in this study has high efficiency and adaptability that it can fulfill the data integration for various new data sources.

8.3 DATA INTEGRATION

8.3.1 *Preliminaries*

In this study, data integration refers to the integration of multiple spatiotemporal transportation datasets. The data integration process contains two aspects: the spatial integration and temporal integration. A set of n spatiotemporal transportation datasets that need to be integrated can be denoted as $X = X_1, \ldots, X_n$. The spatiotemporal values in each dataset is represented by $X_i \in \mathbb{R}^(T_i \times D_i)$, in which T_i and D_i are the temporal and spatial dimensions, respectively, of the i-th dataset. In addition, each dataset X_i is associated with a geospatial/geometric representation of its corresponding roadway network, denoted as G_i. A uniform referencing layer covering the

traffic networks of all the datasets is denoted as G_0. Then, given X, $G = G_1, \ldots, G_n$, and G_0, the geospatial integration process can be defined as a function $\mathcal{F}$ such that

$$\mathcal{F}((X_1, G_1), \ldots, (X_n, G_n); G_0) = ((X_1, G_1, L_1), \ldots, (X_n, G_n, L_n); G_0) \tag{8.1}$$

where L_i is generated linkage information that builds a link between G_i and G_0. The core of the spatial integration is to find the best linkage data set $L = L_1, \ldots, L_n$. As for the temporal integration process, since the temporal information in transportation data are measured in the same unit, the integration can be fulfilled easily by aggregating data based on specific time units.

8.3.2 *Transportation Data Introduction and Categorization*

To build a general traffic data integration framework, it is efficient to classify all the datasets and design the integration framework for each category, in which multiple datasets can be processed in a similar way. Transportation related data are normally collected by location-fixed sensors, mobile devices, probe vehicles, or historical records reported by roadway management agencies. In order to describe the proposed data integration framework, several representative types of data used in this study are introduced in detail in the experimental section and their brief introductions are described as following:

- Location-fixed sensor data: loop detector based traffic speed/volume data, Wi-Fi/Bluetooth based travel time data, cellular mobile phone station based traffic speed data, etc.
- Probe vehicle based data: Google real-time traffic data, HERE segment based travel time data, INRIX segment based traffic speed data, etc.
- Incident data: traffic accident data, construction or alert data, etc.
- Weather information data.

In this study, to describe the proposed data integration framework in a concise and efficient way, several representative datasets are adopted. The main properties of these representative datasets, including covered area, sensing area, sensor location, estimated information, and time interval, are summarized and presented in Table 8.1. The sensing area describes whether a sensor monitors

Table 8.1: Data Summary and Comparison

Data	**Data Source**	**Covered Area**	**Sensing Area**	**Sensor location**	**Information**	**Time Interval**
Loop Data	Inductive loop detectors	Freeway main lanes and ramps	Point/milepost on roadway	On-road	Speed, volume, and occupancy	5-min
NPMRDS Data	Probe vehicles	Freeway and urban corridors	Segment on roadway	On-road	Speed and travel time	5-min
Incident Data	Reported by police and agencies	Freeways	Point/milepost	On-road	Incident information	flexible
Weather Data	Weather stations	All roadways	Point/milepost	Off-road	Weather information	5-min

traffic for a point or a segment of a roadway. The sensor location denotes whether a sensor is deployed on the roadway (on-road) or not (off-road). The time interval indicates the minimum or suitable time interval of a data source for analysis. Further, to demonstrate the various sensor coverage areas of these data sources, sample datasets in the Seattle area is illustrated on the real map as shown in Figure 8.1. The NPMRDS data, including the HERE and INRIX data, and Wi-Fi/Bluetooth based data can cover urban corridors, which can hardly be fulfilled by other types of data.

According to the SHRP2 L02 product report (List et al., 2014), transportation data can be classified based on multiple principles, like data collection methods, types of information disseminates, and methods of information dissemination. However, according to the comparison, as shown in Figure 8.2, the geospatial attributes, i.e. the sensor location and the sensing area, are the most significant ones that distinguish the studied data sources from each other. Thus, the sensor location and the sensing area are chosen as the two main metrics to group data sources into four categories, as shown in Table 8.1. Given these two metrics, not only the data sources used in this study can be well-classified, but other types of transportation data can also be categorized for data integration.

It should be noted that the GPS-based vehicle trajectory data can also be classified into the on-road point-based data and integrated via similar methods. However, as a type of raw data for

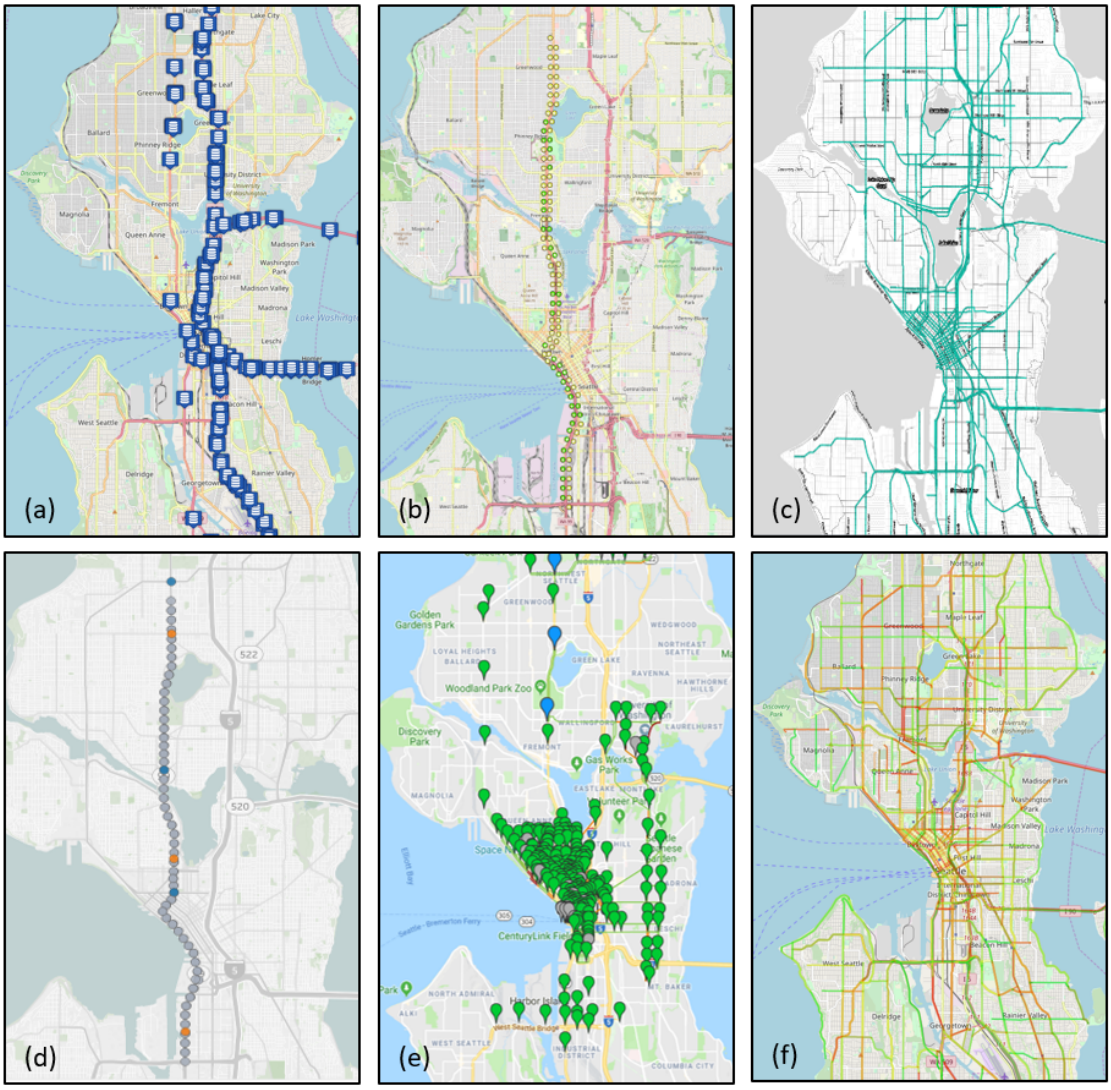


Figure 8.1: Representative data sources. (a) Loop detector sensors. (b) Verizon virtual sensors. (c) HERE traffic state map. (d) Surveillance cameras. (e) Bluetooth/Wi-Fi sensors. (f) INRIX traffic state map.

traffic performance measurement, trajectory data has different storage structures and analysis methodologies comparing to other on-road point-based data. Thus, the vehicle trajectory data is not included and discussed in this paper. In addition, some data sources may not be perfectly classified by these two metrics, considering some sensors can be deployed on road or off road, like Bluetooth and Wi-Fi sensors.

Sensing Area / Sensor Location	Segment-based	Point-based
On-Road	• NPMRDS Data (HERE and INRIX) • Real-time Traffic Data, like Google Traffic	• Loop Detector Data • Incident Data • Construction/Event Data • Surveillance Camera Data
Off-Road	• Verizon Speed Data • Pedestrian/Cycle Path Data	• Weather Station Data • Bluetooth/Wi-Fi Data

Figure 8.2: Data Categorization based on Sensor Location and Sensing Area

8.4 DATA INTEGRATION FRAMEWORK

A well-designed traffic data integration framework should work effectively and efficiently for all types of data. In order to fulfill this requirement, a traffic data integration framework based on a uniform roadway referencing layer is proposed. By using this framework, only a roadway referencing layer and some linking layers/tables, which link data sources to the referencing layer, are needed for further analysis. In this way, all the data sources can retain their original data formats. Figure 8.3 shows the traffic integration framework. Each data source contains its own roadway layer or geospatial information, and each dataset connects to the roadway referencing layer via a linking table, acting as the linkage data L_i as shown in Equation 8.1. To make terminologies clear in this study, the geometric layer of a data source that needs to be conflated is called the conflated layer, according to Chen et al. (2006). The referencing layer acts as an interface, by which the analysis tool does not need to care about the geospatial information of each dataset. When analyzing roadway performance, the analysis tool only needs to specify the analysis zone on the roadway referencing layer.

Due to most of the state agencies focus on freeway performance measurement and the data acquired for this study mostly measures freeway traffic states, the authors select the freeway system in Washington State as the study area. In this study, the WSDOT 24k map layer [1] is selected as the referencing layer. The 24k map contains a routable freeway network and each roadway segment contains multiple critical information for data integration, including segment

1 https://www.wsdot.wa.gov/mapsdata/geodatacatalog/

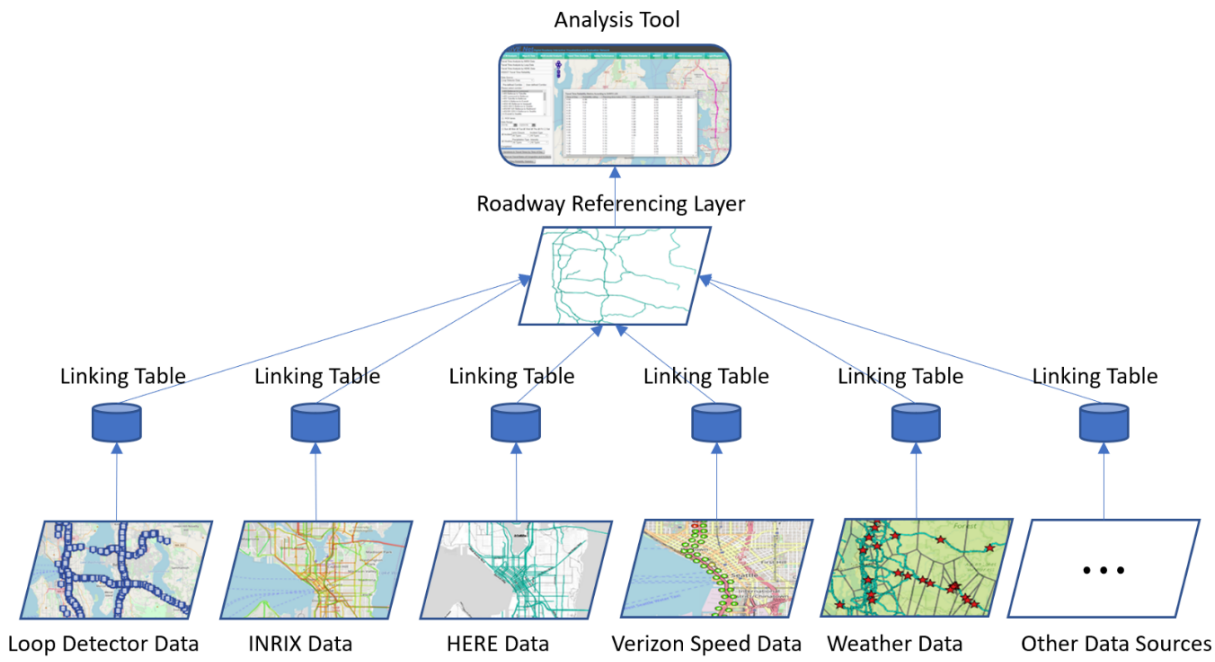


Figure 8.3: Proposed traffic data integration framework based on a uniform roadway referencing layer.

ID, direction, route name, route ID, segment length, starting milepost, ending milepost, starting point ID, ending point ID, segment geometric information, etc.

The data integration framework has the capability of integrating the four types of data, i.e. on-road point-based, on-road segment-based, off-road point-based, and off-road segment-based data. The detailed integration algorithms are introduced as follows.

8.4.1 *On-road Segment-based Data*

The on-road segment-based data normally has its own roadway map layer, like INRIX and HERE data have their own TMC-based roadway layers. Due to the definitions of segments in the referencing layer and the roadway layer of on-road segment-based data are totally different, these layers should be conflated for further data integration and analysis. To this end, an iterative map conflation algorithm for on-road segment-based roadway layers is proposed. In this section, the HERE data is considered as the representative on-road segment-based data and the HERE map layer is considered as the conflated layer, for an example, to demonstrate the details of the map conflation algorithm.

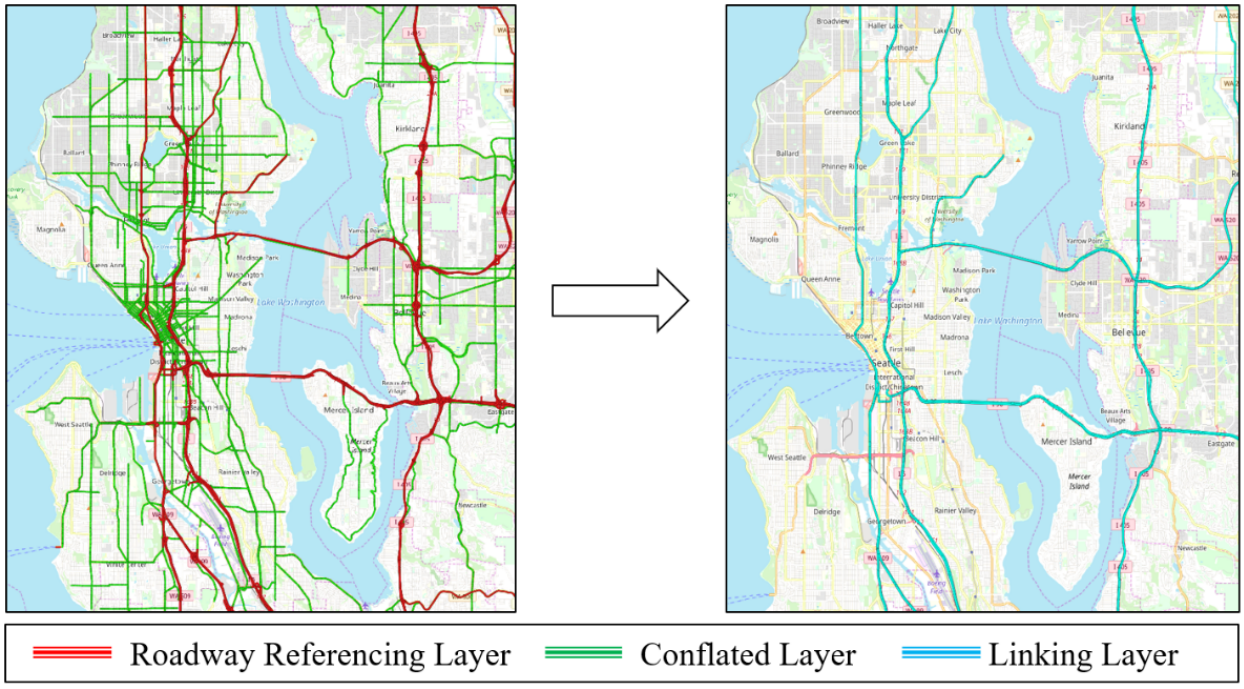


Figure 8.4: The roadway referencing layer and the conflated layer on the real map. The goal of the map conflation algorithm is to build a linking layer.

General Map Conflation Process: Figure 8.4 shows the spatial structure of the referencing layer and the conflated (HERE) layer on the real map. It is obvious that the referencing layer mainly contains freeways and the conflated layer covers more corridors, especially arterials and urban streets. The goal of the map conflation algorithm is to build a linking layer or generate a table of linkage data to connect the referencing and conflated layer.

The basic map conflation process is demonstrated in Figure 8.5 and briefly summarized as following steps:

1. For each roadway segment in the referencing layer with a Referencing segment ID (RID), break it into small segments (around 30 meters). Each newly generated small segment shares the same RID.

2. For each newly generated small segment, search and find the nearest segment with a conflation ID (CID) in the conflated layer to match based on several geometrical parameters. In this case, each newly generated small segment has the same CID.

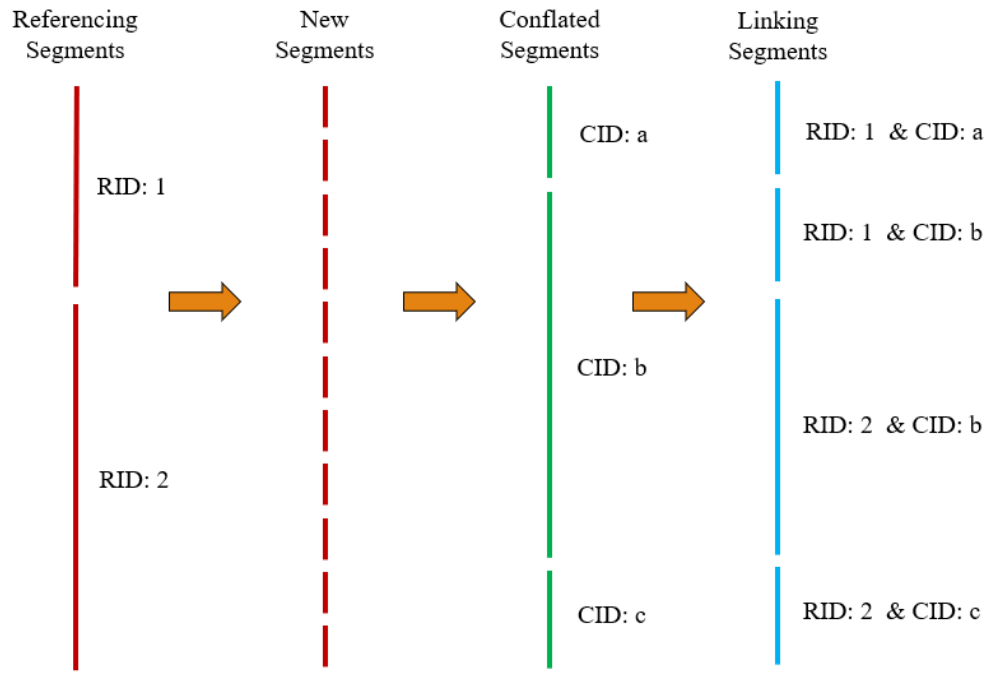


Figure 8.5: A demo of the map conflation process. RID denotes the Referencing segment ID and CID denotes the Conflated segment ID.

3. Aggregate a group of newly generated small segments with the same CID and HID into a new link.

4. Create an entry for each new links in the linkage table, which contains the RID and HID information.

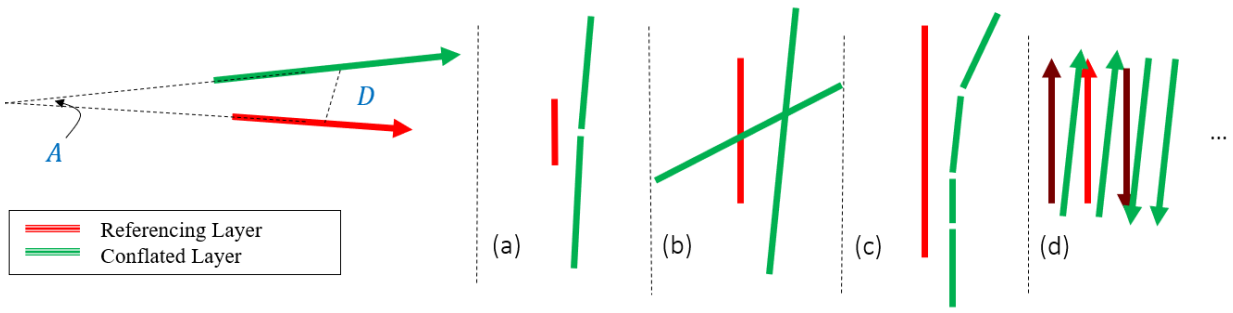


Figure 8.6: A demo of the map conflation process. RID denotes the Referencing segment ID and CID denotes the Conflated segment ID.

Iterative Map Conflation (IMC) Algorithm: The searching-and-matching step in the map conflation process is one of the core steps. Several important thresholds are utilized to find the nearest roadway segment pairs between the referencing and conflated layers. These thresholds include

- The distance between the centroids of two segments in the referencing and conflated layers, D.
- The angle between two segments in the referencing and conflated layers, A.

Besides, some other factors also need to be considered, like lane types and the definitions of the of roadway segment direction in different datasets. Figure 8.6 demonstrates the thresholds and several searching-and-matching scenarios. During the matching process, a pair of nearest segments are selected from the referencing layer and the conflated layer, given the two thresholds, D and A. According to the matching scenarios, as shown in Figure 8.6 (a-d), the thresholds, D and A, need to vary according to scenarios if an efficient searching-and-matching process is required.

Thus, an iterative map conflation algorithm is proposed, which can iteratively adjust the thresholds, D and A, to conduct the map conflation until all segments are conflated. The detailed algorithm is shown in Algorithm 8.1. The thresholds, D and A, are initialized as the same for each segment in the referencing layer. If a newly generated segment cannot find the nearest segment in the conflated layer under this threshold pair (D,A), the values of D and A will be increased until all the segments are paired. The increasing mechanism can follow a linear, exponential, or other

patterns. In addition, the increase function can let the values of D and A increased simultaneously or separately.

Algorithm 8.1: Iterative Map Conflation Algorithm

Result: Set of conflated pairs $P = \{(R_i^k, C_p)\}$

Initialization: $\mathcal{R} = \{R_0, \ldots, R_i, \ldots, R_M\}$, a set of roadway segments in the referencing layer, where R_i is the RID of segment i; $\mathcal{C} = \{C_0, \ldots, C_j, \ldots, C_M\}$, a set of segments in the conflation layer, where C_j is the CID of segment j ;

while $\mathcal{R}$ *is not null* **do**

  Split R_i into a set, $R_i = \{R_i^0, \ldots, R_i^k, \ldots, R_i^K\}$ containing all newly generated segments;

  Initialize thresholds, distance D and angle A;

  for $i \leftarrow 0$ to M do ;

  if R_i *is not null* **then**

    Search and Match R_i^k from C based on thresholds (D, A);

    Find the nearest pair (R_i^k, C_p) such that $\text{Dist}(R_i^k, C_p) \geq D$ & $\text{Angle}(R_i^k, C_p) \leq A$;

    $R_i.\text{pop}(R_i^k)$ & $P.\text{add}((R_i^k, C_p))$;

  end

  $(D, A) \leftarrow \text{Increase}(D, A)$

end

After running the map conflation algorithm, the linking layer, which connected the referencing layer and the conflated layer, will be automatically generated.

8.4.2 *Off-road Segment-based Data*

The off-road segment-based data is similar to the on-road segment-based data because they all measure traffic states of roadway segments. However, due to the sensors are deployed off-road, integrating off-road segment-based data needs to firstly map sensors to roadways, and then, utilize the introduced iterative map conflation algorithm. In this section, we take the Verizon speed data as an example to show the off-road sensor mapping process.

Although the Verizon speed data are collected via cellular towers, the Verizon data provider defines multiple virtual traffic measuring segments on roadways between towers to acquire speed

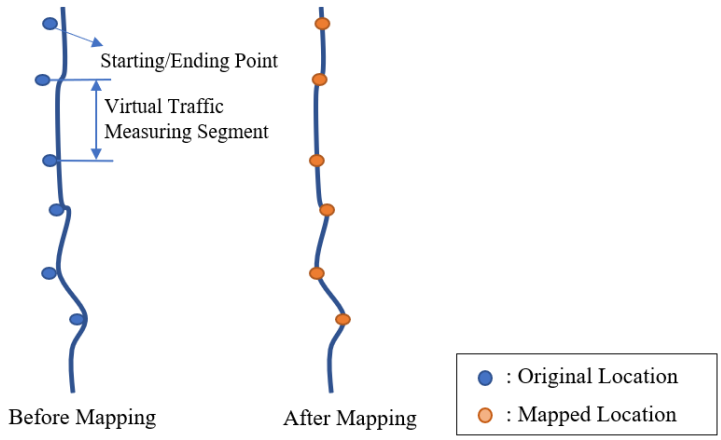


Figure 8.7: A demo of off-road segment-based traffic data integration, taking the Verizon speed data on SR-99 in Seattle as an example.

data with higher precision and spatial resolution. However, the starting and ending points of these virtual segments are not perfectly located on the roadway segments of the referencing layer. Thus, a mapping method is proposed and demonstrated in Figure 8.7. The nearest point on the referencing layer is found for each of the virtual segment starting/ending points. These nearest points on the referencing layer are considered as new starting/ending points, and the new segments between these points are considered as new traffic monitoring segments. In the next step, these new segments can be integrated into the referencing layer by using the proposed iterative map conflation algorithm.

8.4.3 *On-road Point-based Data*

The on-road point-based traffic data is the easiest type of data for data integration, due to this type of data contains the linear referencing information, i.e. the route number and the mileposts of sensors. This type of data, such as the loop detector data and incident data, does not need their own geometric roadway layers to be conflated. Since the referencing layer contains milepost information of each roadway segment, by matching the milepost information, each on-road point-based sensor can be accurately located on the referencing layer. Hence, the integration of the on-road point-based traffic data is inherently completed with the help of the route number and milepost information.

8.4.4 *Off-road Point-based Data*

As a representative off-road point-based data, weather data is critical for roadway performance measurement, and thus, it needs to be integrated to the referencing layer. In this section, we take the weather data as an example to show the integration method for off-road point-based traffic data.

An inherent property of the weather data is the coordinates of weather stations. However, weather stations are not perfectly located on or near freeways. What's more, the sizes of areas covered by weather stations are different. Due to the weather data cannot provide detailed coverage information, a mapping mechanism for data integration is proposed, which maps each roadway segment to its nearest weather station.

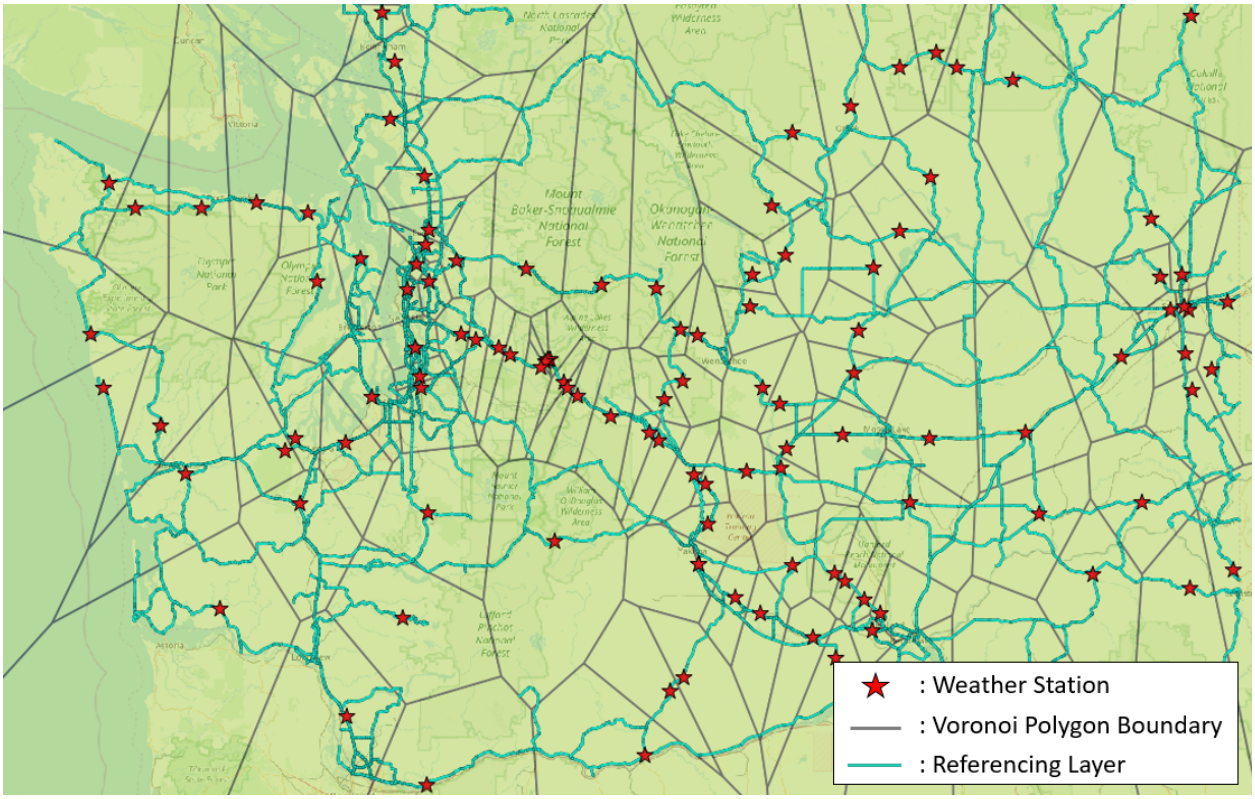


Figure 8.8: A demo of weather station integration based Voronoi map

To find the nearest weather station of each roadway segment, a spatial partitioning method based on the Voronoi diagram (Okabe et al., 2009) is adopted in this framework. The whole Washington State is separated by Voronoi polygons, which are centered at the weather stations.

Figure 8.8 shows a map with all the weather station centered polygons and the WSDOT 24k map layer. The edges between polygons are the intersected midlines between weather station pairs. In this case, the polygons perfectly partition the whole map and each roadway segment on referencing layer is covered by a polygon centered at the weather station. The red stars demonstrate the locations of the weather stations. In this way, each roadway segment in the referencing layer is covered by at least one polygon. If a roadway segment is covered solely be one polygon, then the segment is mapped to the weather station at the center of this polygon. Otherwise, the roadway segment will be mapped to a weather station whose corresponding polygon covers most of the roadway segment. The spatial partitioning and mapping process is fulfilled by using PostGIS.

8.4.5 *Framework Summary*

Due to the four types of data have different characteristics, the proposed traffic data integration framework provides different solutions for them. Figure 8.9 illustrates the proposed traffic data framework based on a uniform roadway referencing layer and the solutions for the four types of data. Comparing to existing data integration framework (Chen et al., 2006; Graettinger et al., 2009; Green et al., 2013; Kang et al., 2015; Ma et al., 2011; Valadkhani et al., 2017), the proposed traffic data integration framework has several advantages as follows:

1. This framework is flexible, due to it provides data integration solutions to four types of traffic data and most of the transportation data can be categorized into those four types.

2. This framework is efficient, owing to it only needs a uniform roadway referencing layer to manipulate all types of transportation data. The integration process is also very efficient in terms of the percentage of the automatically integrated road segments and the computation time.

3. This framework is extensible that it can be applied for any other transportation data integration tasks.

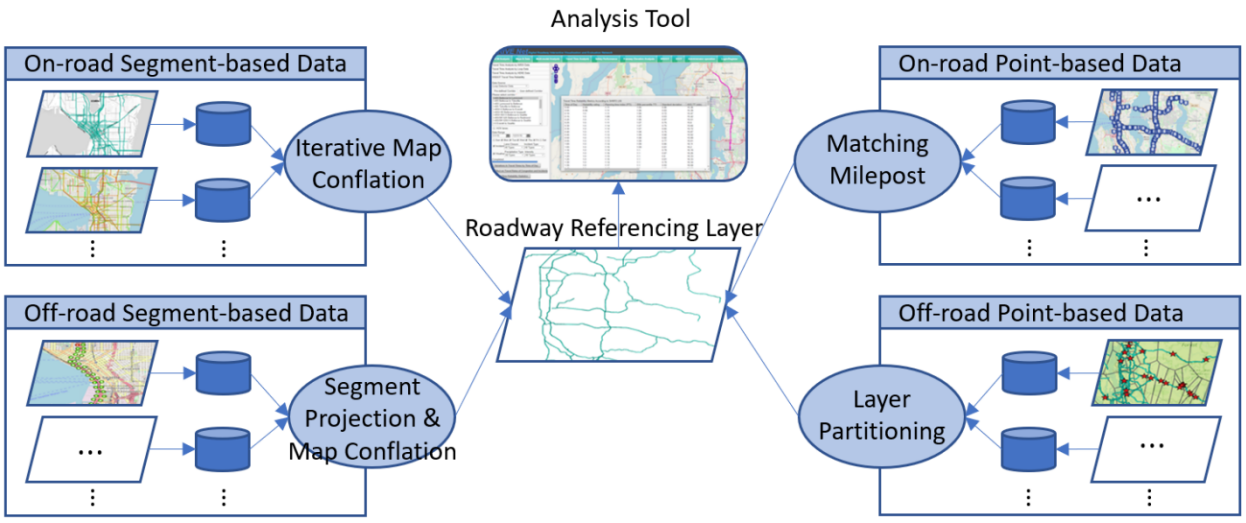


Figure 8.9: The proposed traffic data integration framework based on a uniform roadway referencing layer for four types of traffic data.

8.5 EXPERIMENTAL RESULTS

8.5.1 *Data Description*

The datasets used in this study contains freeway loop data, NPMRDS data, Verizon speed data, incident data, and weather data. All the data are collected or provided by the Washington State Department of Transportation (WSDOT) and the Seattle Department of Transportation (SDOT). The selected study area mainly focuses on the freeway system in Washington State. The year 2015, which is covered by all datasets, is selected as the study period. The capability of integrating data sources of the proposed framework is not restricted by the aforementioned datasets. However, the sensing areas of some other data sources, like Bluetooth/Wi-Fi data and surveillance camera data, are quite limited that they cannot cover most of the freeway system. Thus, they are not used in this study. Detailed information on each data source is described in the following subsection.

8.5.1.1 *Freeway Loop Data*

Inductive loop detectors are widely used to monitor freeway performance in the United States because of their reliability and durability (Klein et al., 2006). An inductive loop detector is a conductive coil embedded in the pavement, and it detects a moving vehicle passing over it with

electromagnetics. Speed, volume and occupancy are three key indicators that traffic detectors can collect during a fixed time interval.

8.5.1.2 *NPMRDS Data*

The National Performance Management Research Data Set (NPMRDS) data is used by states to monitor system performance. The INRIX data was selected as the current NPMRDS data provider since 2017 and the HERE data was the previous provider. INRIX combines multiple data sources, including Global Positioning System (GPS)-equipped devices and cell phones and aggregate heterogeneous speed data based on a series of statistical models. INRIX data cover almost the entire roadway network in Washington, including freeways, highways, and most arterials and side streets. INRIX has adopted the Traffic Message Channel (TMC) code, which is used to identify a specific road segment. The INRIX speed data were aggregated into 5-minute intervals.

Similar to INRIX data, HERE data combines data sources from multiple categories, including phone and auto GPS navigation devices. HERE data are collected separately from trucks and other vehicles, thereby making it possible to provide data for both trucks and passenger vehicles. HERE also adopted the Traffic Message Channel (TMC) as its base network, but the TMC network used by HERE is slightly different from that of INRIX. For each TMC, instead of providing speed data, HERE provides travel time data. The time interval of HERE data is also 5-minute.

8.5.1.3 *Incident Data (WITS data)*

The Washington State's Incident Response (IR) Team collects and maintains traffic incident data in the Washington Incident Tracking System (WITS). WITS data includes most incidents that happen on freeways and Washington state highways. For each incident, the Washington State IR team logs details such as incident location (route and milepost), notified time, clear time, and closure lanes.

8.5.1.4 *Weather Data*

The weather data is retrieved from WSDOT. The retrieved raw data includes the information of precipitation type, precipitation intensity, precipitation start and end time. Each weather station is associated with a latitude and longitude pair. In this case, weather data can be visualized on a mapping system.

8.5.2 *Experimental Settings*

In this study, all the spatiotemporal datasets characterizing roadway traffic properties are stored in Microsoft SQL server database on a Windows Server 2012 after pre-processing. Their associated roadway geometric datasets are stored in PostgreSQL database on a Linux Server. The spatial database extender, PostGIS, is used for supporting geospatial queries during the data integration process. All the experiments are tested on a Windows 10 computer with 32GB memory.

In this study, the WSDOT 24k map is selected as the referencing layer for the whole data integration framework. The experimental settings of the framework's four components are listed as follows: On-road segment-based data integration: The HERE data is considered as a representative on-road segment-based data and the HERE map layer is considered as the conflated layer. The proposed IMC algorithm for on-road segment-based data integration is implemented mainly using Python, SQL, and PostGIS. In the map conflation process, 10482 referencing map segments and 28007 HERE segments are utilized for the segment matching process. The initial distance thresholds D of the IMC algorithm is set as D=0.0005, which is around 40 meters. The unit of D is an spatial reference identifier (SRID) equaling 4326 in the PostGIS (Ramsey and others, 2005) environment and A is initialized as 0.25 radian. The a linearly increasing function, Increase(D, A), which contains two steps, is defined as

$$\begin{cases} A = A + \Delta a & \text{if} A \leq 0.65 \\ D = D + \Delta d, \quad A = 0.25 & \text{if} A > 0.65 \end{cases} \tag{8.2}$$

where Δa = 0.2 and Δd = 0.0001. The detailed experimental settings for other categories are listed as follows:

- Off-road segment-based data integration: The experimental settings are identical to that of on-road segment-based data integration.
- On-road point-based data integration: Since this type of data containing linear referencing information is inherently integrated into the referencing layer, no setting is needed.
- Off-road point-based data integration: The proposed Voronoi polygon method is implemented mainly based on SQL and PostGIS.

Table 8.2: Data integration performance of the four categories of data

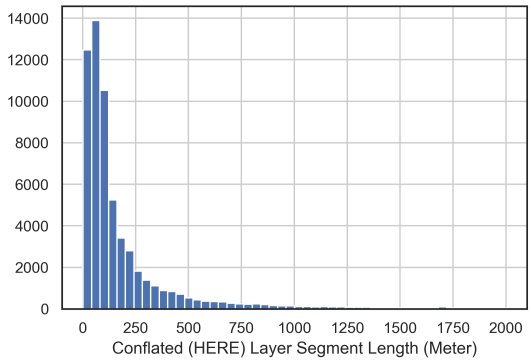
Data type	Dataset	Data Scale	Percentage of Integrated data	Running Time
On-road Segment-based	HERE data	28007 segments	95.4%	18.2 minutes
Off-road Segment-based	Verizon data	93 segments	100%	2.3 seconds
On-road Point-based	Incident data	51117 incidents	100%	Automatically
Off-road Point-based	Weather data	124 stations	100%	0.2 seconds

8.5.3 *Data Integration Performance*

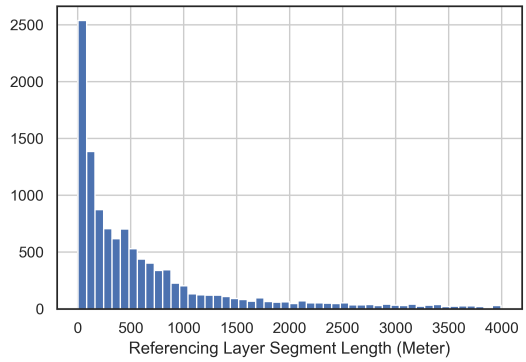
The integration performance of the four types of data are shown in Table 8.2. Except for the on-road segment-based data, all other types of data are 100% integrated. The on-road segment-based data integration method also performs very well that more than 95% of the segments overlapping to the referencing layer automatically integrated. The running time of the on-road segment-based data integration is around 18 minutes. Considering that processing and integrating all these datasets may take days or weeks in practice, the integration of all the four types of data is very fast. Although the incident data contains more than 50000 records, the on-road point-based data does not need any integration method, because it contains the referencing information, including route number and milepost. Due to the integration of both on-road and off-road segment-based data are designed based on the proposed IMC algorithm, the IMC algorithm is analyzed in the next section taking the on-road segment-based data as an example. In summary, the map conflation algorithm works well in terms of the time consumption and the conflation accuracy for the roadway segments in the whole state.

8.5.4 *Analysis of Map Conflation Algorithm*

In this section, the proposed map conflation algorithm is analyzed taking the off-road segment-based data (HERE data) as an example. Figure 8.10 shows the distributions of the length of roadway segments of conflated and referencing layers. The trends of the two sub-figures are both similar to Poisson distribution. However, the number of segments in the conflated layer is far more than that in the referencing layer, since the HERE data covers urban corridors and the referencing layer only covers freeways.



(a) Length distribution of conflated segments

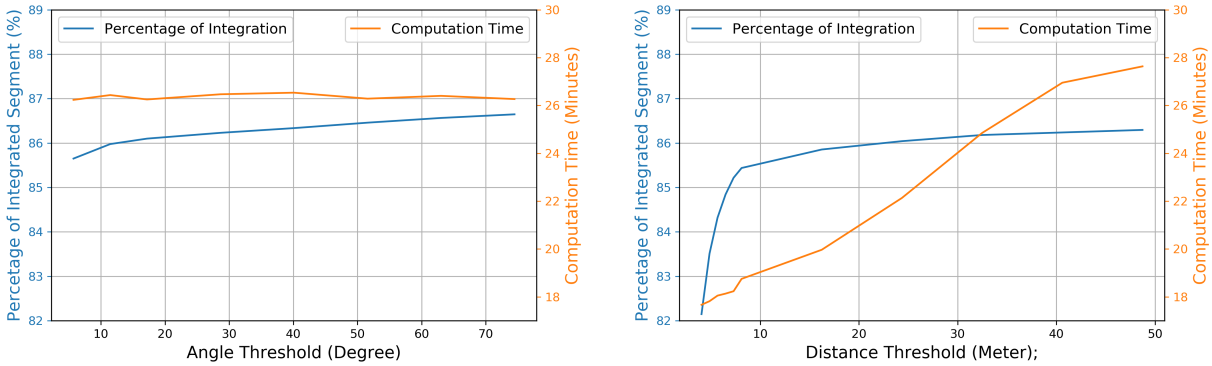


(b) Length distribution of referencing segments

Figure 8.10: Length distributions conflated (a) and referencing segment (b)

The efficiency of the conflation algorithm is measured in terms of percentage of integrated segments and computation time in this study. There are two influencing factors as described in the methodology section, i.e. angle and distance thresholds between conflated segments and referencing segments. Figure 8.11 (a) shows the integration performance when the distance threshold is fixed as 40 meters and the angle threshold increases from 5 degree to 75 degree. The integrated percentage increases slightly and the computation time curves almost does not change. Thus, although increasing angle threshold to large values will not result in more computation time, it will not improve map conflation performance. Figure 8.11 (b) illustrates the integration performance when the angle threshold is fixed as 0.25 radian $\approx$ 28.6 degree and the distance threshold increases from 5 meters to around 50 meters. Unlike the flat curves in Figure 8.11 (a), the running time curve keep increasing with the increase of the distance threshold. However, the

integration percentage curve shows that when the distance threshold is large than 10, it is hard to improve the integration performance by increasing the distance threshold.



(a) Conflated efficiency w.r.t. angle threshold

(b) Conflated efficiency w.r.t. distance threshold

Figure 8.11: Map conflation algorithm efficiency analysis

Since the HERE geometric roadway layer is complex in terms of the number and topology of the segments, some segments inevitably need to be post-processed manually. After the post-process, the on-road segment-based data can also be fully integrated. Figure 8.12 shows an example of the conflated HERE layer. The HERE layer covers most of the freeways and some arterials and urban streets, while the referencing layer only contains freeways in Washington State. Thus, the generated linking roadway layer are mostly located on freeways, shown by green links in Figure 8.12 In the rural areas, some corridors are not covered by both referencing and conflated layers, and therefore, they are not included in the linking layer.

8.5.5 *Applications and Case Studies*

Based on the data integration framework, multiple data sources can be combined together by means of a referencing roadway geometric layer. Figure 8.13 demonstrates the architecture of the potential traffic analysis applications based on the data integration framework. Each type of transportation data is connected to the geospatial referencing database via a linkage table or database. The set of the linkage tables/databases are the key acquirement of this study. In the data extraction modules, multiple analysis parameters can be selected for different specific analysis.

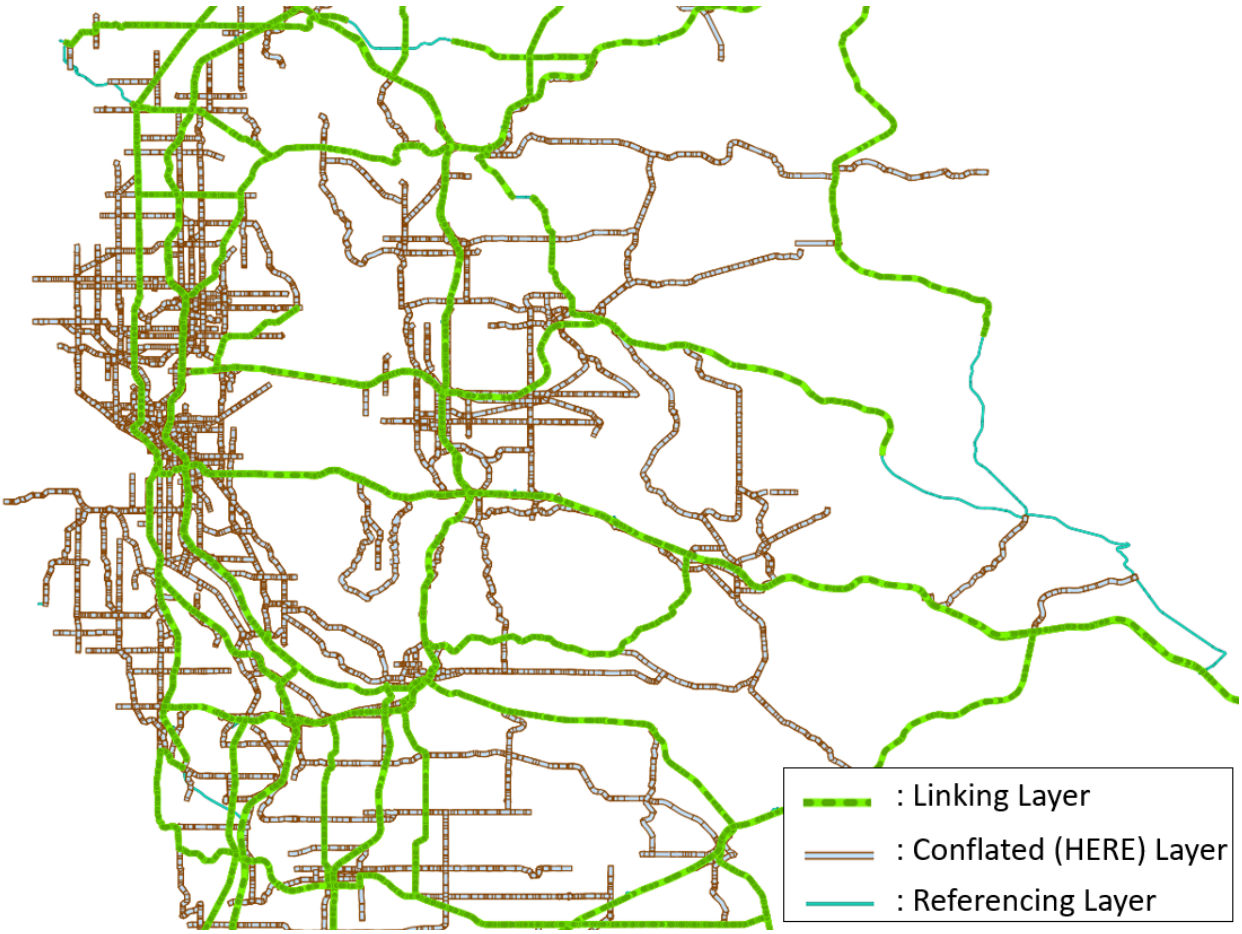


Figure 8.12: Visualized map conflation results for on-road Segment-based Data, taking the HERE data as an example.

The roadway selection can be fulfilled by using multiple advanced spatial query/manipulation functions. The date and time information or other types of parameters can be directly specified from the original transportation databases. Given all the required data extraction parameters are specified, multiple traffic analysis modules can be carried out efficiently using a single system, such as travel time reliability measurement, traffic prediction, traffic network bottleneck detection, level of service analysis, etc. Guided by the SHRP2 L02 product report (List et al., 2014) and L08 product report (Zegeer et al., 2013), several traffic analysis functions are implemented based on this architecture and introduced in the following case studies.

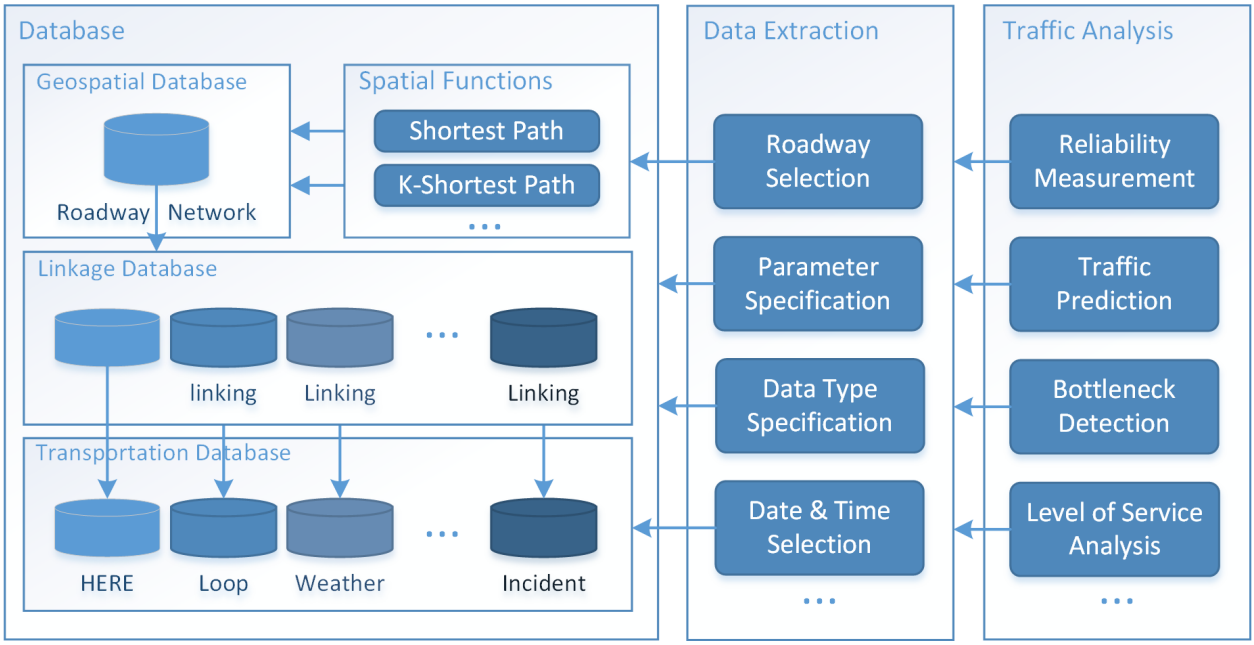


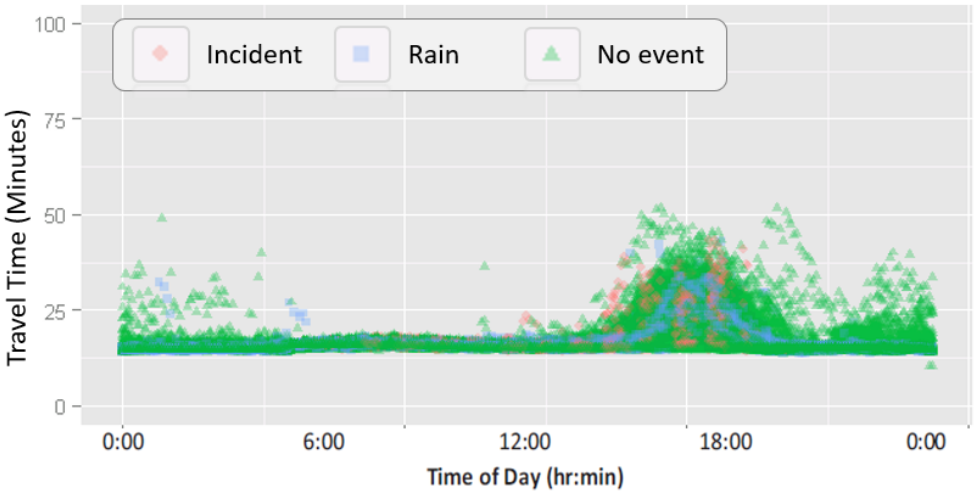
Figure 8.13: Architecture of applications based the multi-source data integration framework.

8.5.5.1 *Case Study: Travel Time Analysis*

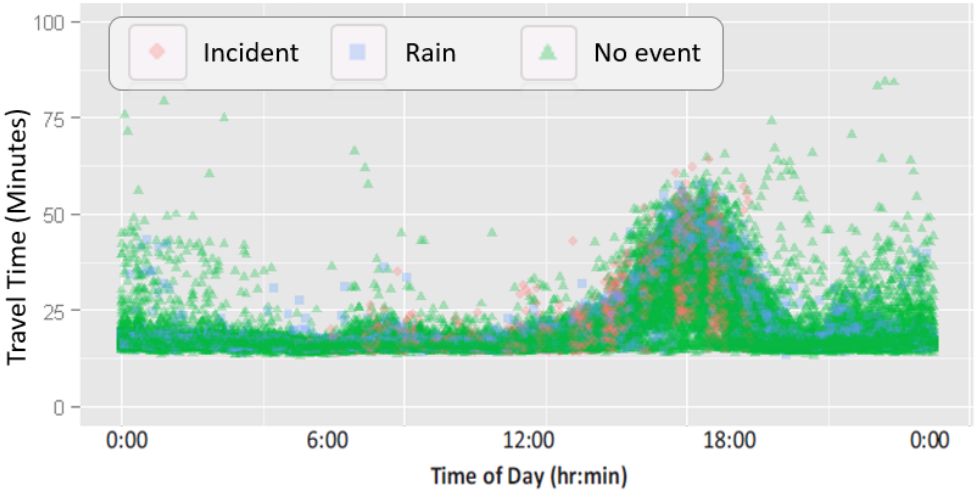
Given multiple transportation data integrated, variations in travel time in different datasets can be compared. Figure 8.14 demonstrates the variations in travel time by time of day measured by loop detector data and HERE data, respectively, in 2015. The Study area is the HOV lane on I-405 from Bellevue to Lynnwood in Washington State. It can be found that the variations in travel time in the two datasets have similar patterns. The travel times influenced by incident and weather are also compared, as shown by red and blue colors, respectively. Since the geospatial representations of HERE data and loop detectors are identical after the map conflation process, obvious differences in the two travel time distributions can still be observed that the average and variance of the travel time measured by HERE data are both higher than that measured by loop detector data. Hence, data quality can efficiently be compared for different data sources based on the proposed data integration framework.

8.5.5.2 *Case Study: Performance Measurement*

With the help of the proposed traffic data integration framework, a variety of reliability performance measurement metrics, such as reliability rating, planning time index (PTI), and 80th



(a) Variations in Travel Time by Time of Day in **Loop Detector Data**



(b) Variations in Travel Time by Time of Day in **HERE Data**

Figure 8.14: Variations in travel time by time of day measured by (a) loop detector data and (b) HERE data. The Study area is the HOV lane on I-405 from Bellevue to Lynnwood in Washington State.

percentile travel time index (TTI), can be analyzed based on different data sources. Figure 8.15 illustrates the PTI distribution by time of day on I-405 from Bellevue to Lynwood in 2015, in which three datasets, the GP lane loop detector data, the HOV lane loop detector data, and the HERE data are compared. PTI is defined as 95th percentile travel time divided by the free-flow travel time. The PTI distribution of HERE data and GP lane loop detector data are similar, and the PTI distribution of HOV lane loop detector data has more obvious evening peak. Based on the data integration framework, multi-source based data analysis can be easily and efficiently fulfilled by

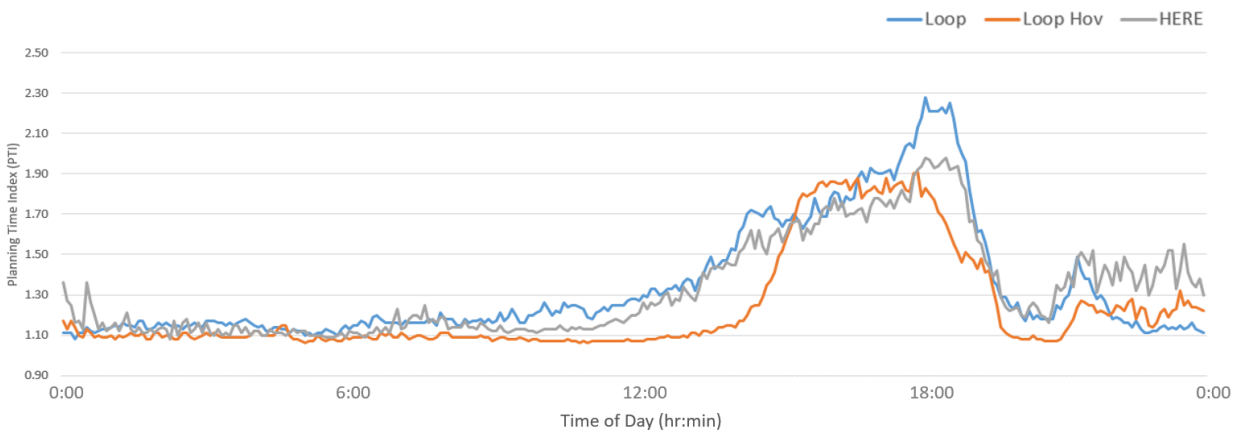


Figure 8.15: Planning Time Index (PTI) distribution by time of day on I-405 from Bellevue to Lynwood. Loop detector data on GP lanes and HOV lane, and HERE data are compared.

transportation data analysis tool. The travel time reliability analysis module is implemented on the DRIVE Net platform.

8.5.5.3 *Case Study: Web-based Analysis Tool*

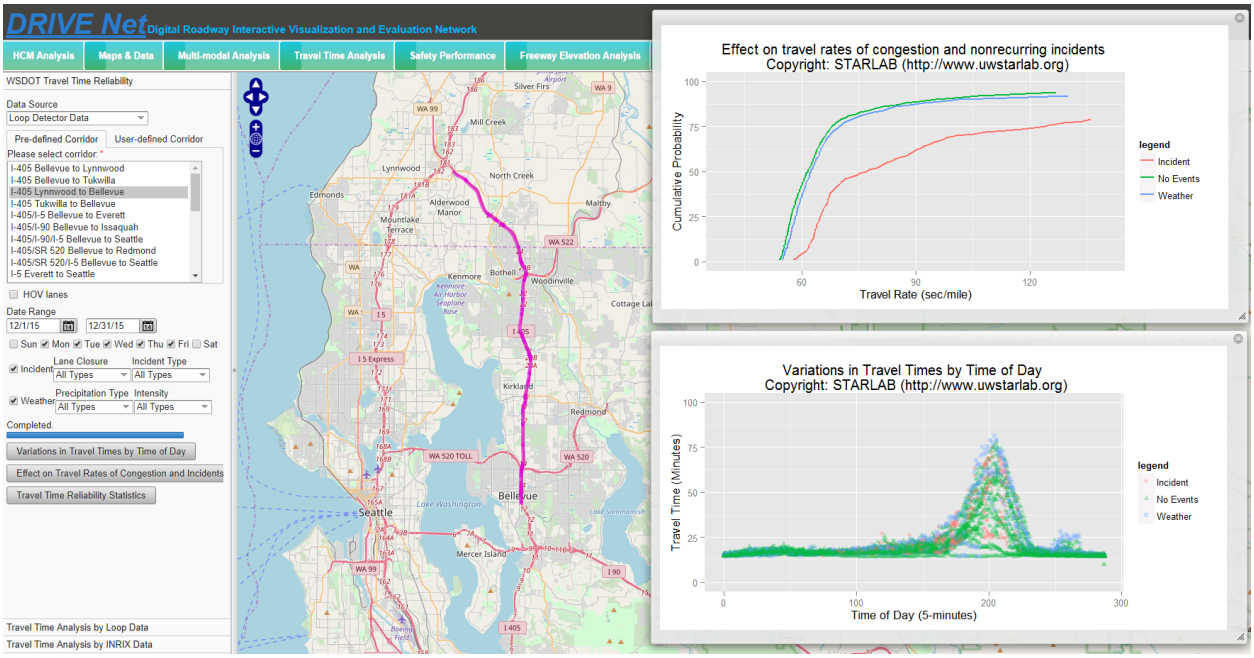


Figure 8.16: The interface and functionality modules of the web-based transportation data analytic platform. Two figures generated showing travel rates, i.e. travel time reliability, and travel time variations are demonstrated on the platform interface.

The proposed traffic data integration framework is fully developed and implemented on an online interactive publicly-accessible transportation data platform, DRIVE Net [2]. On the DRIVE Net platform, multiple traffic analysis modules, such as travel time analysis, travel time reliability analysis, transportation emission analysis, and traffic safety analysis, are implemented, shown in Figure 8.16. Due to the framework integrated multiple data sources together via a uniform referencing layer, the referencing layer is utilized by the tool to provide convenient route selection modules, including pre-defined route selection and user-defined route selection. Besides, the tool can provide customized analysis periods by letting users select the date and day of the week. Further, the integrated datasets, which have great impacts on travel time reliability, such as weather and incident data, can be easily combined in the analysis module. By implementing the multi-source transportation data integration framework, using such an efficient transportation data analysis tool can dramatically reduce workforce and computation investment and cost.

8.6 CHAPTER SUMMARY

In this chapter, to solve the main hurdles of analyzing multi-source transportation data, a transportation data integration framework based on a uniform roadway referencing layer is proposed. Four types of traffic analysis related data, including on-road segment-based data, on-road point-based data, off-road segment-based data, and off-road point-based data, are categorized from the perspective of sensor locations and sensing areas to deal with multi-source traffic data more efficiently. Meanwhile, an iterative map conflation algorithm is proposed mainly for integrating on-road segment-based data. By implementing the data integration framework on a transportation big data platform, several real-world case studies and applications for traffic analysis are realized and presented. The experimental results show that the proposed framework performs well in terms of accuracy and efficiency.

The main advantages of the proposed framework are flexibility and adaptability. With the data integration framework, when a new data source needs to be integrated into the data analytical system, the same referencing layer can be utilized to integrate and manipulate the new data source. In the experiments, nearly all types of data can be 100 percent integrated. However, due to data

2 http://www.uwdrive.net/

sources have different data formats, some special cases inevitably exist during the integration process. The framework sometimes cannot automatically integrate 100% of all data sources, taking the HERE data integration as an example. In spite of the existence of extreme cases, the framework can still work perfectly after manually checking and correcting the miss-integrated data to facilitate further analysis in research work and real applications.

The main contribution of this chapter is to propose the data integration framework, which is of practical importance, although there is still much room for improvement in the methodology sections. In the future, studies will explore more intelligent methods to build connections between the geospatial metadata of different data sources and design more efficient and accurate map conflation methods. When the further complicated analysis is required in the future, more data sources, such as roadway elevation data, vehicle trajectory data, and connected vehicle related data, will be tested to integrated based on the proposed framework. [3]

3 This chapter is a slightly modified version of "**Establishing Multi-source Data Integration Framework for Transportation Data Analytics**" published in *Journal of Transportation Engineering, Part A: Systems* and has been reproduced here with permission from ASCE. (doi: https://doi.org/10.1061/JTEPBS.0000331)

9

TRANSPORTATION ARTIFICIAL INTELLIGENCE PLATFORM

9.1 OVERVIEW

The advancement of new smart traffic sensing, mobile communication, and artificial intelligence technologies has stimulated significant growth in the volume and variety of transportation data. A huge volume of newly generated transportation data is playing an important role in modern smart transportation and smart city research and applications. Although transportation data has great potential to enhance smart transportation applications in the development of smart cities, we are still facing challenges regarding how to fully and properly use these various massive transportation data sets. In recent years, new transportation sensing technologies have constantly emerged, which provide more options to transportation practitioners and researchers to collect necessary transportation data. Due to the diversity and variety of transportation data, both transportation practitioners and researchers are facing substantial opportunities and challenges.

Before the "big data" term was widely used, most of the previous transportation applications and studies were designed and conducted based on real data. However, real transportation problems are normally intricate and related to many unexpected influential factors. The sizes of the datasets used in previous studies are usually too small to reflect the real-world complexity of these problems. Meanwhile, many methodologies in the transportation field are built based on complicated mathematical models without comprehensive validation from real transportation datasets. In this way, biased models may be generated in the process of understanding the core of transportation problems. Thus, it is critical to find proper and practicable ways to comprehensively utilize it.

In recent years, the increased computation power enabled by advanced hardware and the rise of artificial intelligence (AI) technologies, especially in the deep learning field, have provided great opportunities to comprehensively utilize transportation big data. The deep learning-based models with powerful non-linear fitting capabilities can provide great power for dealing with complicated transportation problems that cannot be easily solved by classical methods. However, the deep learning-based models are normally highly flexible, and thus, the design of these models needs to be customized depending on specific problems and datasets. For a specific problem, like traffic prediction, the design of a model structure will directly affect the model performance. In addition, although transportation is a proper scientific domain for developing and applying novel deep learning models, most of the existing deep learning models are not originally designed for transportation problems. Hence, developing novel or customized deep learning models for classical transportation problems is another challenge for transportation practitioners and researchers.

To efficiently overcome these challenges, developing multiple deep learning models and comparing their performance by testing on existing standard datasets can stimulate the emergence of new methodologies. Previous chapters introduce several deep learning-based traffic prediction models. Additionally, tremendous deep learning-based traffic prediction models with various neural network structures has been proposed in recent five years. However, those models are tested on different datasets collected from everywhere in the world, and the majority of the proposed models did not publish the source code. Thus, most of the existing studies cannot efficiently facilitate the future research. It will be hugely beneficial for the traffic prediction research field to build a platform that can provide the comparison results of those models' prediction performance based on standard traffic prediction datasets. With this idea, this study aims to build a platform with standard procedures to train and test different deep learning models to solve various transportation-related problems. The testing results can be easily compared to assist in selecting the more effective models for further studies or implementations.

9.1.1 *Contribution and Organization of the Chapter*

The originality and contribution of this work can be summarized as follows:

1. We develop an transportation AI platform for solving challenging transportation problems that require large-volume high-dimensional transportation data and complex models.

2. A second version of the transportation AI platform, TraffiX.ai, is established for sharing datasets and evaluating novel state-of-the-art deep learning-based traffic forecasting models.

3. Four standardized datasets are formatted, documented, and shared for evaluating traffic forecasting models.

4. The evaluation results of state-of-the-art traffic forecasting models are shared as benchmark to facilitate future research. The related source code and training parameters are also published to further enhance the reproducibility of the existing deep learning-based traffic prediction models.

The rest of this paper is organized as follows: Section 9.2 describes the related studies on artificial intelligence platforms and traffic forecasting. Section 9.3 mainly describes the functionality of the two versions of the proposed transportation artificial intelligence platform. Section 9.4 demonstrates traffic forecasting models' testing results as benchmark for future research. The concluding remarks are presented in the Section 9.5.

9.2 RELATED WORK

Data science is a set of fundamental principles that support and guide the principled extraction of information and knowledge from data (Provost and Fawcett, 2013). The core task of data science is to extract knowledge from data via technologies that incorporate these principles. Accordingly, transportation data science can be realized by applying the fundamental principles of data science in the transportation field. Specifically, transportation data science can be defined as the computationally intensive investigation of transportation issues using immense data sets. Artificial intelligence, especially deep learning methods, applied to the analysis of emerging transportation datasets, has brought new power to the transportation research field. To adequately bring the power of AI to the transportation fields, this study aims to build a transportation AI platform to stimulate and enhance the design of novel transportation-oriented deep learning

algorithms. In this section, a number of AI-based transportation applications and platforms are introduced. As a typical complicated transportation problem, traffic prediction studies that can comprehensively incorporate deep learning methods are also introduced.

9.2.1 *Artificial Intelligence Applied in Transportation*

Artificial intelligence (AI) has the potential to solve problems that are hard for traditional methods to address, and various AI methods have achieved state-of-the-art performances in speech recognition, visual object recognition, object detection and many other domains (LeCun et al., 2015). Some AI-based methods even surpass human-level performance on some specific problems (Mnih et al., 2015). With increasing population, vehicles, and mobility demands, improving the safety, efficiency, and sustainability of the transportation system remains a challenge, and traditional methods may not be able to fully address it. To overcome these issues, an increasing amount of studies have been conducted to apply AI-based methods to solve complicated transportation problems including traffic signal control (Abdulhai et al., 2003; Arel et al., 2010; Li et al., 2016), traffic prediction, and microscopic traffic modeling (Wang et al., 2017; Zhou et al., 2017; Zhu et al., 2018).

9.2.2 *Existing Transportation Data/AI Platforms*

Well-designed platforms or systems are capable of properly utilizing the existing immense transportation data sets and AI methods. For traffic signal control the DeepDrive platform [1] is developed to provide adaptive traffic signal control based on deep reinforcement learning. For traffic congestion detection and traffic prediction, PTV Optima [2] can generate traffic prediction information for up to 60 minutes in the future. Traffic congestion can be detected by the speed and traffic flow detected in the field or calculated from roadway traffic states data (e.g., floating car and license plate identification data). The Miovision TrafficLink platform [3] is developed to

1 https://deepdrive.berkeley.edu/
2 https://www.ptvgroup.com/en-us/solutions/products/ptv-optima/
3 https://miovision.com/trafficlink/

assist traffic engineers to create more responsive and efficient traffic networks. TIMON (Osaba et al., 2016) is a European research project, whose main objective is to provide real-time services through a web-based platform and a mobile application for drivers. Microsoft research team also pioneered the use of machine learning methods to build predictive models for traffic (Amershi et al., 2019). The developed models can infer and predict traffic flow at different time periods in the future based on the analysis of large amounts of data over months and years.

9.3 TRANSPORTATION ARTIFICIAL INTELLIGENCE PLATFORM

The transportation artificial intelligence platform is originally designed to solve cutting-edge transportation problems, and thus, it should have the ability to support hosting multiple types of models and datasets to solve multiple tasks, such as traffic forecasting, data imputation, and vehicle detection, whose performance can be quantitatively evaluated by well-established metrics. However, since more easily-accessible and flexible deep learning packages, such as TensorFlow and PyTorch, are available for public users, developing a platform for developing and running established fixed types of deep learning models is not as urgent as developing a platform to share various state-of-the-art methodologies, standard datasets for specific tasks, and tested results as benchmarks. Thus, we extend the transportation artificial intelligence platform from a computation-centric platform to a model and benchmark sharing platform. In this section, we will introduce the first version of the platform, including its architecture and core functionalities. We will then introduce the extended version of the transportation AI platform, TraffiX.ai, in the next section.

9.3.1 *Architecture*

The transportation AI platform is built based on transportation-related datasets, and thus there should be a data management system to store, query, process, and manage all the datasets to make the modeling process efficient. Because the artificial intelligence methods are mostly neural network-based models and the training and testing of neural networks are quite time-consuming,

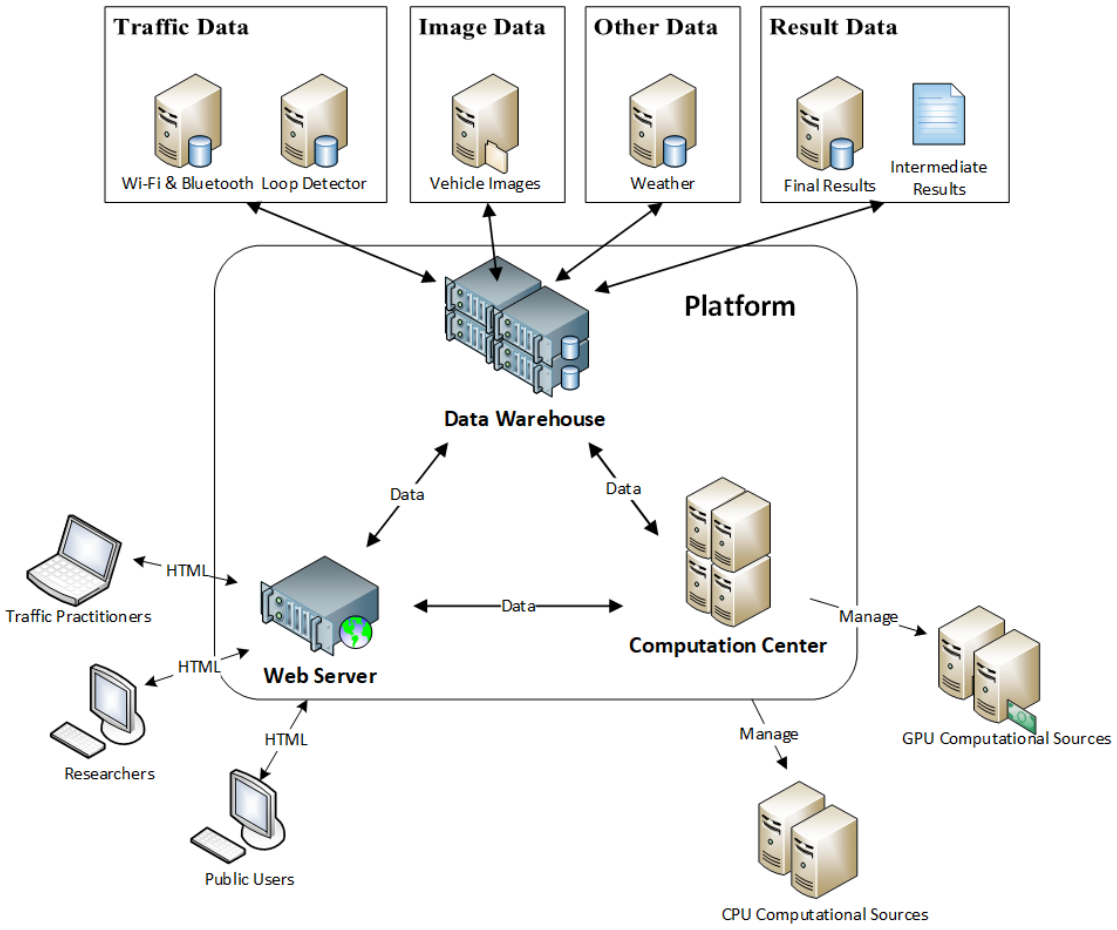


Figure 9.1: Architecture of the Transportation AI Platform.

a computer or a cluster, or even a cloud with the power of conducting multiple training and testing tasks, should be incorporated as a main component of the transportation AI platform. In addition, to let the users conveniently and interactively manipulate the models and datasets on the platform, the platform should also have a user-friendly interface.

Thus, to fulfill the requirements, the first version of the transportation AI platform mainly contains three main components, i.e. a data warehouse, a web server, and a computation center, as shown in Figure 9.4. All three components are connected and the communications between those components are mainly data transmission. The data warehouse hosting multiple types of databases can manage and provide datasets for solving transportation problems. The computation center uses computers or the cloud which contains multiple graphics processing units (GPUs) capable of efficiently training and testing deep learning models. The web server hosts a website

that allows users to access the developed transportation AI platform remotely and help manage user accounts and training and testing tasks.

9.3.1.1 *Data Management and Database Design*

The data warehouse is capable of hosting multiple types of data, including tabular data, image data, and structured data. The traffic state data, mainly collected from loop detector sensors and other types of traffic sensors, are mostly stored in relational databases in the tabular format. The image or video data normally should be traffic monitoring data for solving problems, such as traffic flow counting or traffic detection. The data describing the traffic network's physical or topological structures are normally stored using various types of databases or files, depending on the types of tasks that need to be solved. Basically, the aforementioned data is used for training and testing models to solve specific transportation problems. However, when a model is trained or tested on the platform, many types of metadata and result data are generated and should also be stored.

For a specific task, the data used for the training and testing process should be identical when different users attempt to adjust hyper-parameters or conduct the training/testing process multiple times. Hence, in this study, the formats of the datasets used for training/testing models are fixed and those datasets are stored as files that can be read using the same data loading procedure. As the training/testing processes are conducted in the computation center, to reduce data communication and efficiently load data to the models, all the well-processed training and testing data are stored as files in the computation center. However, since different users may conduct different tasks and the results of these tasks will be reused and visualized by the transportation AI platform, the query process of these result data should be convenient and flexible. Hence, the task information and task result data are stored in the database in the data warehouse side.

9.3.1.2 *Computation Center*

The computation center has the ability to receive task requests from the web server and execute the specific task. The task results will be sent to the data warehouse to be permanently stored. Since multiple users may use the platform at the same time, multiple task requests may arrive

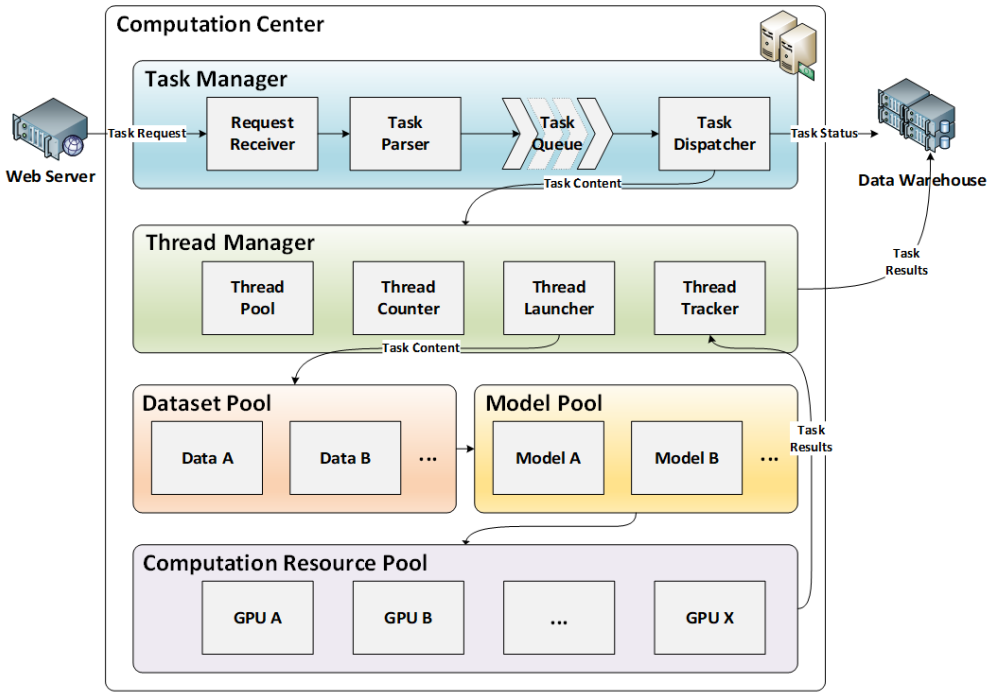


Figure 9.2: Structure of the computation center.

at the computation center at the same time or in a short period of time. It is necessary for the computation center to be able to manage all the received tasks and execute those tasks in a reasonable order based on the amount of computation resources the computation center has. To reasonably manage all the received tasks, the computation center is built based on several important components, including a task manager, a thread manager, a dataset pool, a model pool, and a computation resource pool, as shown in Figure 9.2.

The task manager is the core component of the computation center, which consists of a request receiver, a task parser, a task queue, and a task dispatcher. The request receiver acts as a server that monitors all the task requests via the http request technologies. When a task request is received, the encoded context of the task request will be decoded by the task parser. After the task content is successfully decoded without errors, the task request will be input into a task queue, which will attempt to first launch the earliest-arrived task request. The main reason to add a task queue in the task manager is that when the number of requested tasks is larger than the amount of available GPUs on the platform, the task requests that arrived last need to be stored in the task queue. When a GPU is available to be used to train or test deep learning models, the first task will

be removed from the task queue and passed to the task dispatcher. The task dispatcher will input the task content to the thread manager to execute the task by creating a new thread.

The thread manager manages the threads to optimize the distribution of computation resources among the tasks. The thread manager contains a thread pool, thread counter, thread launcher, and a thread tracker. The thread manager has a maximum amount of threads in the thread pool and the thread counter is used to count how many threads are operating. The maximum number of threads is based on the amount of available computation resources. After a thread is created, the thread launcher will start to load the correct dataset and model from the dataset pool and the model pool, respectively, to execute the task. When the thread is created, the thread launcher will also assign a GPU from the computation resource pool to the task. The dataset will be loaded to the memory and the model will be initialized in the assigned GPU to start the training process. At the same time, the thread tracker will monitor the GPU's status to help dispatch tasks.

While the task is executing, the intermediate results and final outputs will be sent back to and stored in the data warehouse. In the end, via the thread manager and the task manager, the status of the computation resource pool will be updated to assist with the arrangement of upcoming tasks.

9.3.1.3 *Interface*

The web server is the interface that connects the platform and users. Most web applications are designed based on the Model-View-Controller (MVC) framework. The architecture of the transportation AI platform is also similar to the MVC framework which contains the view, the controller, and models. However, since the transportation AI platform has the computation center, the web server can be simplified as the computation center is responsible for the modeling and computation jobs. Thus, the web server mainly serves for the view and controller parts.

The user interface (UI) of the transportation AI platform is designed to be simple and clear. Thus, the whole UI of the platform is designed as a system administration console providing all the required functions directly on the web page. The logical level of the transportation AI platform is controlled based on the procedure for efficiently and easily creating a new task on the platform. The procedure for creating new tasks is introduced in detail in the following section. Based on the designed architecture and the introduced key technologies, the core functions of the transportation

AI platform are developed in this study. In this section, three main functions are demonstrated by showing the functions' user interface (UI). It should be noted that the transportation AI platform is designed as a prototype and the UI will be adjusted or redesigned with the future development process.

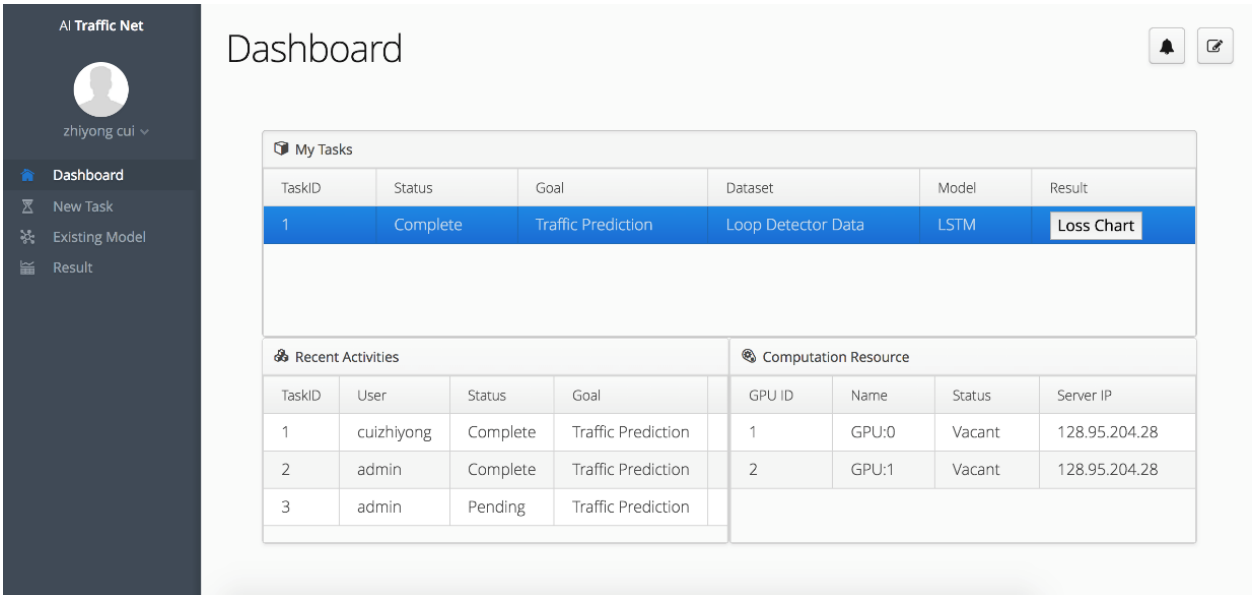
9.3.1.4 *Platform Demonstration*

The established platform has a dashboard to display the current status of the platform, including the running tasks, the historical tasks, and the status of the computation center. Figure 9.3 (a) shows the UI of the dashboard. The left section of the UI is the menu of the platform and the right section is the functional panel that displays the platform information. The top part of the panel shows the existing tasks launched by the users and the left bottom part of the panel shows the tasks of other users. The right bottom part of the panel shows the status of the GPUs in the computation center. The user can also check the status of the existing task by visualizing the training and validation loss, as shown in Figure 9.3 (b). Since this version of the Transportation AI Platform is no longer maintained and publicly accessible, the details of core architectures, technologies and used software will not be introduced here. For more information, the readers can refer to (Wang et al., 2019c).

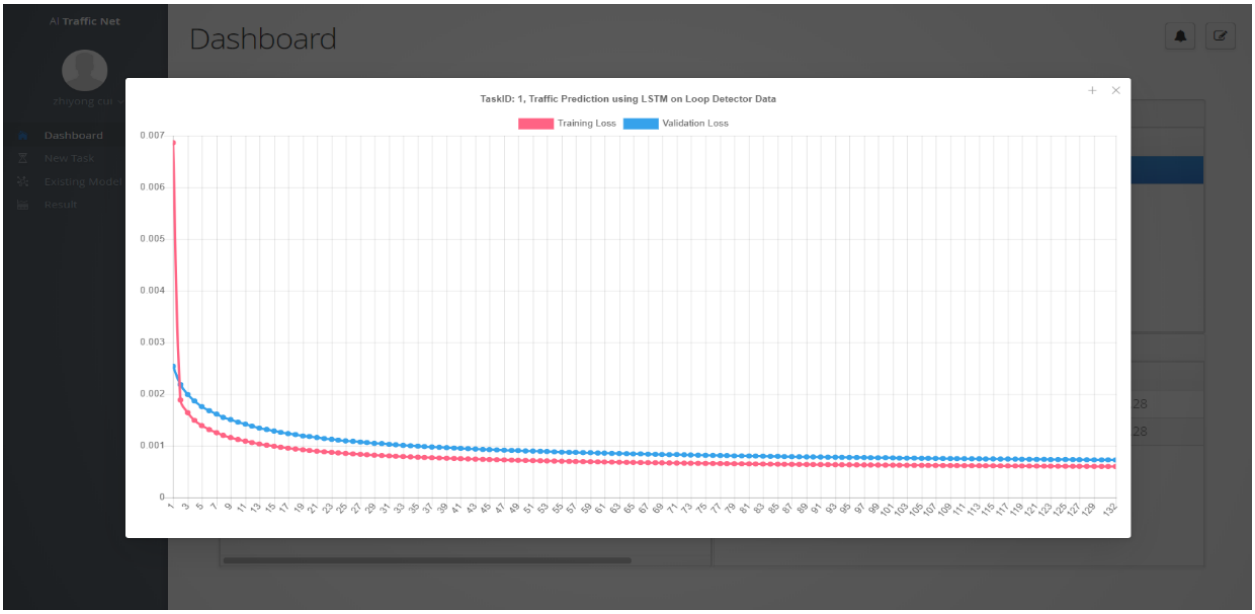
9.3.2 *TraffiX.ai*

The TraffiX.ai [4] platform is established by extending the first version of the Transportation AI Platform for sharing standard datasets and evaluating state-of-the-art traffic forecasting models to speed up the research progress in the short-term deep learning based traffic prediction research field. The architecture of this platform is different from the first version. It focus more on evaluating more established or published deep learning models based on the unified/standard datasets. The tested results of different models and their related hyper-parameters will be presented on the platform to give the users an clear overview of pros and cons of those evaluated models. The TraffiX.ai platform is more like a website with several modules, including a online tutorials on

4 http://ai.uwstarlab.org/



(a) Platform dashboard.



(b) Overview of the result of a training task on the platform dashboard.

Figure 9.3: Demonstration of the first version of the Transportation AI Platform

short-term traffic prediction, existing datasets, and evaluated model performance on different traffic prediction tasks as benchmarks.

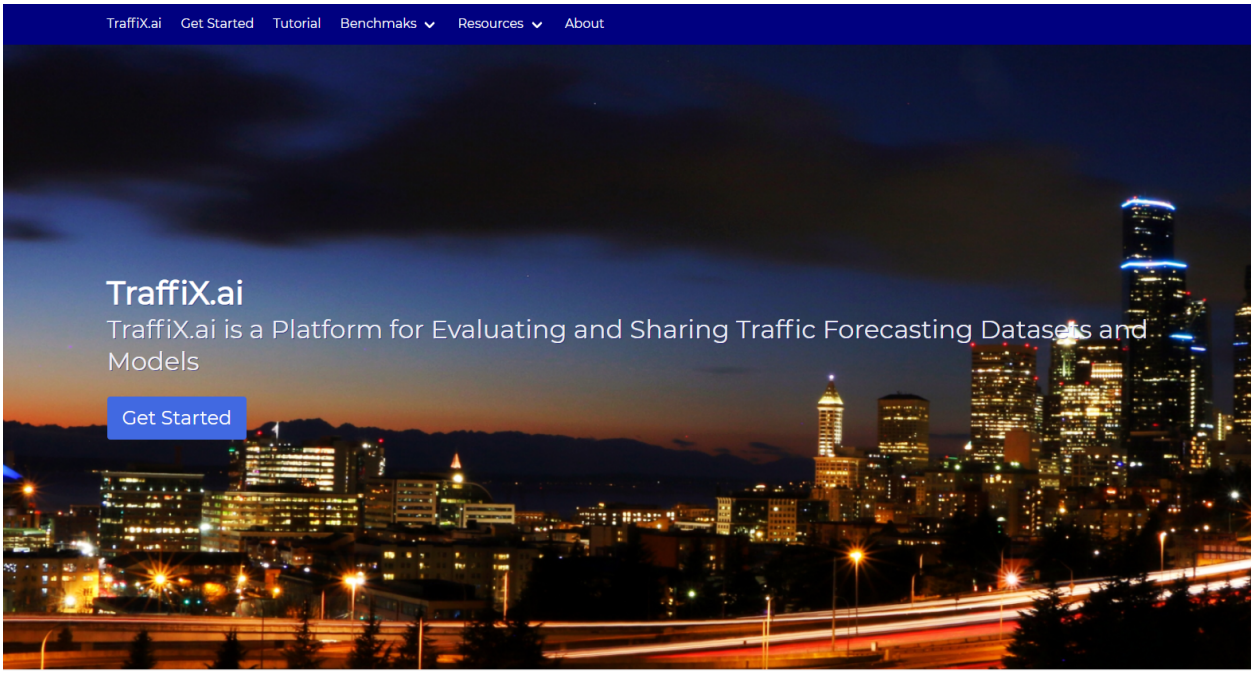


Figure 9.4: New version of the transportation artificial intelligence platform, TraffiX.ai.

9.3.2.1 *Datasets*

TraffiX.ai provides several datasets that have been mentioned in the previous sections of this dissertation. They are all network-wide traffic state (speed) datasets. The details of the four major datasets are listed below:

- LOOP-SEA: The second data set is the loop detector data set which is collected by inductive loop detectors deployed on the freeway system. In this data set, the loop detector data covers four connected freeways in the Greater Seattle areas, including I-5, I-90, I-405, and SR-520. The raw data contains three basic traffic flow characteristics, including traffic speed, volume, and density. The time interval in the raw data is 20 seconds. 323 traffic sensing locations are selected and integrated to form the loop detector data set. The time interval is integrated into 5-minute intervals. This dataset is published at GitHub [5] and also Zenodo [6].

- INRIX-SEA: This dataset provided by WSDOT originates from the Federal Highway Administration (FHWA)'s National Performance Management Research Data Set (FHWA, 2019).

5 https://github.com/zhiyongc/Seattle-Loop-Data
6 https://doi.org/10.5281/zenodo.3258904

This data set contains the speed data of roadway links in the Seattle downtown area, which is mostly collected by probe vehicles. In this area, the road network is very complex in that it contains principal arterials, minor arterials, one-way streets, freeways, ramps, express lanes, etc. This dataset covers the year 2012 and the time interval is 5-minute. The roadway network contains more than 1000 roadway links, but we select the largest connected roadway network containing 745 segments in the experiment.

- METR-LA: This traffic dataset contains traffic information collected from loop detectors in the highway of Los Angeles County (Jagadish et al., 2014). We select 207 sensors and collect 4 months of data ranging from Mar 1st 2012 to Jun 30th 2012 for the experiment. The total number of observed traffic data points is 6,519,002. This dataset is firstly published in GitHub [7].

- PEMS-BAY: This traffic dataset is collected by California Transportation Agencies (CalTrans) Performance Measurement System (PeMS). We select 325 sensors in the Bay Area and collect 6 months of data ranging from Jan 1st 2017 to May 31th 2017 for the experiment. The total number of observed traffic data points is 16,937,179. This dataset is also firstly published by Li et al. (2018) in GitHub [8].

Given that this four datasets have similar data formats, they still need to be further processed to be the standard datasets for traffic forecasting modeling evaluation. TraffiX.ai thus provides several online tutorials to introduce the required data pre-processing and data formulation process prior to the training and testing procedures [9].

9.4 BENCHMARK

One of the main targets of developing TraffiX.ai is to facilitate the research in the network-wide traffic prediction field. Given several well-formatted standard datasets, the newly proposed models can be evaluated and compared. Since the provided datasets are collected from different cites, the roadway geometric structures are totally different. In addition, the four dataset covers different

7 https://github.com/liyaguang/DCRNN

8 https://github.com/liyaguang/DCRNN

9 https://zhiyongcui.com/TRAFFIX_Web/pages/tutorials/

types of roadways, such as freeways and arterials. Hence, the four datasets have totally different spatial coverage and temporal patterns which can facilitate the validation of newly proposed models' generalization property. To testing the existing traffic prediction models' performance, three widely used metrics are adopted, including mean absolute error (MAE), mean absolute percentage error (MAPE), and Root Mean Squared Error (RMSE):

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{9.1}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}|\frac{y_i - \hat{y}_i}{y_i}| \times 100\% \tag{9.2}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{9.3}$$

where $\hat{y}_i$ is the predicted data, y_i is the label data, and N is the size of the samples for testing various methods.

Additionally, the traffic prediction performance is not solely determined by the structure of a deep learning model, it is also highly affected by the hyper-parameters and the training strategies. Thus, in order to consider the tested results as the benchmark, it is necessary to test different combinations of parameter values. To avoid outliers resulted by incidentally failed training, we also need to train and test the existing models with fixed random seed or with different random seeds and pick the best/average performance.

9.4.1 *Performance Results as Benchmark*

Given all the datasets and parameters are fixed, multiple widely used traffic prediction models or algorithms can be tested. TraffiX.ai provided the test results of most RNN related models, including LSTM, GRU, and graph convolutional LSTM, which has already been introduced in Chapter 3 and Chapter 4. Figure **??** shows the test results on the four datasets. Besides the three metric values, the model's name, learning rate, optimizer, and the total number of training epoch are also presented on the TraffiX.ai website as references for future research. Additionally, to

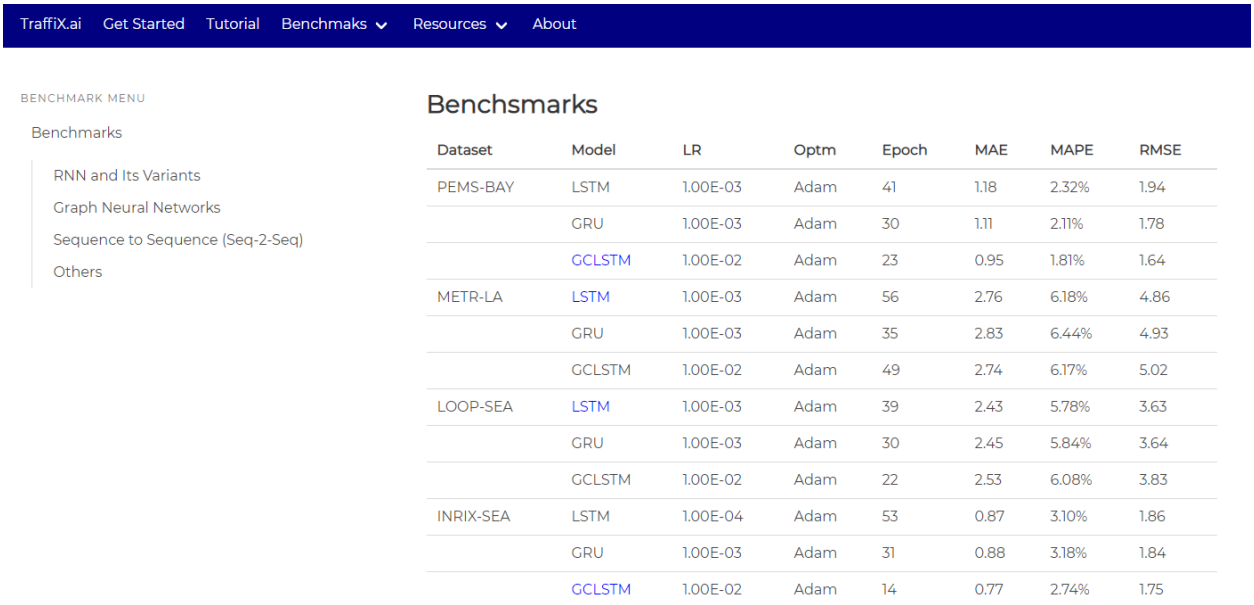


Figure 9.5: Benchmarks of RNN-based models on TraffiX.ai.

enhance the reproducibility of existing research, the training and testing code of each model will be published on this website. Public users can download the code and reproduce the same results on their own machine, as long as they follow the same procedure using the identical environment and parameters. The reproducibility of these compared methods is the most valuable asset of this transportation AI model evaluation and sharing platform.

Deep learning based traffic prediction models have various types of model structures. Figure 9.5 shows the performance of RNN-based method. Another type of popular methods are the sequence-to-sequence (Seq2Seq)models, which are adopted by multiple state-of-the-art models, such as the DCRNN proposed by Li et al. (2018). Seq2Seq models have the superiority of predicting multi-steps of traffic state ahead. Thus, TraffiX.ai also compares Seq2Seq-based models with RNN-based models, as shown in Figure 9.6. More details of the comparison results can be found in the Benchmark section of the platform.

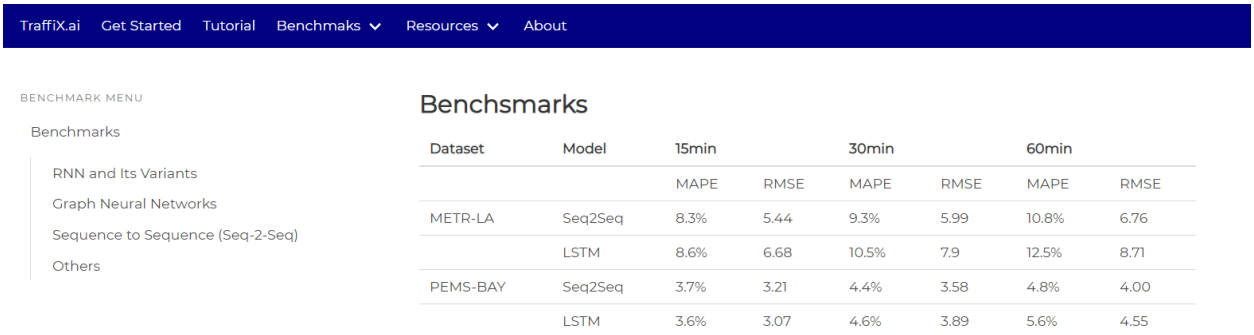


Figure 9.6: Benchmarks of Seq2Seq-based models on TraffiX.ai.

9.5 CHAPTER SUMMARY

The emergence of novel traffic sensing, data communication, and artificial intelligence technologies has greatly stimulated the growth of transportation data. To overcome the challenges brought on by immense transportation data, we propose an artificial intelligence platform for solving challenging transportation problems using large-volume high-dimensional transportation data and complex models. The transportation AI platform is capable of providing standardized datasets and novel deep learning-based models for solving specific pre-defined transportation problems. We also designed a novel architecture for the transportation AI platform to enhance the efficiency of the transportation data processing, management, and communication and increase the computational power of the platform. For a specific problem, the platform offers standardized training and testing procedures to assist in the evaluation of emerging novel methodologies.

In the future, we will focus on displaying the best performance of each model for each specific task on the platform to reduce researchers' duplicated modeling work and stimulate the emergence of novel technologies.

10

10 This chapter is a partially from the tech report "**An Artificial Intelligence Platform for Network-Wide Congestion Detection and Prediction using Multi-Source Data**" published by *Connected Cities for Smart Mobility toward Accessible and Resilient Transportation Center (C2SMART)* and extended based on the new progress of the traffix.ai project. (National Transportation Library: https://rosap.ntl.bts.gov/view/dot/49166)

10

MEASURING NETWORK-WIDE TRAFFIC PERFORMANCE USING TRAFFIC PERFORMANCE SCORE

10.1 OVERVIEW

Traffic performance measurement is incredibly beneficial to both transportation agencies, who use this data as a source of strategy for operations and management, and to the general public, as traffic patterns impact daily life. Many classical traffic performance metrics, including speed, volume, travel time, etc., provide relatively intuitive information about the traffic status of a roadway segment or a specific corridor. However, there are few existing metrics that can indicate network-wide traffic performance. This is especially true when the traffic network scale is large enough to cover an entire city. Further, because most existing traffic metrics only measure one element of traffic stream characteristics, they have difficulty distinguishing complicated traffic scenarios. For example, two roadways with the same traffic speed, one with a high volume and one low volume, at two different times, indicates two entirely different traffic scenarios. Therefore, to overcome the shortcomings of these classical traffic metrics, this study attempts to develop a measurement metric to indicate the network-wide traffic performance in a more comprehensive way.

The first COVID-19 case in the United States was reported in January 2020. Since that time, cases have occurred in all 50 U.S. states. By the end of May, 2020, the U.S. has more than 1,800,000 confirmed cases and more than 100,000 deaths (David, 2020). This pandemic has not only dramatically influenced the social and economic activities, but also hugely affected the transportation system. Traffic patterns in most cities have been entirely reshaped since January,

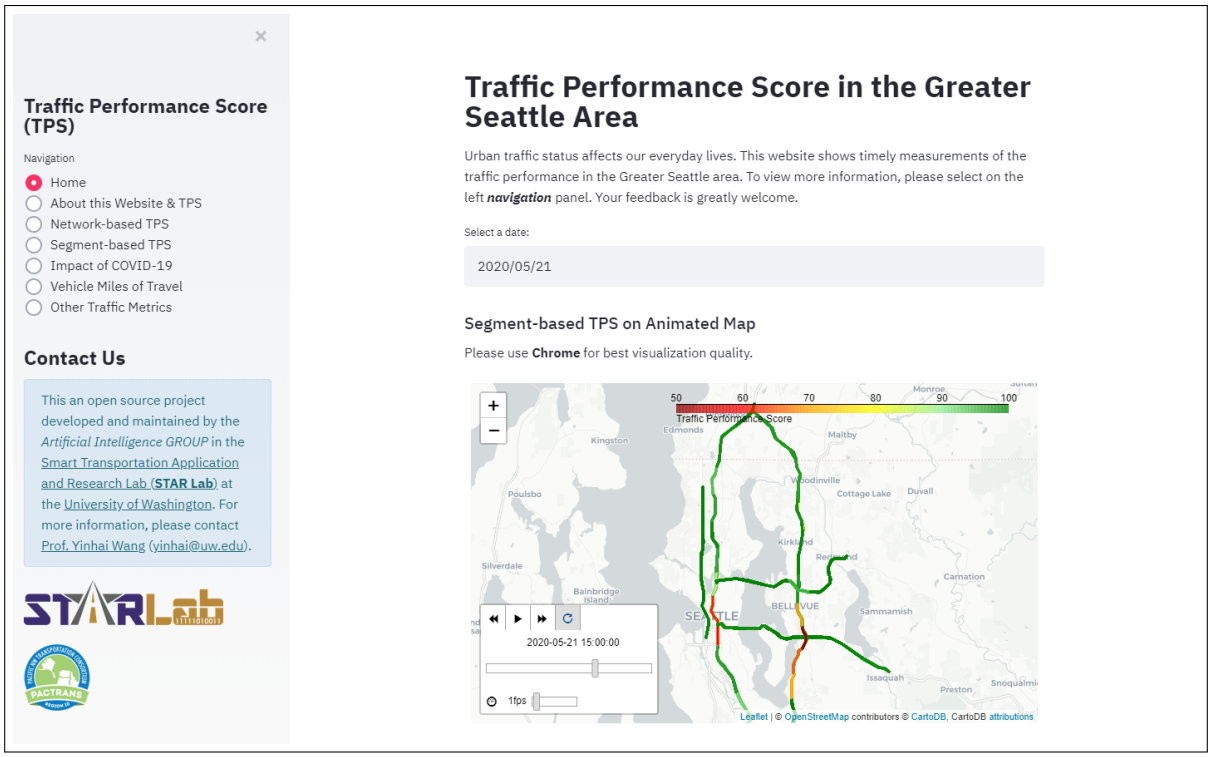


Figure 10.1: The web interface of the traffic performance score platform.

due to COVID-19. For example, historical traffic data indicates that the traffic volume in the Greater Seattle, Washington, area started to decline four weeks before the Washington State's stay-at-home order on March 23. On the week the stay-at-home order was released, traffic volume dropped significantly. Many other aspects of the traffic have also been affected by COVID-19, such as dramatic decreased in public transportation, traffic safety and traffic congestion. Since the pandemic reached the U.S., its influence on transportation has been extensively studied by agencies, researchers, and private companies. Although aspects of the impact have been quantitatively and qualitatively analyzed based on massive data, few studies have been able to measure the impact of COVID-19 on urban mobility from both a road segment level and a traffic network level. One of the targets of this study was to build and release a big data-based traffic performance measurement platform to assist in COVID-19 related analysis.

To observe changes in traffic patterns due to COVID-19, interested parties can utilize public map services, such as Google Map and Apple Map, or review performance metrics from public agencies. However, in most cases, these traffic performance measurements only provide fine-grained (i.e. road segment-level) real-time traffic information or coarse-grained (i.e. network-level)

long-term traffic analysis. During the COVID-19 pandemic, segment-level, long-term analysis and network-wide, real-time traffic performance information are vital to guide personal travel activities and support agency transportation operation management. The network-wide lane-level traffic as well as the traffic performance score are good data sources for enhancing traffic forecasting performance. To this end, this study analyzes both segment-level and network-level traffic performance based on a proposed traffic performance score and other existing traffic metrics, such as vehicle miles of travel (VMT). To measure the impact of COVID-19 on traffic, both short-term and long-term mobility patterns are analyzed.

10.1.1 *Contribution and Organization of the Chapter*

The contribution of this paper can be summarized as follows:

1. A traffic performance score (TPS) for measuring network-wide traffic is proposed. The TPS is also implemented on a public accessible TPS platform [1] for the Greater Seattle area, as shown in Figure 10.1.

2. The network-wide traffic performance under the influence of COVID-19 in the Greater Seattle area is analyzed based on the proposed TPS and other metrics. The variations of TPS and VMT show that responses to the pandemic greatly affect urban traffic.

3. The traffic changes of different road segments in the traffic network are analyzed from both travel demand and driving behaviour perspectives, to investigate how COVID-19 is reshaping the urban mobility.

In the following sections, we first introduce the existing traffic performance metrics and COVID-19 related analysis in Section 10.2. Section 10.3 introduces the proposed TPS and describes the design principles and techniques used to develop the TPS platform. Then, Section 10.4 presents the traffic performance changes in response to COVID-19 from network-scale perspective. Finally, Section 10.5 summarizes this chapter.

1 TPS Platform: http://tps.uwstarlab.org/

10.2.1 *Existing Traffic Performance Metrics*

Traffic performance metrics are based on various essential traffic parameters such as speed, travel time, delay, volume, density, and vehicle miles traveled (VMT). This section summarizes several traditional traffic performance metrics and discusses some recent related studies.

- **Travel Time Index (TTI) (Bharti et al., 2013)**: compares peak (investigated) travel time and free-flow travel time, as defined by:

$$TTI = \frac{\text{Peak Period Travel Time}}{\text{Free-Flow Travel Time}} \tag{10.1}$$

  A large TTI value indicates a state of congested traffic. One of the limitations of TTI is that, while it has a lower bound, it does not have any upper bound.

- **Level of Service (LOS)**: one of the most popular traffic performance metrics. The Highway Capacity Manual (HCM) (Manual, 2016) divides traffic operations into six levels based on traffic density: A (free flow), B (reasonably free flow), C (stable flow), D (approaching unstable flow), E (unstable flow), and F (forced or breakdown flow). While LOS is easy to understand, it also has the following limitations: (1) it is hard to measure traffic density; and (2) LOS is not a continuous metric.

- **Lane Mile Duration Index (LMDI)** (Rao and Rao, 2012): calculated by summing over the product of congested segment length and congestion time, as shown in the following equation:

$$LMDI = \sum_{i=0}^{m} L^i * t^i \tag{10.2}$$

  where L^i is the length of the segment i, t^i is the congestion time of segment i, and m is the total number of segments. One of the limitations of this metric is that it is not scaled and thus can not be used to compare the performances of different traffic networks, due to the fact that the total number of segments may be different in each network.

Recently, He et al. (2016) proposed a framework to evaluate traffic performance of road segments and network. This basic metric is named speed performance index which simply averages speed divided by the free flow speed. To aggregate segment-based metrics to network scale, segment lengths were used as weights for aggregation. Lee and Hong (2014) proposed a traffic performance metric called Traffic Congestion Score (TCS) based on link travel speeds. The TCS is calculated as:

$$TCS_i = \begin{cases} 100\% & v_i \leq 0 \\ 0\% & v_i \geq v_l \\ (1 - \frac{v_i}{v_l}) \times 100\% & otherwise \end{cases} \tag{10.3}$$

where v_i is the average travel speed of link i, v_l is the speed limit.

To summarize, there are several important limitations for the existing traffic performance metrics: (1) they consider only single, simple metrics such as speed or travel time, which cannot fully capture the state of traffic; (2) most of the performance metrics can only investigate individual segments rather than corridors or entire traffic networks; and (3) the above summarized methods have not incorporated VMT to reflect the total travel demands. In this work, to overcome these limitations, a metric to measure network-wide traffic performance is proposed.

10.2.2 *Impact of COVID-19 on Transportation*

Since COVID-19 first appeared, most cities around the world have implemented varying degrees of lock-down policies for their residents, which has had a direct impact on human mobility. Accordingly, many recent studies have sought to investigate the impact of COVID-19 on transportation. Further, many organizations and companies, such as INRIX, TomTom, Google, and Mapbox, have provided significant amounts of data and tools for COVID-19 related research.

Most research that investigates COVID-19's impacts on human mobility is based on traffic volume or VMT. Mescheder (2020) shows how traffic data can help inform policy decisions by visualizing residual mobility along side new cases of COVID-19, based on data from TomTom. This work asserts that traffic data suffices as a measure of the policy because of its time and space attributes. Another report (Marchant, 2020) investigating the effect of the COVID-19 lockdown

on mobility reveals that the top ten cities with the highest traffic reductions worldwide all saw reductions of over 80%. Shi and Fang (2020) examined the time-lagged effect between outbound traffic from Wuhan, China, and the status of the COVID-19 pandemic.

Other COVID-19-related research has focused on traffic safety metrics such as incident numbers and implications of social distancing. C2SMART, University Transportation Center at New York University, released a white paper (Gao et al., 2020) on the impact of COVID-19 on the transportation system, which uses the New York metropolitan area as a case study to analyze transit ridership, subway turnstile entries, traffic on bridges and tunnels, travel time, and the number of crashes during COVID-19. Zhang et al. (2020) developed an interactive COVID-19 Impact Analysis Platform introducing nine metrics related to human mobility and social distancing, and fifteen metrics on COVID-19 and health at the nation-, state-, and county-levels. The social distancing index introduced in this research provides meaningful support for decisions on travel and social distancing policies.

Other recent research explores the impact of COVID-19 on traffic congestion. A congestion index (Clarke, 2020) based on the traffic index from TomTom was introduced to compare congestion during peak hours in 2019 and 2020. After the COVID-19 lockdown policies were announced, the congestion indices of cities around the world declined to varying degrees. Hurley (2020) also proposed a traffic congestion index defined as the percent difference between drive times and the average baseline drive time. Based on the data collected from Google Maps Distance Matrix API, the daily and hourly congestion levels were analyzed. This study found that reduced congestion is also related to the weather. Finally, Maricopa Association of Governments in Arizona, released a report (Governments, 2020) tracking traffic congestion during COVID-19 stay-at-home restrictions based on traffic volumes from INRIX data.

Based the work described above, it appears traffic metrics are affected by the COVID-19 pandemic from multiple perspectives. Based on this assertion other research has investigated changes in travel patterns due to changing traffic scenarios. Grosso et al. (2020) analyzed changes in public transit, bicycling and scootering, air travel, driving, ride-hailing, and delivery vehicles based on stay-at-home policies. Baruchman (2020) compared traffic patterns from January 13 to April 27 in Seattle based on GPS mobile device data collected by Mapbox. This work finds that traffic on major highways and freeways declined, while traffic on neighborhood streets increased.

However, the methods employed were not sufficient to show detailed data and the trend of the change.

Beyond the present, COVID-19 is likely to have long-term effects even as cases and deaths wane. INRIX published a United States National Activity Re-Emergence Map to track the trend of city activities recovering from COVID-19 (INRIX, 2020). Hu et al. (2020) analyzed the travel mode shift effect on traffic caused by COVID-19. This work predicted travel time, post pandemic, across the United States with an interactive Rebound Calculator. It analyzed shifts from transit commuters to single occupancy, from carpool to single occupancy, job loss, and shifts to remote work. The findings indicate that the long term effects of the pandemic on travel may result in increased travel time as communities reopen. Tardivo et al. (2020) published work on COVID-19's impact on railway system. This paper illustrates the effect on global travel manners and the possible rebound of traffic activities after the COVID-19.

In summary, many studies have analyzed the impact of COVID-19 on transportation from various aspects, including impacts on average speed, traffic volume, traffic congestion, traffic patterns, the future effects and so on. However, few researchers have developed indices to intuitively show traffic variation. Those who have developed indices, narrowly focused on one aspect of the impacts but not comprehensive changes to the network. Thus, this study proposes a traffic performance score to measure traffic conditions, and also analyzes how COVID-19 is reshaping urban mobility from road segment and network perspectives.

10.3 TRAFFIC PERFORMANCE SCORE

Performance monitoring is critical for roadway operations, including real-time applications and operational planning, and transportation planning. According to Turner et al. (2004), travel time is the basis for defining mobility-based performance measures. To that end, many performance measures have been designed, such as average travel speed, travel time, travel rate index, and delay per VMT. However, most of the existing performance measures are road segment or trip-based measures. These existing metrics cannot measure performance over a complicated road network. In this section, this paper outlines a traffic performance score to measure performance from the road network perspective.

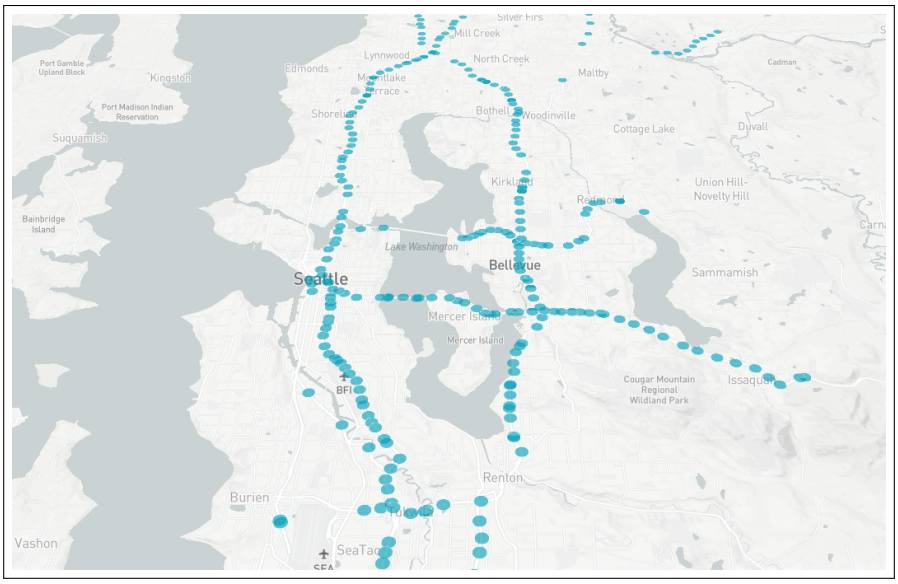


Figure 10.2: The spatial distributions of the loop detector sensors in the Greater Seattle area.

10.3.1 *Data Source*

The TPS is calculated based on data collected from roughly 8000 inductive loop detectors deployed on the freeway network in the northwest region in Washington State. The freeway network mainly includes several major freeways, such as I-5, I-90, I-99, I-167, I-405, and SR-520. The raw data comes from the online Washington State Department of Transportation (WSDOT) Traffic Data Archive Folder [2]. Representative detectors are shown by blue dots in the map in Figure 10.2.

The raw data contains lane-wise speed, volume, and occupancy (density) information collected by each loop detector. Each detector's meta data includes detector category, route, milepost, director, direction, address. Based on the consecutive detectors' location information, freeways can be separated into tiny road segments, each of which contains one loop detector per lane. A road segment's length is then considered as the corresponding detector's covered length. The time interval of the data is one-minute.

Other data sources includes public agencies, public accessible datasets, and official COVID-19 related datasets. In this study, the COVID-19 datasets come from the Wikipedia page of COVID-19 pandemic in Washington [3].

2 Website: http://data.wsdot.wa.gov/traffic/

3 https://en.wikipedia.org/wiki/COVID-19_pandemic_in_Washington_(state)

10.3.2 *Definition*

Normally the physical properties of road segments are invariant, but the traffic stream parameters on each segment, including volume (Q), speed (V), and density (K), keep changing. In order to measure the traffic performance of the traffic network, these traffic parameters at each road segment should be taken into account. Since volume, speed, and density are all related to each other, only two of them need to be incorporated. Thus, volume and speed, which are more general and intuitive parameters are adopted in the design of TPS. The length (L) of the road segment is also taken into consideration by multiplying L with the volume Q, which ($Q \cdot L$) basically represents the vehicle miles of travel (VMT) of the road segment. Then, the TPS is defined as:

$$TPS_t = \frac{\sum_{i=1}^{n} V_t^i \cdot Q_t^i \cdot L^i}{\sum_{i=1}^{n} V_f \cdot Q_t^i \cdot L^i} \times 100\% \tag{10.4}$$

where V_t^i and Q_t^i represent the speed and volume of each road segment i at time t. L^i is the length of i-th detector's covered road segment. V_f is the default free-flow speed.

In this way, the TPS is a value ranging from 0% to 100%. Overall network-wide traffic condition is best when the TPS is 100% and worst when TPS is 0%.

10.3.3 *System Design*

Based on multi-source real-time data, the network-wide and segment-level traffic performance measurement analytical functions and applications are implemented on an interactive publicly accessible web-based traffic performance score platform (http://tps.uwstarlab.org/). This platform integrates real-time data streaming, data source pre-processing, data storage, data modeling, and analytical data visualization using the Streamlit tool [4] which is typically employed to build highly interactive machine learning and data science web applications. The architecture of the platform contains three primary layers is shown in Figure 10.3.

The first layer is the data streaming and pre-processing layer. The real-time inductive loop detector data from WSDOT API is cleaned, archived, and processed in parallel to calculate

4 https://www.streamlit.io/

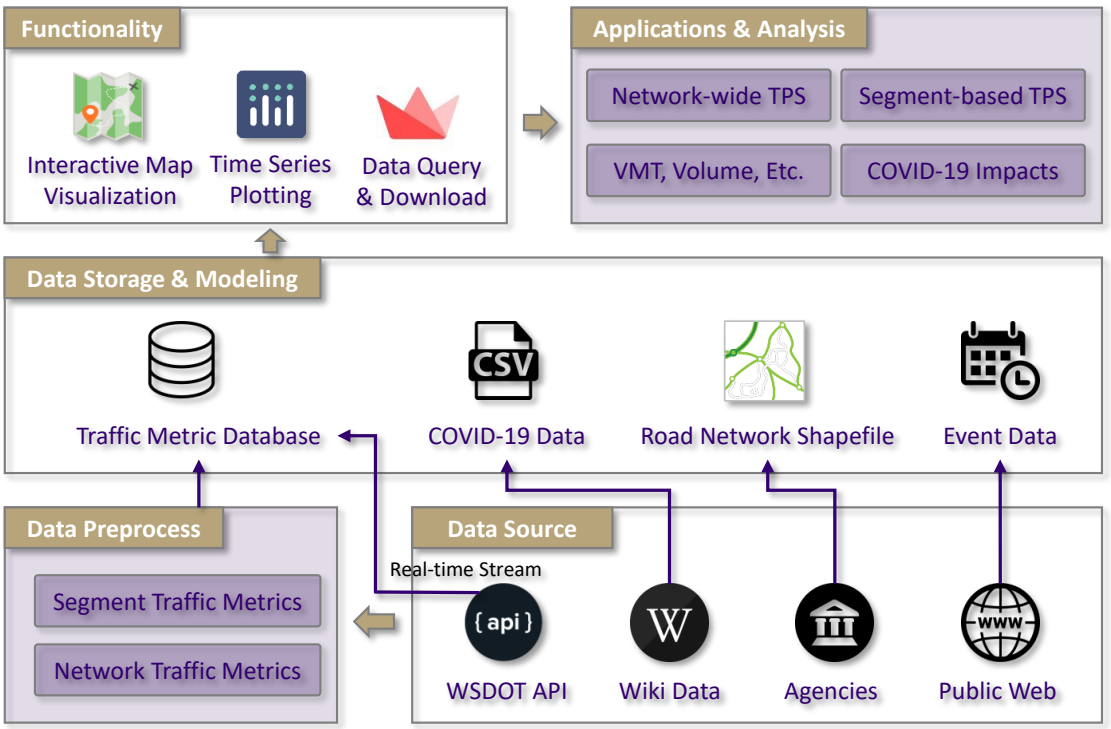


Figure 10.3: Architecture of Traffic Performance Score Platform

network-wide and segment-level traffic performance metrics. The second layer of the platform is mainly responsible for storing traffic raw data and calculated intermediate traffic performance metrics into the database, and building the connections between different datasets based on their spatial-temporal properties. Specifically, each road segment's traffic data is linked to the segment's geospatial properties based on the roadway segment's GIS information, i.e. the WSDOT 24K map [5]. In the third layer, advanced interactive spatial-temporal data visualization tools are employed to demonstrate a variety of network- and segment-level traffic performance measurement analysis results.

10.4 TRAFFIC CHANGES IN RESPONSE TO COVID-19

In this section, the impact of COVID-19 on traffic changes reflected by different traffic performance metrics is discussed. Firstly, government responses to COVID-19 in Washington State are

5 https://www.wsdot.wa.gov/mapsdata/geodatacatalog/maps/NOSCALE/DOT_TDO/LRS/WSDOT_LRS.htm

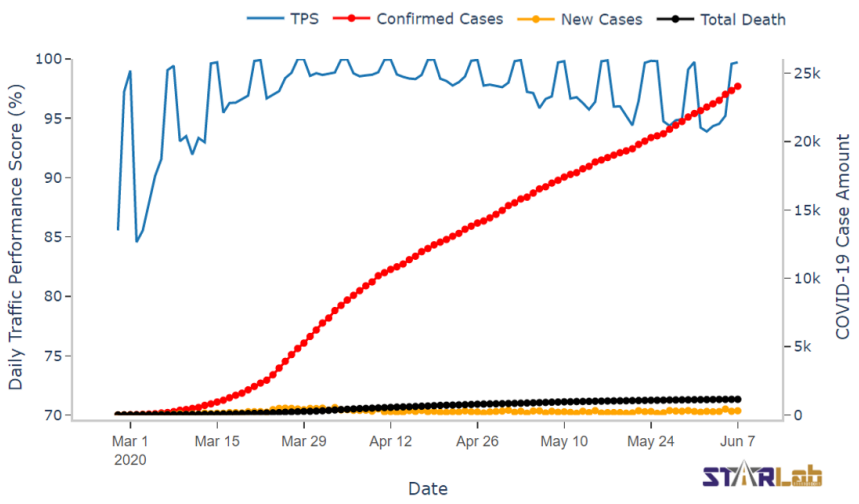


Figure 10.4: COVID-19 cases in Washington State and network-wide traffic performance score in the Greater Seattle area.

investigated, as well as some responses from large employers in and around Seattle, listed as follows:

- March 6: Major tech companies ask Seattle employees to work from home. Amazon and Facebook shut down individual offices as well.
- March 9: University of Washington (UW) suspends on-site classes and finals.
- March 13: Washington state governor announces statewide school closures, expansion of limits on large gatherings.
- March 16: Washington state governor announces statewide shutdown of restaurants, bars and expanded social gathering limits.
- March 23: Washington state governor announces "Stay Home, Stay Healthy" order.
- April 2: Washington state governor extends "Stay Home, Stay Healthy" through May 4.

10.4.1 *Impact of COVID-19 on TPS*

Since the first death from COVID-19 in the U.S. was announced in Kirkland, an east-side suburb of Seattle, on February 29, 2020, the traffic pattern in the Greater Seattle area has gradually

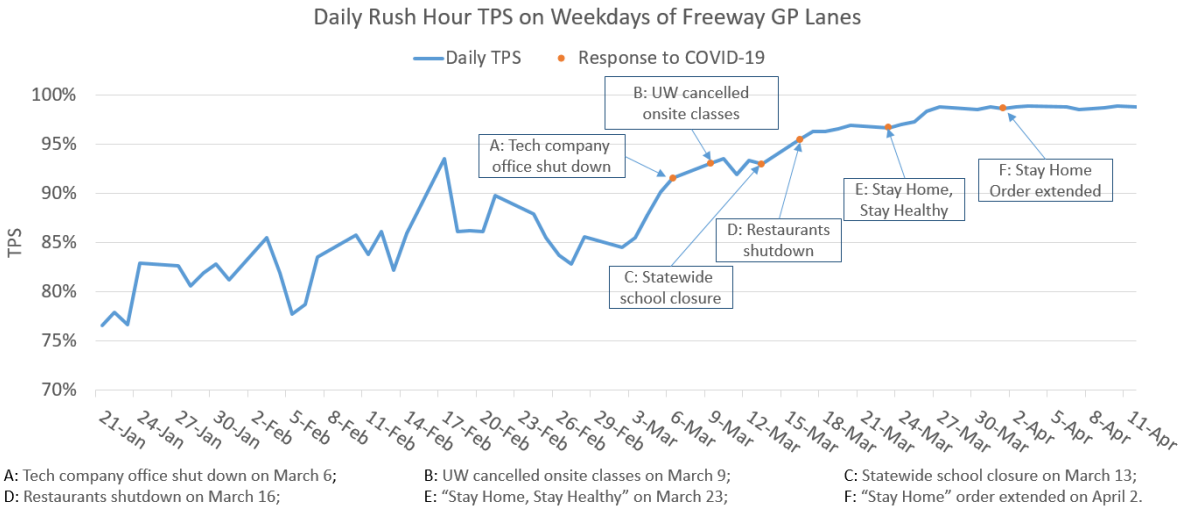


Figure 10.5: Weekday rush hour TPS of freeway general purpose (GP) lanes.

changed along with the public response to COVID-19. COVID-19 cases and the calculated daily network-wide TPS are displayed on the TPS platform, as shown in Figure 10.4. The red, yellow, and black dotted lines shows cumulative confirmed cases, daily new cases, and cumulative deaths, respectively. Although the values of TPS on weekends generally approach 100%, it can be observed that the TPS dramatically increased on weekdays in March. The weekday TPS reached its peak in early April and then gradually decreased, though it still had not recovered to its pre-COVID-19 status as of early June 2020.

In Figure 10.5, the government and major employer responses to the pandemic are illustrated in order to demonstrate their influence on the TPS. Figure 10.5, only shows the weekday rush hour TPS on freeway general purpose (GP) lanes. Based on the preliminary analysis, the average weekday TPS during the rush hour was 83.5% before COVID-19 (from Jan. 21 to Feb. 28). By March 2, the TPS began to increase, implying that residents started to decrease or stagger their traveling activities. After tech companies shut down offices on March 6 and UW moved classes online on March 9, the TPS increased to 92.7%. The orders of statewide school closure and restaurants/bars shutdown led the further increase of the TPS to 95.9%. Finally, once the governor's stay-home order was announced, the TPS leveled out at its highest point around 98%.

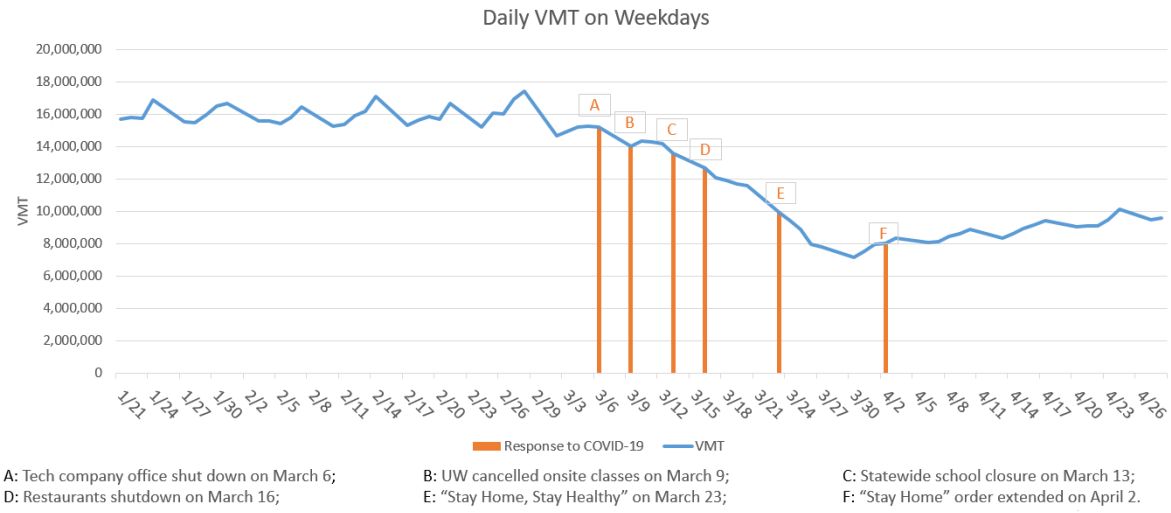


Figure 10.6: Daily VMT changes on weekdays.

10.4.2 *Impact of COVID-19 on VMT*

VMT is another important metric measuring the amount of travel for all vehicles in a geographic region over a given period of time. Given the lane-level loop detector data, the VMT of the freeway system in the Greater Seattle area can be calculated. Figure 10.6 shows the variations of daily VMT on weekdays before and during the presence of the COVID-19 pandemic. Before the outbreak, VMT on the freeway system in the Greater Seattle area was steadily around 16 million per weekday. In correlation with rises in new cases and deaths, and the government and large employer responses, the VMT began to decrease. When Gov. Inslee announced the stay-home order on March 23, the daily VMT dropped by more than half compared to the daily VMT before COVID-19. On March 30, the daily VMT reached the lowest value around 7.1 million.

While the variations of daily VMT are small, weekly VMT clearly reflects the impact of the government and large employer responses to COVID-19. Figure 10.7 shows the percentage of the weekly VMT change with respect to the week prior from early February to late April 2020. It is apparent that weekly VMT remained relatively constant before March, but began to noticeable decline in early March. When the stay-home order was announced in the week starting from March 23, the weekly VMT decreased by its largest single drop, showing a 28.5% decline w.r.t. to that of the previous week. It continued to decline until early April. The weekly VMT started to

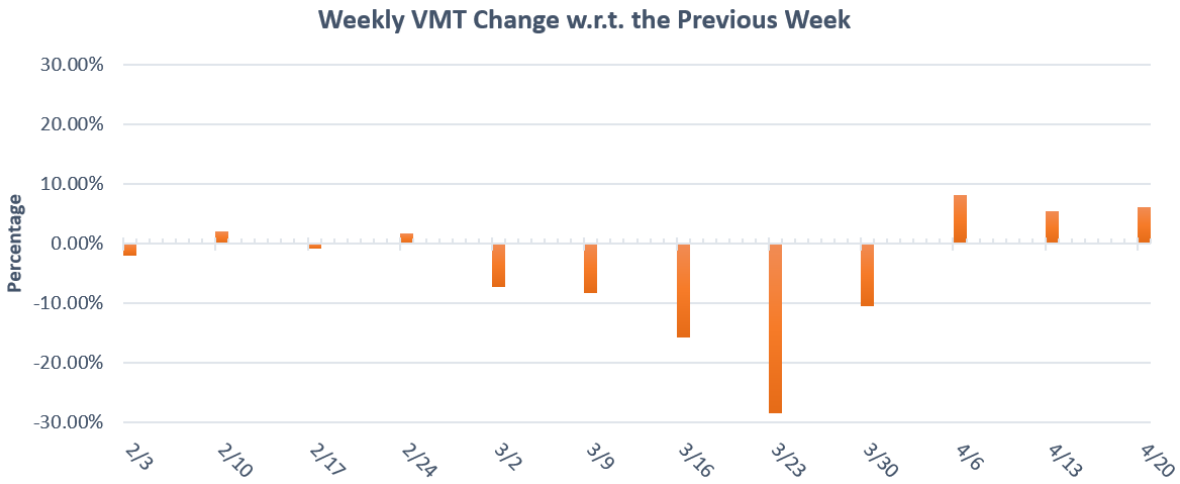


Figure 10.7: Weekly VMT changes with respect to that of the previous week.

increase from the week of April 6, but the growth rate is relatively low compared to the rate of decrease through the month of March, staying around 7%.

The TPS platform also provides a comparison function to contrast the daily VMT with a baseline, i.e. the averaged VMT on each day of the week during a specific period. Figure 10.8 shows the daily VMT by the blue line and VMT changes w.r.t. the baseline from January 19 to February 22 by the green line. The VMT variation in Figure 10.8 shows a similar trend to that in Figure 10.6, where the VMT dropped rapidly at the beginning of COVID-19 and gradually increased after the stay-home order was announced. However, the green line in Figure 10.8 also shows greater detail with regard to weekdays vs. weekends. Namely, weekends showed a much larger drop in the VMT as compared to weekdays, which suggests that weekday traffic during COVID-19 was closer to normal, pre-COVID-19 traffic as compared to weekends. One potential explanation may be that non-essential trips over the weekend reduced significantly during COVID-19, but the necessary trips taken on weekdays for supporting essential social and economic operations, remained more consistent with pre-COVID-19 patterns.

10.4.3 *How is COVID-19 Reshaping Urban Mobility?*

Although TPS and VMT reflect different aspects of traffic performance characteristics, they both measures the network-wide traffic performance. The traffic performance changes reflected by

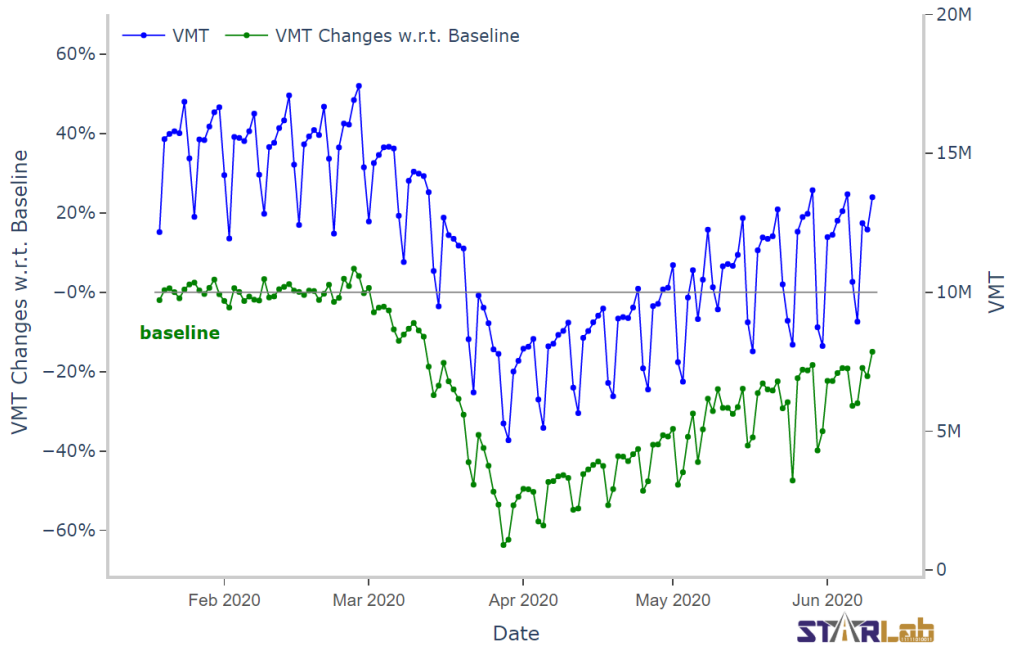


Figure 10.8: VMT changes with respect to baseline. The baseline is the averaged VMT on each day of the week from Jan. 19 to Feb. 22, 2020.

these two metrics show similar trends during the COVID-19 pandemic. However, because road segments in the traffic system distribute vehicles in different spatial regions and have differing functional properties, the impacts on road segments might differ from one another. Thus, to better get at more detailed traffic changes, it is also beneficial to analyze road segment-level traffic performance.

In the Greater Seattle area, four major connected freeways, including I-5, I-90, I-405, and SR-520, form a large network connecting the City of Seattle, the City of Bellevue, the Seattle-Tacoma International Airport, and many other important functional areas. To analyze segment-level traffic performance, this traffic network, consisting of more than 180 miles roadways, is separated into small road segments with the length of two miles. In this way, traffic performance metrics for each segment can be calculated. In this section, the traffic changes of different road segments are analyzed to investigate how COVID-19 is reshaping urban mobility.

Segment	01-19 to 02-29	03-01 to 03-07	03-08 to 03-14	03-15 to 03-21	03-22 to 03-28	03-29 to 04-04	04-05 to 04-11	04-12 to 04-18	04-19 to 04-25	04-26 to 05-02	05-03 to 05-09	05-10 to 05-16
5, SB, 179 to 181	1190	1174	1125	947	717	652	684	726	766	813	908	938
5, SB, 163 to 165	1176	1166	1072	893	688	622	651	683	722	765	846	861
405, NB, 8 to 10	1169	1157	1082	927	699	614	648	682	725	772	867	894
405, NB, 6 to 8	1153	1147	1091	942	711	626	661	697	744	795	888	921
405, SB, 6 to 8	1150	1139	1079	906	695	613	644	683	721	772	857	870
5, SB, 165 to 167	1124	1126	1054	893	674	609	627	667	702	744	830	831
405, SB, 8 to 10	1095	1093	1035	876	673	593	624	658	693	744	831	842
5, SB, 175 to 177	1078	1066	1013	851	648	588	617	654	688	728	812	836
5, SB, 161 to 163	1060	1044	957	796	605	545	542	537	566	603	664	680
405, SB, 0 to 2	1055	1041	963	793	600	545	566	597	630	666	736	744
405, NB, 28 to 30	1047	1011	966	826	628	566	574	630	673	708	798	818
405, NB, 18 to 20	1047	1009	944	793	594	521	559	591	623	666	764	785
5, NB, 167 to 169	1046	1021	918	746	549	492	494	536	565	601	679	693
405, NB, 0 to 2	1041	1053	983	827	640	578	577	627	658	692	760	773
5, SB, 169 to 171	1033	1058	996	819	610	551	582	612	649	689	776	793
405, SB, 24 to 26	1029	999	982	853	645	575	618	655	694	741	838	862
5, NB, 161 to 163	1028	1037	1004	857	683	628	605	704	745	783	855	895
5, SB, 167 to 169	1000	1000	928	758	562	508	518	552	584	622	702	709
5, SB, 171 to 173	998	1007	936	764	561	504	520	564	601	678	766	788
5, SB, 173 to 175	995	988	924	767	577	519	526	568	601	636	714	735
90, WB, 11 to 13	988	978	916	765	533	465	469	512	542	581	695	716
90, WB, 13 to 15	983	956	885	736	513	448	439	493	520	554	659	676
5, SB, 157 to 159	981	964	879	723	533	472	473	504	529	559	622	635
5, NB, 165 to 167	980	966	915	776	571	506	514	558	591	630	702	772
5, SB, 153 to 183	976	969	900	748	563	506	519	551	582	617	694	706
405, SB, 26 to 28	972	940	909	781	585	521	558	591	629	669	757	781
5, NB, 169 to 171	970	958	869	711	526	471	460	517	545	581	659	674
5, NB, 153 to 155	969	951	894	751	575	511	526	556	591	625	689	721
5, NB, 179 to 181	963	924	867	731	547	494	516	550	581	618	693	709
5, SB, 159 to 161	949	934	850	701	516	455	466	487	512	541	604	619
5, SB, 177 to 179	939	914	859	721	552	497	522	560	589	624	698	719
5, NB, 157 to 159	928	903	856	722	536	478	488	517	543	575	614	659
405, SB, 0 to 30	921	893	844	711	534	470	499	529	559	599	680	697
405, NB, 12 to 14	920	882	804	663	480	414	440	466	496	532	615	635
5, NB, 153 to 183	916	897	839	702	527	472	478	515	544	575	641	664
405, SB, 18 to 20	913	873	826	695	521	456	489	520	547	587	673	690
405, NB, 26 to 28	911	883	834	708	529	470	504	528	565	597	678	696
90, EB, 13 to 15	902	851	751	611	419	362	375	404	425	458	554	561
5, NB, 171 to 173	901	884	806	663	493	443	440	486	511	534	606	621

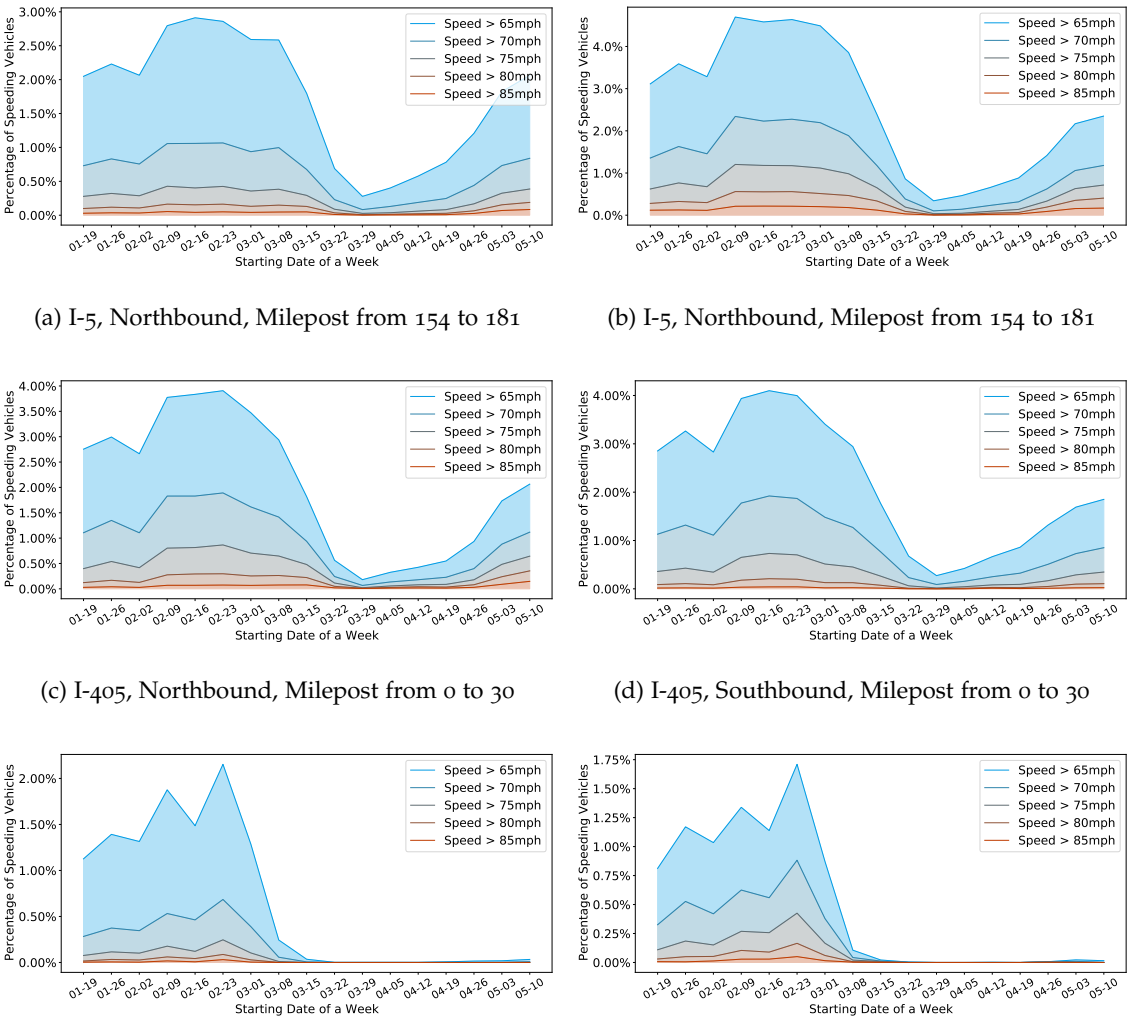
Figure 10.9: Road segment rank based on volume per lane per hour (VpLpH) of the freeway network in the Greater Seattle area. The header shows the time periods, in which the first period (01-19 to 02-29) is considered as the baseline. The values in table cells indicate a segments' VpLpH during each respective period.

10.4.3.1 *Reshaping Travel Demand*

As articulated in the previous section, the traffic volume over the entire freeway network decreased. However, because each road segment serves different functions, the volume of some segments during COVID-19 reduced more dramatically, while others did not. To quantify the relative volume changes of road segments during COVID-19, the volume per lane per hour (VpLpH) of road segments is caluclated during a specific period and then ranked. Figure 10.9 shows 40 segments, using different colors to visual their respective rank during the time period from January to May 2020. The value in each cell represents the VpLpH of the corresponding segment during a specific period, and the headers show the time periods, where the first period, or column, (from Jan. 19 to Feb. 29) is considered the baseline. It can be observed that the ranking is in order from red to green in the first column (the baseline) but then does not stay constant throughout each time period. This shows that although the VpLpH of all the segments decreased, the ranks did not remain constant. The ranks of several segments noticeably increased while the ranks of several others decreased, which is why columns other than the baseline column do not maintain the clear gradient from red to green. The relative increase in VpLpH suggests an increased importance of those road segments marked with red color. Conversely, those road segments (rows) that move to a darker green suggest those segments are less important during the COVID-19 pandemic. The change of ranks, or say the relative importance of road segments, results due to the change of travel demand under the influence of COVID-19. The stay-home order caused many residents to reduce daily travel. But the trips made by the people who maintain essential services likely did not see the same drop. As Figure 10.9 shows, the decreased travel demand is not evenly distributed across the traffic network. This presents an opportunity to further analyze the functional role of each road segment. Interestingly, the VpLpH changes of road segments at I-5 from milepost 161 to 163 in each direction, which located between the City of Seattle and the Sea-Tac Airport, flip; where the southbound segment gets greener over time, suggesting it is of less importance during COVID-19, while the northbound segment gets redder, suggesting it is of more importance. The rank of the northbound segment increased from around 20 to top 2, but the rank of the southbound segment gradually decreased from around 10 to around 30. These significant rank changes reflect the shift

in the respective roles of each road segments on the network during COVID-19 as compared to normal traffic scenarios.

10.4.3.2 *Reshaping Driving Behaviour*



(a) I-5, Northbound, Milepost from 154 to 181

(b) I-5, Northbound, Milepost from 154 to 181

(c) I-405, Northbound, Milepost from 0 to 30

(d) I-405, Southbound, Milepost from 0 to 30

(e) SR-520, Eastbound, Milepost from 0 to 1

(f) SR-520, Westbound, Milepost from 0 to 1

Figure 10.10: Percentages of speeding vehicles on major freeways in the Greater Seattle area before - and during - COVID-19.

Because travel demand during COVID-19 has changed, the traffic scenarios for drivers has also changed. Thus, an individual's driving behavior might also be influenced by COVID-19. News (Kaji et al., 2020; SULLIVAN, 2020) reported a rise in speeding in many States in the U.S.

These conclusions in the news are traditionally drawn from officers' observations or based on the statistics of fatal crashes. In this section, we analyze and compare the speeding rates before, and during, COVID-19.

Here, the percentage of speeding vehicles on a specific corridor or the road network based on the lane-level loop detector data is calculated with one-minute time interval using the following equation

$$SpeedingRate = \frac{\sum_{V_i > V_{TH}} Q_i}{\sum Q_i} \tag{10.5}$$

where V_i and Q_i are the detected speed and volume of a loop detector and V_{TH} is the speeding threshold. Figure 10.10 shows the percentages of speeding vehicles on three major freeways, I-5, I-405, and SR-520, in the Greater Seattle area during each week before, and during, COVID-19 (from Jan. 19 to May. 10). Each sub-figure plots the time-varying percentage curves with different speeding thresholds ranging from 65 miles per hour (mph) to 85 mph. The percentages of speeding vehicles on I-5 and I-405 on both directions, as shown by Figure 10.10 (a) to (d), have similar trends. The speeding rates reached a peak before COVID-19 around mid-February. In March, the speeding rates gradually reduced and reached their lowest points at the end of March. Since then, the speeding rates have started to increase. Ultimately, Figure 10.10 clearly shows that speeding rates during COVID-19 were significantly less than those before COVID-19. The reduction in speeding rates on SR-520, as shown in Figure 10.10 (e) and (f), are even dramatic than those for I-5 and I-405 after the stay-home order, as they suddenly reduced close to zero and did not rebound like the percentage curves on I-5 and I-405.

It is important to note here, that when using loop detector data, the speed values of different vehicles within the one-minute time interval observed by a loop detector are averaged. Thus it is possible that one vehicle passing a detector may be driving at excessive speeds, but the observed averaged speed of the detector within one minute does not capture this outlier. While a small number of drivers may have driven at much higher speeds, the analysis results indicate that the overall speeding rate of freeways in the Greater Seattle area during COVID-19 is noticeably less than that before COVID-19.

10.5 CHAPTER SUMMARY

In this chapter, the traffic performance score to measure network-wide traffic performance is proposed. Based on real-time loop detector data and other data sources, a traffic performance score platform to display both network-level and segment level freeway traffic performance in the Greater Seattle area is developed. Based on real-time traffic data and the calculated traffic metrics, the impact of COVID-19 on traffic before and during COVID-19 are analyzed. The results show that both TPS and VMT have similar patterns. Namely, that the urban traffic has been greatly influenced by COVID-19, especially after the governors stay-home order was announced. The road segment-level travel demand changes and driver behavior changes are also measured, which reflects the ways COVID-19 is reshaping urban mobility.

The traffic patterns, travel demands, and driving behaviors in different segments of the region have greatly changed following the outbreak of the pandemic. These impacts could be considered temporary. However, many large, regional employers, such as technology companies, have begun planning to allow employees to work from home permanently. If COVID-19 ultimately results in a work-from-home revolution as many are hypothesizing, urban mobility will likely not return to pre-COVID-19 patterns. Thus, the newly developed methods described in this work will be vital for quantifying changing traffic patterns into the foreseeable future. Because the TPS platform is built for scalability, it can easily be extended to cover more cities and regions across the country and the globe, and can facilitate more comprehensive analysis functions. It will be an important tool for agencies as they consider new policy directions moving forward.

Part IV

FINAL REMARKS

11

SUMMARY AND FUTURE DIRECTIONS

This dissertation has introduced a series of deep learning-based models for solving the short-term network-wide traffic forecasting problems under two scenarios, i.e. traffic forecasting with and without missing values. The introduced traffic forecasting research can be categorized into two types, (●) **learning traffic as a matrix** using RNN-based networks and (●) **learning traffic as a graph** using graph-based neural networks. Furthermore, it has also introduced a set of methodologies, applications, and platforms to process, integrate, and formulate transportation data for enhancing prediction performance, testing existing traffic forecasting models, and sharing experimental results as benchmarks to facilitate future research.

11.1 RESEARCH FINDINGS AND CONTRIBUTIONS

The main findings and research contributions can be categorized based on three parts of the dissertation. Specifically, the following work has been completed, documented, and discussed:

- **Part i: Prediction**
  - To deal with spatiotemporal traffic state data and fulfill short-term network-wide traffic forecasting tasks, Chapter 3 (●) attempted to stack bidirectional RNN with unidirectional RNN to capture the bidirectional dependencies in time series data. It is found that stacking RNNs can improve the traffic prediction performance. However, the interpretability of RNN-based models should be improved.

- To better characterize spatial properties of traffic data and extract localized topological features from the road network, We learn the traffic network as a graph. In order to incorporate roadway physical properties into the deep learning model, Chapter 4 (●) presents the traffic graph convolutional operation, which incorporates traffic speed and roadway length information, and combine it into the LSTM structure. It is shown that the feature selection is more comprehensive and interpretable.
- To dynamically capture the localized features in road network-based graph, Chapter 5 (●) moves one step further to incorporate graph wavelet operation into the recurrent neural network structure. The sparsity of the graph wavelet weight matrices can help identify the most influential vertices in the road network-based graph and thus can relatively interpret the model.

- **Part ii: Prediction with Missing Values**
  - To deal with missing values in traffic prediction, Chapter 6 (●) presents a new neural network structure, LSTM with an imputation unit, to infer the missing values during the prediction process.
  - To solve missing values in the graph structure, we defined the traffic state transition process as a graph Markov process. Based on the graph Markov process, Chapter 7 (●) proposes a graph Markov network incorporating the spatial dependencies between neighboring links and those links' temporal relationship between different time steps. This new neural network structure is proved to fulfill imputing missing values and predicting future traffic states at the same time.

- **Part iii: Integration and Platforms**
  - To overcome the challenges brought on by immense transportation data, Chapter 8 describes a multi-source traffic data integration framework whose generated integrated data can be a source of traffic prediction tasks.
  - To facilitate future deep learning based network-wide traffic prediction research, Chapter 9 (● ●) introduces an open source platform, TraffiX.ai, which shares datasets, publishes source code, and posts evaluation results of existing deep learning-based traffic

prediction models. Those results as benchmarks can benefit further deep learning-based traffic prediction research.

– As a representative traffic prediction application, a network-wide traffic data based performance measurement platform, TPS, is presented by Chapter 10. The TPS platform has the capability of analyzing traffic performance under special conditions and applying the proposed traffic prediction models into real applications.

11.2 CHALLENGES AND OPPORTUNITIES

Future extensions of this work include carrying out more in-depth studies on efficiently learning features from network-wide traffic network. The impact of missing values in historical traffic state data will be further evaluated. To further facilitate traffic forecasting related research, the datasets and models supported by the transportation AI platform will be expanded, and traffic forecasting performance of more novel models will be tested and updated on the platform.

However, according to the research work and findings in this dissertation, we may expect new challenges in the transportation data mining and traffic prediction research fields, especially considering AI is continuously and acceleratively introduced into modern smart city development. The challenges can be described from at lease three perspectives:

- **Data**: The challenges include but not limit to (1) do we have enough data for traffic prediction task? (2) Are the datasets correctly selected? (3) What is the role of data integration and fusion in traffic prediction? (4) Can we reach a consensus about what are the standard datasets for traffic forecasting problems?

- **Model**: Developing and evaluating traffic forecasting models are also challenging in terms of (1) designing effective feature selection modules, (2) properly tuning parameters in complicated neural networks, and (3) developing customized physics-informed neural networks to meet the intrinsic law of transportation systems.

- **Application**: We may need to answer several questions that are not technical but related to the broad traffic prediction area as well: (1) is the model design the most critical component for a successful traffic prediction application? (2) How to properly transfer the learned

knowledge to apply existing model to different heterogeneous cities' scenarios? (3) When applying deep learning-based traffic forecasting models into real practice, how to combine the strengths of agencies, private companies, and researchers?

In light of these challenges, we see great opportunities to leverage modern advances in artificial intelligence to conduct promising research to fill the observed and potential gaps. As such, we describe several research directions. Firstly, as we are entering the connected and autonomous vehicles era, we can expect the types and volume of data generated from urban area, not only related to transportation, will dramatically increase. Establishing a thorough spatial-temporal data processing and fusion framework will be beneficial for urban data mining research. As for traffic state estimation and prediction research, including estimating and predicting traffic volume, speed, travel time, origin-destination, etc., developing customized advanced deep learning models by combining urban/roadway properties will be promising. We also need to keep it in mind that traffic state estimation and prediction is only a component of intelligent transportation systems. Since the future state of traffic will affect the current drivers' behaviors, combining traffic control with traffic estimation and prediction and further estimating their interactive impacts will also be promising research directions. Lastly, traffic prediction has been widely studied over the past several years. How to apply the proposed model into real practice to benefit the public is still deserved to be explored. A good example is the collaboration between Deep Mind and Google for improving the accuracy of estimated times of arrival (ETAs) by using graph neural networks (Lange and Perez, 2020). It will be promising for researchers to leverage the computation power and datasets from private companies and agencies to solve real-world traffic problems.

BIBLIOGRAPHY

Abdulhai, Baher, Rob Pringle, and Grigoris J Karakoulas (2003). "Reinforcement learning for true adaptive traffic signal control." In: *Journal of Transportation Engineering* 129.3, pp. 278–285 (cit. on pp. 10, 186).

Abu-El-Haija, Sami, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan (2019). "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing." In: *36th International Conference on Machine Learning, ICML 2019* 2019-June, pp. 32–41 (cit. on pp. 42, 49).

Alahi, Alexandre, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese (2016). "Social lstm: Human trajectory prediction in crowded spaces." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–971 (cit. on p. 18).

Amershi, Saleema, Andrew Begel, Christian Bird, Robert Deline, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann (2019). "Software Engineering for Machine Learning: A Case Study." In: *41st ACM/IEEE International Conference on Software Engineering (ICSE 2019)*. URL: https://docs.microsoft.com/en-us/azure/devops/learn/devops-at-microsoft/ (cit. on pp. 14, 187).

Arel, Itamar, Cong Liu, T Urbanik, and A G Kohls (2010). "Reinforcement learning-based multi-agent system for network traffic signal control." In: *IET Intelligent Transport Systems* 4.2, pp. 128–135 (cit. on pp. 10, 186).

Asif, Muhammad Tayyab, Justin Dauwels, Chong Yang Goh, Ali Oran, Esmail Fathi, Muye Xu, Menoth Mohan Dhanya, Nikola Mitrovic, and Patrick Jaillet (2013). "Spatiotemporal patterns in large-scale traffic speed prediction." In: *IEEE Transactions on Intelligent Transportation Systems* 15.2, pp. 794–804 (cit. on p. 96).

Atwood, James and Don Towsley (2016). "Diffusion-convolutional neural networks." In: *Advances in Neural Information Processing Systems*, pp. 1993–2001 (cit. on pp. 13, 41, 45).

Barredo-Arrieta, Alejandro, Ibai Laña, and Javier Del Ser (2019). "What Lies Beneath: A Note on the Explainability of Black-box Machine Learning Models for Road Traffic Forecasting." In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, pp. 2232–2237 (cit. on pp. 14, 128).

Baruchman, Michelle (2020). *See the traffic patterns before and during Washington's coronavirus stay-home order | The Seattle Times*. URL: https://www.seattletimes.com/seattle-news/transportation/see-the-traffic-patterns-before-and-during-washingtons-coronavirus-stay-home-order/ (cit. on p. 204).

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult." In: *IEEE transactions on neural networks* 5.2, pp. 157–166 (cit. on p. 23).

Bharti, Anish Kumar, Satish Chandra, and Ravi Sekhar Chalumuri (2013). "Performance evaluation of urban arterial in Delhi using travel time reliability." In: *Proceedings of the eastern Asia society for transportation studies*. Vol. 9 (cit. on p. 202).

Boquet, Guillem, Jose Lopez Vicario, Antoni Morell, and Javier Serrano (2019). "Missing Data in Traffic Estimation: A Variational Autoencoder Imputation Method." In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 2882–2886 (cit. on p. 139).

Box, George E P, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung (2015). *Time series analysis: forecasting and control*. John Wiley & Sons (cit. on pp. 20, 98).

Bracewell, Ronald Newbold and Ronald N Bracewell (1986). *The Fourier transform and its applications*. Vol. 31999. McGraw-Hill New York (cit. on p. 73).

Bruna, Joan, Wojciech Zaremba, Arthur Szlam, and Yann LeCun (2013). "Spectral networks and locally connected networks on graphs." In: *arXiv preprint arXiv:1312.6203* (cit. on pp. 13, 75).

Cai, Pinlong, Yunpeng Wang, Guangquan Lu, Peng Chen, Chuan Ding, and Jianping Sun (2016). "A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting." In: *Transportation Research Part C: Emerging Technologies* 62, pp. 21–34 (cit. on p. 96).

Chandra, Srinivasa Ravi and Haitham Al-Deek (2009). "Predictions of freeway traffic speeds and volumes using vector autoregressive models." In: *Journal of Intelligent Transportation Systems* 13.2, pp. 53–72 (cit. on pp. 20, 96, 98).

Chang, H, Y Lee, B Yoon, and S Baek (2012). "Dynamic near-term traffic flow prediction: system-oriented approach based on past experiences." In: *Iet Intelligent Transport Systems* 6.3, pp. 292–305 (cit. on pp. 11, 71).

Che, Zhengping, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu (2018). "Recurrent neural networks for multivariate time series with missing values." In: *Scientific reports* 8.1, p. 6085 (cit. on pp. 14, 97, 101, 104, 111, 117, 128, 140, 141).

Chen, Cen, Kenli Li, Sin G. Teo, Guizi Chen, Xiaofeng Zou, Xulei Yang, Ramaseshan C. Vijay, Jiashi Feng, and Zeng Zeng (2018a). "Exploiting Spatio-Temporal Correlations with Multiple 3D Convolutional Neural Networks for Citywide Vehicle Flow Prediction." In: *Proceedings - IEEE International Conference on Data Mining, ICDM*. Vol. 2018-Novem. IEEE, pp. 893–898. ISBN: 9781538691588. DOI: 10.1109/ICDM.2018.00107 (cit. on p. 44).

Chen, Chao, Jaimyoung Kwon, John Rice, Alexander Skabardonis, and Pravin Varaiya (2003). "Detecting errors and imputing missing data for single-loop surveillance systems." In: *Transportation Research Record: Journal of the Transportation Research Board* 1855, pp. 160–167 (cit. on p. 101).

Chen, Chao, Jaimyoung Kwon, and Pravin Varaiya (2002). "The quality of loop data and the health of California's freeway loop detectors." In: *PeMS Development Group*, pp. 5–6 (cit. on p. 139).

Chen, Ching-Chien, Craig A Knoblock, and Cyrus Shahabi (2006). "Automatically conflating road vector data with orthoimagery." In: *GeoInformatica* 10.4, pp. 495–530 (cit. on pp. 162, 170).

Chen, Xinyu, Zhaocheng He, Yixian Chen, Yuhuan Lu, and Jiawei Wang (2019a). "Missing traffic data imputation and pattern discovery with a Bayesian augmented tensor factorization model." In: *Transportation Research Part C: Emerging Technologies* 104, pp. 66–77 (cit. on p. 112).

Chen, Xinyu, Zhaocheng He, and Lijun Sun (Jan. 2019b). "A Bayesian tensor decomposition approach for spatiotemporal traffic data imputation." In: *Transportation Research Part C: Emerging Technologies* 98, pp. 73–84. ISSN: 0968090X. DOI: 10.1016/j.trc.2018.11.003. URL: https://www.sciencedirect.com/science/article/pii/S0968090X1830799X (cit. on pp. 11, 13, 71, 101, 111, 117, 125, 127, 128, 130).

Chen, Xinyu, Zhaocheng He, and Jiawei Wang (2018b). "Spatial-temporal traffic speed patterns discovery and incomplete data recovery via SVD-combined tensor decomposition." In: *Transportation research part C: emerging technologies* 86, pp. 59–77 (cit. on pp. 12, 127, 128).

Chen, Yuan-yuan, Yisheng Lv, Zhenjiang Li, and Fei-Yue Wang (2016). "Long short-term memory model for traffic congestion prediction with online open data." In: *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, pp. 132–137 (cit. on pp. 18, 19, 31, 33, 97–100, 102, 107).

Chien, Steven I J, Xiaobo Liu, and Kaan Ozbay (2003). "Predicting travel times for the South Jersey real-time motorist information system." In: *Transportation Research Record* 1855.1, pp. 32–40 (cit. on p. 96).

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 1724–1734. DOI: 10.3115/v1/d14-1179 (cit. on pp. 3, 12, 23, 41, 44, 69, 97, 140).

Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling." In: *arXiv preprint arXiv:1412.3555* (cit. on p. 23).

Claramunt, C, B Jiang, and A Bargiela (2000). "A new framework for the integration, analysis and visualisation of urban traffic data within geographic information systems." In: *Transportation Research Part C: Emerging Technologies* 8.1-6, pp. 167–184 (cit. on p. 157).

Clarke, Seán (2020). *The traffic data that shows the road into - and out of - Covid-19 lockdown* (cit. on p. 204).

Cremer, M and St Schrieber (1996). "Monitoring traffic load profiles with heterogeneous data source configurations." In: *TRANSPORTATION AND TRAFFIC THEORY. PROCEEDINGS OF THE 13TH INTERNATIONAL SYMPOSIUM ON TRANSPORTATION AND TRAFFIC THEORY, LYON, FRANCE, 24-26 JULY 1996* (cit. on p. 157).

Cui, Zhiyong, Kristian Henrickson, Ruimin Ke, and Yinhai Wang (2019). "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting." In: *IEEE Transactions on Intelligent Transportation Systems* (cit. on pp. 3, 6, 68, 72, 74, 82, 100, 127, 139).

Cui, Zhiyong, Ruimin Ke, and Yinhai Wang (2017). "Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction." In: *6th Interna-*

tional Workshop on Urban Computing (UrbComp 2017). URL: http://arxiv.org/abs/1801.02143 (cit. on pp. 6, 12, 13, 41, 44, 46, 71, 79, 100, 127).

Cui, Zhiyong, Shen Zhang, Kristian C K.C. Henrickson, and Yinhai Wang (2016). "New progress of DRIVE Net: An E-science transportation platform for data sharing, visualization, modeling, and analysis." In: *Smart Cities Conference (ISC2), 2016 IEEE International*. IEEE, pp. 1–2. ISBN: 97815090184 51. DOI: 10.1109/ISC2.2016.07580736 (cit. on pp. 29, 100, 110, 156).

Daganzo, Carlos F. (1994). "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory." In: *Transportation Research Part B* 28.4, pp. 269–287. ISSN: 01912615. DOI: 10.1016/0191-2615(94)90002-7 (cit. on p. 47).

Daneshgar, Farzad, Kaveh Farokhi Sadabadi, and Ali Haghani (2018). *A Conflation Methodology for Two GIS Roadway Networks and Its Application in Performance Measurements*. Tech. rep. (cit. on p. 158).

David, Welna (2020). *'We All Feel At Risk': 100,000 People Dead From COVID-19 In The U.S.* (Cit. on p. 199).

De Boor, Carl, Carl De Boor, Etats-Unis Mathématicien, Carl De Boor, and Carl De Boor (1978). *A practical guide to splines*. Vol. 27. Springer-Verlag New York (cit. on p. 101).

Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2016). "Convolutional neural networks on graphs with fast localized spectral filtering." In: *Advances in Neural Information Processing Systems*. Advances in Neural Information Processing Systems, pp. 3844–3852. ISBN: 978-1-5108-3881-9. URL: http://arxiv.org/abs/1606.09375 (cit. on pp. 13, 41, 42, 45, 51, 58, 68, 74, 81, 129, 132).

Du, Mengnan, Ninghao Liu, Fan Yang, Shuiwang Ji, and Xia Hu (2019). "On Attribution of Recurrent Neural Network Predictions via Additive Decomposition." In: *The World Wide Web Conference*. ACM, pp. 383–393 (cit. on pp. 14, 128).

Duan, Yanjie, Yisheng Lv, Yu-Liang Liu, and Fei-Yue Wang (2016a). "An efficient realization of deep learning for traffic data imputation." In: *Transportation research part C: emerging technologies* 72, pp. 168–181 (cit. on pp. 13, 44, 125, 128).

Duan, Yanjie, Yisheng Lv, and Fei Yue Wang (2016b). "Travel time prediction with LSTM neural network." In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. IEEE,

pp. 1053–1058. ISBN: 9781509018895. DOI: 10.1109/ITSC.2016.7795686 (cit. on pp. 12, 19, 97, 98, 100, 102, 107).

Eck, Douglas and Juergen Schmidhuber (2002). "A first look at music composition using lstm recurrent neural networks." In: *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale* 103 (cit. on p. 18).

El Faouzi, Nour-Eddin, Henry Leung, and Ajeesh Kurian (2011). "Data fusion in intelligent transportation systems: Progress and challenges–A survey." In: *Information Fusion* 12.1, pp. 4–10 (cit. on pp. 4, 5, 154, 156).

Estrach, Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun (2014). "Spectral networks and deep locally connected networks on graphs." In: *2nd International Conference on Learning Representations, ICLR*. Vol. 2014 (cit. on pp. 41, 45, 51, 68, 74).

FHWA (2019). *National Performance Management Research Data Set (NPMRDS)*. URL: https://ops.fhwa.dot.gov/perf_measurement/index.htm (cit. on pp. 80, 194).

Fourati, Hassen (2015). *Multisensor Data Fusion: From Algorithms and Architectural Design to Applications (Book)* (cit. on p. 157).

Fu, Rui, Zuo Zhang, and Li Li (2016). "Using LSTM and GRU neural network methods for traffic flow prediction." In: *Chinese Association of Automation (YAC), Youth Academic Annual Conference of*. IEEE, pp. 324–328 (cit. on p. 19).

Gal, Yarin and Zoubin Ghahramani (2016). "A theoretically grounded application of dropout in recurrent neural networks." In: *Advances in neural information processing systems*, pp. 1019–1027 (cit. on p. 101).

Gao, Jingqin, Suzana Duran Bernardes, Zilin Bian, Kaan Ozbay, and Shri Iyer (2020). "Initial Impacts of COVID-19 on Transportation Systems: A Case Study of the US Epicenter, the New York Metropolitan Area." In: *arXiv preprint arXiv:2010.01168* (cit. on p. 204).

Garcia-Laencina, Pedro J, José-Luis Sancho-Gómez, and Anibal R Figueiras-Vidal (2010). "Pattern classification with missing data: a review." In: *Neural Computing and Applications* 19.2, pp. 263–282 (cit. on p. 101).

Geng, Xu, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu (2019). "Spatiotemporal Multi-Graph Convolution Network for Ride-Hailing Demand Forecasting."

In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 3656–3663. DOI: 10.1609/aaai.v33i01.33013656 (cit. on pp. 13, 45).

Gers, Felix A, Jürgen Schmidhuber, and Fred Cummins (1999). "Learning to forget: Continual prediction with LSTM." In: (cit. on pp. 23, 24, 100).

Gettman, Douglas, Alan Toppen, Kelsey Hales, Alison Voss, Shane Engel, Dayana El Azhari, Cambridge Systematics, and others (2017). *Integrating Emerging Data Sources into Operational Practice: Opportunities for Integration of Emerging Data for Traffic Management and TMCs.* Tech. rep. United States. Dept. of Transportation. ITS Joint Program Office (cit. on p. 158).

Ghosh, Bidisha, Biswajit Basu, and Margaret O'Mahony (2009). "Multivariate Short-Term Traffic Flow Forecasting Using Time-Series Analysis." In: *IEEE Transactions on Intelligent Transportation Systems* 10.2, pp. 246–254 (cit. on pp. 11, 71, 126).

Gondara, Lovedeep and Ke Wang (2017). "Multiple imputation using deep denoising autoencoders." In: *arXiv preprint arXiv:1705.02737* (cit. on p. 139).

Governments, Maricopa Association of (2020). *Newsroom | COVID-19's Impact on Regional Traffic* (cit. on p. 204).

Graettinger, Andrew, Xiao Qin, Gabriel Spear, Steven Parker, and Susie Forde (2009). "Combining state route and local road linear referencing system information." In: *Transportation Research Record: Journal of the Transportation Research Board* 2121, pp. 152–159 (cit. on pp. 157, 170).

Graves, Alex, Navdeep Jaitly, and Abdel-rahman Mohamed (2013). "Hybrid speech recognition with deep bidirectional LSTM." In: *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, pp. 273–278 (cit. on pp. 18, 26, 107, 108).

Graves, Alex and Jürgen Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures." In: *Neural Networks* 18.5-6, pp. 602–610 (cit. on p. 107).

Green, E, John Ripy, Mei Chen, and Xu Zhang (2013). "Conflation Methodologies to Incorporate Consumer Travel Data into State HPMS Datasets." In: *Transportation Research Board 92 nd Annual Meeting, Transportation Research Board*. Vol. 92, pp. 1–15 (cit. on p. 170).

Greff, Klaus, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, and Jurgen Schmidhuber (2017). "LSTM: A Search Space Odyssey." In: *IEEE Transactions on Neural Networks and Learning*

Systems 28.10, pp. 2222–2232. ISSN: 21622388. DOI: 10.1109/TNNLS.2016.2582924 (cit. on pp. 23, 24, 89, 102, 103).

Grosso, Rachel, Amanda Leahy, and Jorge Barrios (2020). *How COVID-19 Is Impacting Travel Patterns and Transportation Mode Choice*. URL: https://www.kittelson.com/ideas/how-covid-19-is-impacting-travel-patterns-and-transportation-mode-choice/ (cit. on p. 204).

Guo, Jianhua, Wei Huang, and Billy M Williams (2014). "Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification." In: *Transportation Research Part C: Emerging Technologies* 43, pp. 50–64 (cit. on p. 30).

Guo, Shengnan, Youfang Lin, Shijie Li, Zhaoming Chen, and Huaiyu Wan (2019). "Deep Spatialoral 3D Convolutional Neural Networks for Traffic Data Forecasting." In: *IEEE Transactions on Intelligent Transportation Systems* 20.10, pp. 3913–3926. ISSN: 15580016. DOI: 10.1109/TITS.2019.2906365 (cit. on pp. 12, 44).

Haklay, Mordechai and Patrick Weber (2008). "Openstreetmap: User-generated street maps." In: *Ieee Pervas Comput* 7.4, pp. 12–18 (cit. on pp. 5, 155).

Hamed, Mohammad M., Hashem R. Ai-Masaeid, and Zahi M. Bani Said (1995). "Short-term prediction of traffic volume in urban arterials." In: *Journal of Transportation Engineering* 121.3, pp. 249–254. ISSN: 0733947X. DOI: 10.1061/(ASCE)0733-947X(1995)121:3(249) (cit. on pp. 40, 58, 81).

Hammond, David K, Pierre Vandergheynst, and Rémi Gribonval (2011). "Wavelets on graphs via spectral graph theory." In: *Applied and Computational Harmonic Analysis* 30.2, pp. 129–150. ISSN: 1063-5203. DOI: 10.1016/J.ACHA.2010.04.005 (cit. on p. 69).

He, Feifei, Xuedong Yan, Yang Liu, and Lu Ma (2016). "A traffic congestion assessment method for urban road networks based on speed performance index." In: *Procedia engineering* 137, pp. 425–433 (cit. on p. 203).

Henaff, Mikael, Joan Bruna, and Yann LeCun (June 2015). "Deep Convolutional Networks on Graph-Structured Data." In: *arXiv preprint arXiv:1506.05163*. URL: http://arxiv.org/abs/1506.05163 (cit. on pp. 13, 42, 45, 68, 74).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory." In: *Neural computation* 9.8, pp. 1735–1780 (cit. on pp. 3, 12, 18, 23, 41, 44, 52, 58, 69, 97, 140).

Hu, Yue, William Barbour, Samitha Samaranayake, and Dan Work (2020). "Impacts of Covid-19 mode shift on road traffic." In: *arXiv preprint arXiv:2005.01610* (cit. on p. 205).

Hua, Jiuyi and Ardeshir Faghri (1994). "Apphcations of artificial neural networks to intelligent vehicle-highway systems." In: *Transportation Research Record* 1453, p. 83 (cit. on pp. 18, 43, 100).

Huang, Wenhao, Guojie Song, Haikun Hong, and Kunqing Xie (2014). "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning." In: *IEEE Transactions on Intelligent Transportation Systems* 15.5, pp. 2191–2201. ISSN: 15249050. DOI: 10.1109/TITS.2014.2311123 (cit. on pp. 11, 41, 43).

Hunter-Zaworski, Kate; et al. (2003). *Transportation Engineering Lab Manual*. URL: http://www.webpages.uidaho.edu/niatt_labmanual/index.htm (cit. on p. 47).

Hurley, David (2020). *Impact of COVID-19 on traffic congestion*. URL: https://towardsdatascience.com/flattening-traffic-congestion-b668b1fcda0f (cit. on p. 204).

INRIX (2020). *Monitoring Activity Re-Emergence As We Rebound From COVID-19* (cit. on p. 205).

Jagadish, H V, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi (2014). "Big data and its technical challenges." In: *Communications of the ACM* 57.7, pp. 86–94 (cit. on pp. 138, 195).

Jiang, Xiaomo and Hojjat Adeli (2004). "Wavelet Packet-Autocorrelation Function Method for Traffic Flow Pattern Analysis." In: *Computer-Aided Civil and Infrastructure Engineering* 19.5, pp. 324–337 (cit. on pp. 19, 98).

Jin, Wenwei, Youfang Lin, Zhihao Wu, and Huaiyu Wan (2018). "Spatio-temporal recurrent convolutional networks for citywide short-term crowd flows prediction." In: *ACM International Conference Proceeding Series*, pp. 28–35. ISBN: 9781450363594. DOI: 10.1145/3193077.3193082 (cit. on pp. 12, 44).

Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever (2015). "An empirical exploration of recurrent network architectures." In: *International Conference on Machine Learning*, pp. 2342–2350 (cit. on pp. 18, 97).

Kaji, Mina, Luke Barr, and Amanda Maile (2020). *Police see uptick in speeding, fatal crashes amid pandemic*. URL: https://abcnews.go.com/US/police-uptick-speeding-fatal-crashes-amid-pandemic/story?id=70751844 (cit. on p. 216).

Kalafatas, Georgios (2007). *An Exact Graph Structure for Single Destination Dynamic Traffic Assignment*. Tech. rep. (cit. on pp. 13, 45).

Kamarianakis, Yiannis, H Oliver Gao, and Poulicos Prastacos (2010). "Characterizing regimes in daily cycles of urban traffic using smooth-transition regressions." In: *Transportation Research Part C: Emerging Technologies* 18.5, pp. 821–840 (cit. on pp. 20, 98).

Kang, Bumjoon, Jason Y Scully, Orion Stewart, Philip M Hurvitz, and Anne V Moudon (2015). "Split-Match-Aggregate (SMA) algorithm: integrating sidewalk data with transportation network data in GIS." In: *International Journal of Geographical Information Science* 29.3, pp. 440–453 (cit. on pp. 158, 170).

Karlaftis, Matthew G and Eleni I Vlahogianni (2011). "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights." In: *Transportation Research Part C: Emerging Technologies* 19.3, pp. 387–399 (cit. on pp. 18, 96).

Kaushik, Kartik, Elham Sharifi, and Stanley Ernest Young (2015). "Computing Performance Measures with National Performance Management Research Data Set." In: *Transportation Research Record: Journal of the Transportation Research Board* 2529, pp. 10–26 (cit. on pp. 5, 155).

Ke, Ruimin, Wan Li, Zhiyong Cui, and Yinhai Wang (2020). "Two-Stream Multi-Channel Convolutional Neural Network for Multi-Lane Traffic Speed Prediction Considering Traffic Volume Impact." In: *Transportation Research Record: Journal of the Transportation Research Board*, p. 036119812091105. ISSN: 0361-1981. DOI: 10.1177/0361198120911052 (cit. on pp. 6, 12, 44, 71).

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (cit. on pp. 111, 141).

Kipf, Thomas N. and Max Welling (2019). "Semi-supervised classification with graph convolutional networks." In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*. arXiv:1609.02907 (cit. on pp. 13, 41, 45, 49, 51).

Klein, Lawrence A, Milton K Mills, David Gibson, and Lawrence A Klein (2006). *Traffic detector handbook: Volume II*. Tech. rep. United States. Federal Highway Administration (cit. on p. 171).

Kong, Fanhui, Jian Li, Bin Jiang, Tianyuan Zhang, and Houbing Song (2019). "Big data-driven machine learning-enabled traffic flow prediction." In: *Transactions on Emerging Telecommunications Technologies*, e3482 (cit. on pp. 3, 11, 41, 44).

Koren, Yehuda, Robert Bell, and Chris Volinsky (2009). "Matrix factorization techniques for recommender systems." In: *Computer* 42.8 (cit. on p. 101).

Kreindler, David M and Charles J Lumsden (2012). "The effects of the irregular sample and missing data in time series analysis." In: *Nonlinear Dynamical Systems Analysis for the Behavioral Sciences Using Real Data*, p. 135 (cit. on p. 101).

Ku, Wei Chiet, George R Jagadeesh, Alok Prakash, and Thambipillai Srikanthan (2016). "A clustering-based approach for data-driven imputation of missing traffic data." In: *2016 IEEE Forum on Integrated and Sustainable Transportation Systems (FISTS)*. IEEE, pp. 1–6 (cit. on pp. 13, 127).

Laña, Ibai, Ignacio (Iñaki) Olabarrieta, Manuel Vélez, and Javier Del Ser (May 2018). "On the imputation of missing data for road traffic forecasting: New insights and novel techniques." In: *Transportation Research Part C: Emerging Technologies* 90, pp. 18–33. ISSN: 0968090X. DOI: 10.1016/j.trc.2018.02.021 (cit. on pp. 124, 139).

Lange, Oliver and Luis Perez (2020). *Traffic prediction with advanced Graph Neural Networks | DeepMind*. URL: https://deepmind.com/blog/article/traffic-prediction-with-advanced-graph-neural-networks (cit. on p. 223).

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning." In: *nature* 521.7553, p. 436 (cit. on pp. 10, 19, 26, 99, 108, 186).

Lee, Jiwan and Bonghee Hong (2014). "Congestion score computation of big traffic data." In: *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*. IEEE, pp. 189–196 (cit. on p. 203).

Lee, Sangsoo and Daniel B Fambro (1999). "Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting." In: *Transportation Research Record* 1678.1, pp. 179–188 (cit. on p. 125).

Lek, S, J L Giraudel, and Jean-François Guégan (2000). "Neuronal networks: algorithms and architectures for ecologists and evolutionary ecologists." In: *Artificial Neuronal Networks*. Springer, pp. 3–27 (cit. on p. 11).

Lenzerini, Maurizio (2002). "Data integration: A theoretical perspective." In: *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, pp. 233–246 (cit. on pp. 5, 155).

Li, Fuliang, Junfeng Gong, Yunyi Liang, and Jiali Zhou (2016). "Real-time congestion prediction for urban arterials using adaptive data-driven methods." In: *Multimedia Tools and Applications* 75.24, pp. 17573–17592 (cit. on p. 186).

Li, Li, Xiaonan Su, Yi Zhang, Jianming Hu, and Zhiheng Li (2014). "Traffic prediction, data compression, abnormal data detection and missing data imputation: An integrated study based on the decomposition of traffic time series." In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 282–289 (cit. on pp. 13, 127, 128).

Li, Li, Xiaonan Su, Yi Zhang, Yuetong Lin, and Zhiheng Li (2015). "Trend modeling for traffic time series analysis: An integrated study." In: *IEEE Transactions on Intelligent Transportation Systems* 16.6, pp. 3430–3439 (cit. on pp. 2, 68, 69).

Li, Linna and Jose Valdovinos (2018). "Optimized Conflation of Authoritative and Crowd-Sourced Geographic Data: Creating an Integrated Bike Map." In: *Information Fusion and Intelligent Geographic Information Systems (IF&IGIS'17)*. Springer, pp. 227–241 (cit. on p. 158).

Li, Yaguang and Cyrus Shahabi (2018). "A brief overview of machine learning methods for short-term traffic forecasting and future directions." In: *SIGSPATIAL Special* 10.1, pp. 3–9 (cit. on p. 112).

Li, Yaguang, Rose Yu, Cyrus Shahabi, and Yan Liu (2018). "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting." In: *International Conference on Learning Representations* (cit. on pp. 10, 12, 41, 42, 44, 58, 68, 72, 110, 127, 138, 195, 197).

Liang, Yunyi, Zhiyong Cui, Yu Tian, Huimiao Chen, and Yinhai Wang (Oct. 2018). "A Deep Generative Adversarial Architecture for Network-Wide Spatial-Temporal Traffic-State Estimation." In: *Transportation Research Record* 2672.45, pp. 87–105. ISSN: 21694052. DOI: 10.1177/0361198118798737. URL: http://journals.sagepub.com/doi/10.1177/0361198118798737https://journals.sagepub.com/doi/abs/10.1177/0361198118798737 (cit. on pp. 6, 12, 44, 71, 127).

Liao, Binbing, Douglas McIlwraith, Jingqing Zhang, Tong Chen, Chao Wu, Shengwen Yang, Yike Guo, and Fei Wu (2018). "Deep sequence learning with auxiliary information for traffic prediction." In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 537–546. ISBN: 9781450355520. DOI: 10.1145/3219819.3219895 (cit. on pp. 6, 11, 12, 44, 71).

Lin, Lei, Zhengbing He, and Srinivas Peeta (2018). "Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach." In: *Transportation Research Part C: Emerging Technologies* 97, pp. 258–276 (cit. on p. 72).

Lin, Yilun, Xingyuan Dai, Li Li, and Fei Yue Wang (2019). "Pattern Sensitive Prediction of Traffic Flow Based on Generative Adversarial Framework." In: *IEEE Transactions on Intelligent Transportation Systems* 20.6, pp. 2395–2400. ISSN: 15249050. DOI: 10.1109/TITS.2018.2857224 (cit. on pp. 12, 44).

Lipton, Zachary C, David Kale, and Randall Wetzel (2016). "Directly modeling missing data in sequences with rnns: Improved classification of clinical time series." In: *Machine Learning for Healthcare Conference*, pp. 253–270 (cit. on pp. 19, 97).

List, George F et al. (2014). *Establishing Monitoring Programs for Travel Time Reliability*. Transportation Research Board (cit. on pp. 160, 177).

Longley, P A, M F Goodchild, D J Maguire, and D W Rhind (2001). *Geographic Information Systems and Science Wiley* (cit. on pp. 5, 155).

Lv, Yisheng, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang (2015). "Traffic flow prediction with big data: A deep learning approach." In: *Ieee Transactions on Intelligent Transportation Systems* 16.2, pp. 865–873 (cit. on pp. 6, 10, 11, 31, 32, 41, 44, 71, 111).

Lv, Zhongjian, Jiajie Xu, Kai Zheng, Hongzhi Yin, Pengpeng Zhao, and Xiaofang Zhou (2018). "LC-RNN: A Deep Learning Model for Traffic Speed Prediction." In: *IJCAI 2018: 27th International Joint Conference on Artificial Intelligence*, pp. 3470–3476 (cit. on pp. 6, 71).

Ma, Xiaolei, Zhuang Dai, Zhengbing He, Jihui Ma, Yunpeng Yong Wang, and Yunpeng Yong Wang (2017). "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction." In: *Sensors (Switzerland)* 17.4, p. 818. ISSN: 14248220. DOI: 10.3390/s17040818 (cit. on pp. 3, 6, 10–12, 41, 43, 44, 67, 68, 71, 100, 127).

Ma, Xiaolei, Zhimin Tao, Yunpeng Yinhai Wang, Haiyang Yu, and Yunpeng Yinhai Wang (May 2015). "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data." In: *Transportation Research Part C: Emerging Technologies* 54, pp. 187–197. ISSN: 0968090X. DOI: 10.1016/j.trc.2015.03.014. URL: https://www.sciencedirect.com/science/article/pii/S0968090X15000935 (cit. on pp. 6, 10, 12, 17–19, 23, 30, 40, 41, 44, 67, 71, 81, 96–100, 107, 127).

Ma, Xiaolei, Yao-Jan Wu, and Yinhai Wang (2011). "DRIVE Net: E-science transportation platform for data sharing, visualization, modeling, and analysis." In: *Transportation Research Record: Journal of the Transportation Research Board* 2215, pp. 37–49 (cit. on pp. 29, 156, 170).

Ma, Xiaolei et al. (Nov. 2020). "Forecasting Transportation Network Speed Using Deep Capsule Networks With Nested LSTM Models." In: *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12. ISSN: 1524-9050. DOI: 10.1109/tits.2020.2984813. URL: https://arxiv.org/abs/1811.04745 (cit. on pp. 6, 12, 44, 71, 127).

Manual, Highway Capacity (2016). "A guide for multimodal mobility analysis." In: *Transportation Research Board, Washington, DC* (cit. on p. 202).

Marchant, Andy (2020). *What Can Traffic Data Tell Us about the Impact of the Coronavirus? | TomTom Blog*. URL: https://www.tomtom.com/blog/moving-world/covid-19-traffic/ (cit. on p. 203).

Mescheder, Daniel (2020). *Can Traffic Data Help Us Find a Way out of the Lockdown? | TomTom Blog*. URL: https://www.tomtom.com/blog/moving-world/covid-19-traffic-mobility/ (cit. on p. 203).

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning." In: *Nature* 518.7540, p. 529 (cit. on pp. 10, 186).

Mondal, Debashis and Donald B Percival (2010). "Wavelet variance analysis for gappy time series." In: *Annals of the Institute of Statistical Mathematics* 62.5, pp. 943–966 (cit. on p. 101).

Okabe, Atsuyuki, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu (2009). *Spatial tessellations: concepts and applications of Voronoi diagrams*. Vol. 501. John Wiley & Sons (cit. on p. 169).

Okutani, Iwao and Yorgos J Stephanedes (1984). "Dynamic prediction of traffic volume through Kalman filtering theory." In: *Transportation Research Part B-methodological* 18.1, pp. 1–11 (cit. on pp. 11, 71, 126).

Osaba, Eneko, Pedro López-Garc\'\ia, Antonio D Masegosa, Enrique Onieva, Hugo Landaluce, and Asier Perallos (2016). "TIMON Project: Description and Preliminary Tests for Traffic Prediction Using Evolutionary Techniques." In: *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pp. 1471–1472 (cit. on pp. 14, 187).

Park, Dongjoo and Laurence R. Rilett (1999). "Forecasting freeway link travel times with a multilayer feedforward neural network." In: *Computer-Aided Civil and Infrastructure Engineering* 14.5, pp. 357–367. ISSN: 10939687. DOI: 10.1111/0885-9507.00154 (cit. on pp. 2, 10, 18, 40, 41).

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine learning in Python." In: *Journal of machine learning research* 12.Oct, pp. 2825–2830 (cit. on p. 81).

Pérez-Rendón, Antonio F and Rafael Robles (2004). "The convolution theorem for the continuous wavelet tranform." In: *Signal Processing* 84.1, pp. 55–67. ISSN: 0165-1684. DOI: 10.1016/J.SIGPRO.2003.07.014 (cit. on p. 77).

Polson, Nicholas G. and Vadim O. Sokolov (2017). "Deep learning for short-term traffic flow prediction." In: *Transportation Research Part C: Emerging Technologies* 79, pp. 1–17. ISSN: 0968090X. DOI: 10.1016/j.trc.2017.02.024 (cit. on pp. 11, 43).

Provost, Foster and Tom Fawcett (2013). "Data science and its relationship to big data and data-driven decision making." In: *Big data* 1.1, pp. 51–59 (cit. on p. 185).

Pu, Ziyuan, Chenglong Liu, Yinhai Wang, Xianming Shi, and Chao Zhang (2019). *Road Surface Condition Prediction using Long Short-Term Memory Neural Network based on Historical Data*. Tech. rep. (cit. on p. 71).

Ramsey, Paul and others (2005). "Postgis manual." In: *Refractions Research Inc* (cit. on p. 173).

Ran, Bin, Huachun Tan, Yuankai Wu, and Peter J Jin (2016). "Tensor based missing traffic data completion with spatial–temporal correlation." In: *Physica A: Statistical Mechanics and its Applications* 446, pp. 54–63 (cit. on pp. 13, 127, 128).

Rao, Amudapuram Mohan and Kalaga Ramachandra Rao (2012). "Measuring urban traffic congestion-a review." In: *International Journal for Traffic & Transport Engineering* 2.4 (cit. on p. 202).

Rue, Havard and Leonhard Held (2005). *Gaussian Markov random fields: theory and applications*. Chapman and Hall/CRC (cit. on p. 133).

SULLIVAN, CHRIS (2020). *Washington sees rise in speeding on empty freeways*. URL: https://mynorthwest.com/1831406/drivers-speeding-roads-empty/ (cit. on p. 216).

Sainath, Tara N, Oriol Vinyals, Andrew Senior, and Hasim Sak (2015). "Convolutional long short-term memory fully connected deep neural networks." In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, pp. 4580–4584 (cit. on pp. 32, 33).

Schuster, Mike and Kuldip K Paliwal (1997). "Bidirectional recurrent neural networks." In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681 (cit. on pp. 25, 106).

Seabold, Skipper and Josef Perktold (2010). "Statsmodels: Econometric and statistical modeling with python." In: *Proceedings of the 9th Python in Science Conference*. Vol. 57. Scipy, p. 61 (cit. on p. 81).

Shi, Zaixing and Ya Fang (2020). "Temporal relationship between outbound traffic from Wuhan and the 2019 coronavirus disease (COVID-19) incidence in China." In: *medRxiv* (cit. on p. 204).

Shuman, David I., Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst (2013). "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains." In: *IEEE Signal Processing Magazine* 30.3, pp. 83–98. ISSN: 10535888. DOI: 10.1109/MSP.2012.2235192 (cit. on pp. 13, 45, 51, 69, 72).

Simonovsky, Martin and Nikos Komodakis (2017). "Dynamic edge-conditioned filters in convolutional neural networks on graphs." In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Vol. 2017-Janua, pp. 29–38. ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.11 (cit. on p. 42).

Smith, Brian L, Billy M Williams, and R Keith Oswald (2002). "Comparison of parametric and nonparametric models for traffic flow forecasting." In: *Transportation Research Part C: Emerging Technologies* 10.4, pp. 303–321 (cit. on pp. 11, 71, 127).

Smola, Alex J. and Bernhard Schölkopf (2004). "A tutorial on support vector regression." In: *Statistics and Computing* 14.3, pp. 199–222. ISSN: 09603174. DOI: 10.1023/B:STCO.0000035301.49549.88 (cit. on pp. 40, 58).

Song, Xuan, Hiroshi Kanasugi, and Ryosuke Shibasaki (2016). "DeepTransport: Prediction and Simulation of Human Mobility and Transportation Mode at a Citywide Level." In: *IJCAI*, pp. 2618–2624 (cit. on pp. 12, 18, 97, 98, 100, 102).

Sun, Huijun, Jianjun Wu, Dan Ma, and Jiancheng Long (2014). "Spatial distribution complexities of traffic congestion and bottlenecks in different network topologies." In: *Applied Mathematical Modelling* 38.2, pp. 496–505. ISSN: 0307904X. DOI: 10.1016/j.apm.2013.06.027 (cit. on pp. 13, 45).

Sun, Lijun and Xinyu Chen (2019). "Bayesian Temporal Factorization for Multidimensional Time Series Prediction." In: *arXiv preprint arXiv:1910.06366* (cit. on p. 101).

Sun, Shiliang, Changshui Zhang, and Guoqiang Yu (2006). "A bayesian network approach to traffic flow forecasting." In: *IEEE Transactions on Intelligent Transportation Systems* 7.1, pp. 124–132 (cit. on pp. 11, 71, 127).

Sun, Yunchuan, Xinpei Yu, Rongfang Bie, and Houbing Song (2017). "Discovering time-dependent shortest path on traffic graph for drivers towards green driving." In: *Journal of Network and Computer Applications* 83, pp. 204–212. ISSN: 10958592. DOI: 10.1016/j.jnca.2015.10.018 (cit. on pp. 13, 45).

Tan, Huachun, Yuankai Wu, Bin Shen, Peter J Jin, and Bin Ran (2016). "Short-term traffic prediction based on dynamic tensor completion." In: *IEEE Transactions on Intelligent Transportation Systems* 17.8, pp. 2123–2133 (cit. on pp. 13, 96, 125, 127, 128).

Tang, Jinjun, Guohui Zhang, Yinhai Wang, Hua Wang, and Fang Liu (2015). "A hybrid approach to integrate fuzzy C-means based imputation method with genetic algorithm for missing traffic volume data estimation." In: *Transportation Research Part C: Emerging Technologies* 51, pp. 29–40 (cit. on pp. 13, 128).

Tardivo, Alessio, Celestino Sánchez Mart\'\in, and Armando Carrillo Zanuy (2020). "Covid-19 impact in Transport, an essay from the Railways' system research perspective." In: (cit. on p. 205).

Taylor, Michael A P, Jeremy E Woolley, and Rocco Zito (2000). "Integration of the global positioning system and geographical information systems for traffic congestion studies." In: *Transportation Research Part C: Emerging Technologies* 8.1-6, pp. 257–285 (cit. on p. 158).

Tian, Yan, Kaili Zhang, Jianyuan Li, Xianxuan Lin, and Bailin Yang (Nov. 2018). "LSTM-based traffic flow prediction with missing data." In: *Neurocomputing* 318, pp. 297–305. ISSN: 0925-2312. DOI: 10.1016/J.NEUCOM.2018.08.067. URL: https://www.sciencedirect.com/science/article/pii/S0925231218310294 (cit. on pp. 14, 125, 128, 140, 141).

Tieleman, Tijmen and Geoffrey Hinton (2012). "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." In: *COURSERA: Neural networks for machine learning* 4.2, pp. 26–31 (cit. on pp. 59, 82).

Turner, Shawn M, Rich Margiotta, Tim Lomax, and others (2004). *Lessons learned: monitoring highway congestion and reliability using archived traffic detector data*. Tech. rep. United States. Federal Highway Administration. Office of Operations (cit. on p. 205).

Valadkhani, Amir Hosein, Yang Hong, and Mohsen Ramezani (2017). "Integration of loop and probe data for traffic state estimation on freeway and signalized arterial links." In: *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*. IEEE, pp. 1–6 (cit. on pp. 157, 170).

Van Lint, J W C (2008). "Online learning solutions for freeway travel time prediction." In: *IEEE Transactions on Intelligent Transportation Systems* 9.1, pp. 38–47 (cit. on p. 96).

Van Lint, J. W.C., S. P. Hooqendoorn, and H. J. Van Zuvlen (2002). "Freeway travel time prediction with state-space neural networks modeling state-space dynamics with recurrent neural networks." In: *Transportation Research Record* 1811, pp. 30–39. ISSN: 03611981. DOI: 10.3141/1811-04 (cit. on pp. 11, 18, 41, 43, 100).

Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumitru Erhan (2015). "Show and tell: A neural image caption generator." In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, pp. 3156–3164 (cit. on p. 18).

Vlahogianni, Eleni I., John C. Golias, and Matthew G. Karlaftis (Sept. 2004). "Short-term traffic forecasting: Overview of objectives and methods." In: *Transport Reviews* 24.5, pp. 533–557. ISSN: 0144-1647. DOI: 10.1080/0144164042000195072. URL: http://www.tandfonline.com/doi/abs/10.1080/0144164042000195072 (cit. on pp. 2, 10, 67).

Vlahogianni, Eleni I, Matthew G Karlaftis, and John C Golias (2014). "Short-term traffic forecasting: Where we are and where we're going." In: *Transportation Research Part C: Emerging Technologies* 43, pp. 3–19 (cit. on pp. 2, 40, 67, 96, 124, 126).

Wang, Jiawei, Ruixiang Chen, and Zhaocheng He (2019a). "Traffic speed prediction for urban transportation network: A path based deep learning approach." In: *Transportation Research Part C: Emerging Technologies* 100, pp. 372–385 (cit. on pp. 6, 71).

Wang, Meng-Di, Qi-Rong Qiu, and Bing-Wei Cui (2012). "Short-term wind speed forecasting combined time series method and arch model." In: *Machine Learning and Cybernetics (ICMLC), 2012 International Conference on*. Vol. 3. IEEE, pp. 924–927 (cit. on p. 19).

Wang, Xiangrong and Piet Van Mieghem (Nov. 2015). "Orthogonal Eigenvector Matrix of the Laplacian." In: *2015 11th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, pp. 358–365. ISBN: 978-1-4673-9721-6. DOI: 10.1109/SITIS.2015.35. URL: http://ieeexplore.ieee.org/document/7400588/ (cit. on p. 137).

Wang, Xiao, Rui Jiang, Li Li, Yilun Lin, Xinhu Zheng, and Fei-Yue Wang (2017). "Capturing car-following behaviors by deep learning." In: *IEEE Transactions on Intelligent Transportation Systems* 19.3, pp. 910–920 (cit. on pp. 10, 186).

Wang, Yinhai, Xuegang (Jeff) Ban, Zhiyong Cui, and Meixin Zhu (June 2019b). *Freeway Inductive Loop Detector Dataset for Network-wide Traffic Speed Prediction*. DOI: 10.5281/zenodo.3258904. URL: https://doi.org/10.5281/zenodo.3258904 (cit. on pp. 100, 110).

Wang, Yinhai, Xuegang Jeff Ban, Zhiyong Cui, Meixin Zhu, and others (2019c). "An artificial intelligence platform for network-wide congestion detection and prediction using multi-source data." In: (cit. on p. 192).

Wang, Yinhai, Ruimin Ke, Weibin Zhang, Zhiyong Cui, and Kris Henrickson (2016). "Digital Roadway Interactive Visualization and Evaluation Network Applications to WSDOT Operational Data Usage." In: *Diss. University of Washington Seattle, Washington* (cit. on pp. 28, 29, 80).

Wells, Brian J, Kevin M Chagin, Amy S Nowacki, and Michael W Kattan (2013). "Strategies for handling missing data in electronic health record derived data." In: *eGEMs* 1.3 (cit. on p. 97).

Williams, Billy M and Lester A Hoel (2003). "Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results." In: *Journal of transportation engineering* 129.6, pp. 664–672 (cit. on pp. 96, 111).

Williams, Billy (2001). "Multivariate vehicular traffic flow prediction: evaluation of ARIMAX modeling." In: *Transportation Research Record: Journal of the Transportation Research Board* 1776, pp. 194–200 (cit. on pp. 11, 71, 126).

Wu, Chun-Hsin, Jan-Ming Ho, and Der-Tsai Lee (2004). "Travel-time prediction with support vector regression." In: *IEEE transactions on intelligent transportation systems* 5.4, pp. 276–281 (cit. on pp. 11, 30, 31, 71, 81, 111, 127).

Wu, Yuankai and Huachun Tan (2016). "Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework." In: *arXiv preprint arXiv:1612.01022* (cit. on pp. 18, 19, 33, 97, 99, 102, 107).

Wu, Yuankai, Huachun Tan, Lingqiao Qin, Bin Ran, and Zhuxi Jiang (May 2018). "A hybrid deep learning based traffic flow prediction method and its understanding." In: *Transportation Research Part C: Emerging Technologies* 90, pp. 166–180. ISSN: 0968090X. DOI: 10.1016/j.trc.2018.03.001.

URL: https://www.sciencedirect.com/science/article/pii/S0968090X18302651 (cit. on pp. 6, 12, 44, 71).

Xu, Bingbing, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng (2019). "Graph Wavelet Neural Network." In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=H1ewdiR5tQ (cit. on pp. 13, 69, 72, 77).

Yang, Hao, Chenxi Liu, Christopher Gottsacker, Xuegang Ban, Chao Zhang, and Yinhai Wang (2019). *Cell-Speed Prediction Neural Network (CPNN): A Deep Learning Approach for Trip-Based Speed Prediction*. Tech. rep. (cit. on p. 127).

Yao, Huaxiu, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li (2018). "Deep multi-view spatial-temporal network for taxi demand prediction." In: *Thirty-Second AAAI Conference on Artificial Intelligence* (cit. on p. 72).

Ye, Qing, Wai Yuen Szeto, and Sze Chun Wong (2012). "Short-term traffic speed forecasting based on data recorded at irregular intervals." In: *IEEE Transactions on Intelligent Transportation Systems* 13.4, pp. 1727–1737 (cit. on p. 18).

Yin, Hongbin, S. C. Wong, Jianmin Xu, and C. K. Wong (2002). "Urban traffic flow prediction using a fuzzy-neural approach." In: *Transportation Research Part C: Emerging Technologies* 10.2, pp. 85–98. ISSN: 0968090X. DOI: 10.1016/S0968-090X(01)00004-3 (cit. on pp. 11, 18, 43, 100).

Yoon, Jinsung, James Jordon, and Mihaela Schaar (2018). "GAIN: Missing Data Imputation using Generative Adversarial Nets." In: *International Conference on Machine Learning*, pp. 5675–5684 (cit. on pp. 13, 128, 130).

Yu, Bing, Mengzhang Li, Jiyong Zhang, and Zhanxing Zhu (Mar. 2019). "3D Graph Convolutional Networks with Temporal Graphs: A Spatial Information Free Framework For Traffic Forecasting." In: URL: http://arxiv.org/abs/1903.00919 (cit. on p. 41).

Yu, Bing, Haoteng Yin, and Zhanxing Zhu (2018). "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting." In: *IJCAI International Joint Conference on Artificial Intelligence* 2018-July, pp. 3634–3640. ISSN: 10450823. DOI: 10.24963/ijcai.2018/505 (cit. on pp. 3, 12, 13, 44, 45, 68, 72, 127).

Yu, Haiyang, Zhihai Wu, Shuqin Wang, Yunpeng Wang, and Xiaolei Ma (2017a). "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks." In: *Sensors*

(Switzerland) 17.7, p. 1501. ISSN: 14248220. DOI: 10.3390/s17071501 (cit. on pp. 3, 11, 40, 41, 43, 44, 68, 98, 100, 102, 127).

Yu, Rose, Yaguang Li, Cyrus Shahabi, Ugur Demiryurek, and Yan Liu (2017b). "Deep learning: A generic approach for extreme condition traffic forecasting." In: *Proceedings of the 17th SIAM International Conference on Data Mining, SDM 2017*. SIAM, pp. 777–785. ISBN: 9781611974874. DOI: 10.1137/1.9781611974973.87 (cit. on pp. 3, 12, 18, 19, 32, 41, 44, 99, 107, 111).

Zarei, Narjes, Mohammad Ali Ghayour, and Sattar Hashemi (2013). "Road traffic prediction using context-aware random forest based on volatility nature of traffic flows." In: *Asian Conference on Intelligent Information and Database Systems*. Springer, pp. 196–205 (cit. on p. 111).

Zegeer, John et al. (2013). *Incorporating travel time reliability into the Highway Capacity Manual*. Tech. rep. United States. National Transportation Library [distributor] (cit. on p. 177).

Zhang, Chaoyun and Paul Patras (2018). "Long-term mobile traffic forecasting using deep Spatio-Temporal neural networks." In: *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. ACM, pp. 231–240. ISBN: 9781450357708. DOI: 10.1145/3209582.3209606 (cit. on pp. 12, 44).

Zhang, Da and Mansur R. Kabuka (2018). "Combining weather condition data to predict traffic flow: A GRU-based deep learning approach." In: *IET Intelligent Transport Systems* 12.7, pp. 578–585. ISSN: 1751956X. DOI: 10.1049/iet-its.2017.0313 (cit. on pp. 12, 44, 45).

Zhang, Jiani, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung (2018). "Gaan: Gated attention networks for learning on large and spatiotemporal graphs." In: *arXiv preprint arXiv:1803.07294* (cit. on pp. 13, 68, 72).

Zhang, Junbo, Yu Zheng, and Dekang Qi (2017). "Deep spatio-temporal residual networks for citywide crowd flows prediction." In: *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 1655–1661 (cit. on p. 41).

Zhang, Lei, Sepehr Ghader, Michael L Pack, Chenfeng Xiong, Aref Darzi, Mofeng Yang, Qianqian Sun, AliAkbar Kabiri, and Songhua Hu (2020). "An interactive COVID-19 mobility impact and social distancing analysis platform." In: *medRxiv* (cit. on p. 204).

Zhang, Yanru and Yunlong Zhang (2016). "A comparative study of three multivariate short-term freeway traffic flow forecasting methods with missing data." In: *Journal of Intelligent Transportation Systems* 20.3, pp. 205–218 (cit. on p. 125).

Zhao, Zheng, Weihai Chen, Xingming Wu, Peter C.V. Chen, and Jingmeng Liu (2017). "LSTM network: A deep learning approach for short-term traffic forecast." In: *IET Image Processing* 11.1, pp. 68–75. ISSN: 17519659. DOI: 10.1049/iet-its.2016.0208 (cit. on pp. 12, 30, 44, 97, 100, 102).

Zheng, Xiaoping and Mengting Liu (2009). "An overview of accident forecasting methodologies." In: *Journal of Loss Prevention in the process Industries* 22.4, pp. 484–491 (cit. on p. 19).

Zhou, Mofan, Xiaobo Qu, and Xiaopeng Li (2017). "A recurrent neural network based microscopic car following model to predict traffic oscillation." In: *Transportation research part C: emerging technologies* 84, pp. 245–264 (cit. on pp. 10, 13, 186).

Zhou, Zhenpeng and Xiaocheng Li (2017). "Graph Convolution: A High-Order and Adaptive Approach." In: *arXiv preprint arXiv:1706.09916*. URL: http://arxiv.org/abs/1706.09916 (cit. on pp. 42, 45, 49, 72).

Zhu, Meixin, Xuesong Wang, and Yinhai Wang (2018). "Human-like autonomous car-following model with deep reinforcement learning." In: *Transportation research part C: emerging technologies* 97, pp. 348–368 (cit. on p. 186).