Machine Learning Models for Traffic Flow Prediction

by

Ta Jiun Ting

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

Graduate Department of Mechanical and Industrial Engineering
University of Toronto

Machine Learning Models for Traffic Flow Prediction

Ta Jiun Ting
Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto
2021

Abstract

Many methods of traffic prediction have been proposed over the years, from the time series models over 40 years ago to the latest deep learning models today, which prompts the need for an in-depth comparison and the critical question of whether deep learning offers significant improvements over the traditional methods. This thesis addresses this situation by systematically evaluating the different methods. We first procure a diverse set of traffic data from simulation software and real-world sensors. We then compare the different methods and perform further analysis on the latest deep learning models. Finally, we also consider the task of predicting long-term traffic up to a week in advance. Overall, we demonstrate that deep learning models are effective in short-term prediction. However, the classical random forest regression provides the best performance in both short-term and long-term prediction, which suggests that there is still room for improvement for deep learning methods.

Acknowledgments

First and foremost, I would like to express my deepest appreciation to Professor Scott Sanner and Professor Baher Abdulhai for their tireless guidance and supervision throughout the two years of my master's study. I would also like to extend my deepest gratitude to my family for their unrelenting encouragement, this accomplishment would not have been possible without their support. I must also thank Xiaoyu, Parth, Xiaocan, and others in the traffic research group as well as members of the D3M lab for their camaraderie over the past two years.

During my master's study, we saw the onset of a global pandemic that sent the world into unprecedented chaos and all research activities were affected. I am tremendously thankful for my supervisors, my family, and others at the University that made it possible to continue my research and finish this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Traffic congestion imposes a significant cost on the economic prosperity of a region. This cost not only materializes as an opportunity cost to the travellers due to time delays but also as a negative externality in the form of increased accident rate and vehicle emissions [1]. In the Greater Toronto and Hamilton Area, the estimated total cost of traffic congestion was $6 billion in 2006, and this cost is projected to exceed $15 billion by 2031 [1]. Evidently, measures that reduce traffic congestion have significant economic benefits.

As land becomes more scarce in cities and the cost of new roads increases, it is less feasible to alleviate congestion by building more roads. Therefore, intelligent transportation systems (ITS) address congestion through effective traffic management aided by technology, such as employing variable speed limits and adaptive traffic signal control. Accurate traffic prediction is essential in this strategy as it allows these management technologies to be proactive rather than reactive. In adaptive traffic signal control, the adaptive system uses the predicted traffic information to optimize the phase timing, thus accurate predictions allow the adaptive system to respond optimally [2]. Similarly, in the case of variable speed limits, accurate prediction allows the system to better assess the impact of its decisions and set the optimal speed limit [3]. Accurate prediction also allows users to make more informed travel decisions and make timely adjustments in response to traffic incidents and congestion. Altogether, accurate prediction leads to reduced road congestion by better managing the demand for a transportation system.

Traffic prediction and modelling have been studied extensively since the work of Lighthill and Whitham in 1955 [4]. As computational and data collection technology improved, researchers applied increasingly complex methods to this problem including

time series analysis, regression analysis, artificial neural networks, and deep neural networks. Notably, in the past 5 years, the prevailing model for traffic prediction has shifted to graph neural networks (GNNs) [5]–[8], a type of artificial neural network that generates predictions by exploiting the road topology to detect correlations between different road segments. These methods are discussed in further detail in Chapter 2 of this thesis.

In our review of the recent literature, we noticed that while there is a continual development of new graph neural network variants, there is a distinct lack of comprehensive evaluation against other types of traffic prediction methods. In addition, while the prediction errors are seemingly lowered with each new improvement, there lacks a broad comparison between graph neural networks to identify the components and techniques that are crucial to performance. Therefore, our work begins with a thorough evaluation of the different methods first as an attempt to validate the recent advancements in the field of traffic prediction, also to provide insights into the success or failure of the different models in various settings.

1.2 Contributions

The goal of this thesis is to examine the plethora of prediction models that exist in the literature in order to identify candidate models for field deployment and potential directions for future traffic research. The main contributions of this thesis are threefold.

1. We conduct a comprehensive survey of the different classes of short-term traffic prediction methods that have appeared over the past 50 years. Afterwards, we select representative models from each class and conduct a thorough evaluation of their capabilities. In this analysis, we employ traffic simulation software to generate data that includes various traffic scenarios potentially seen in the real world. We demonstrate the effectiveness of regression analysis techniques and identify useful input features to a traffic prediction model. Furthermore, we highlight the detrimental effects of sharing model parameters in GNNs, which motivates the further investigation that culminates in the second contribution of this thesis. Lastly, to the best of our knowledge, this is the first comprehensive study that evaluates different classical and contemporary traffic prediction models in a variety of settings with the aid of traffic simulation software.

2. We perform an in-depth evaluation of GNNs that have appeared in the recent traffic prediction literature by analyzing the effects of individual GNN components on prediction performance. We demonstrate that the state-of-the-art

GNNs can potentially learn a model that indicates direct traffic influence among faraway links, which is inconsistent with traffic behaviour. Lastly, we showcase that the performance of GNNs may be overstated in the literature since their prediction accuracy is largely comparable to the traditional machine learning model of random forest that predates GNNs by over 15 years; however, it is important to note that GNNs contain significantly fewer model parameters.

3. We create a traffic prediction toolkit that includes all methods in our evaluations, including both short-term and long-term predictions. This toolkit is beneficial for future research in this area as it allows us to easily compare and visualize different methods. Initially, we integrate this toolkit with a traffic model for the Greater Toronto and Hamilton Area created in the Aimsun [9] traffic simulation software.

1.3 Outline

The remainder of this thesis proceeds as follows. Chapter 2 introduces the relevant background material, including a more precise definition of the traffic prediction problem, the common evaluation metrics and an overview of the evolution of traffic prediction models over the past 50 years. Afterwards, Chapter 3 discusses the methodology and findings of our evaluation of the different classes of traffic prediction methods using simulation data, which corresponds to our first contribution. Meanwhile, Chapter 4 focuses on our analysis of graph neural networks and their components, which corresponds to our second contribution. Subsequently, Chapter 5 examines the methods concerning long-term traffic prediction with prediction horizons that are multiple days in advance, where we assess several recent time series forecasting methods in the literature by comparing them against classical methods. Finally, Chapter 6 is the conclusion of this thesis, where we summarize our findings and provide recommendations for future traffic prediction research for both short-term and long-term predictions.

Chapter 2

Background

2.1 Overview

This chapter introduces the background material relevant to the contribution of this thesis. We begin by defining the traffic prediction problem in Section 2.2 and the evaluation metrics in Section 2.3. Afterwards, we broadly classify the different traffic prediction methods that are present in the literature into four categories and discuss them in the four remaining sections of this chapter. This discussion is not intended to be an exhaustive survey; in particular, the primary focus is placed on methods that are relevant to our contributions.

2.2 Problem Definition

Road traffic is a highly dynamic process evolving over space and time. For a stretch of highway, traffic conditions at a given location influence its downstream sections as traffic moves forward. Meanwhile, traffic conditions at a given location also influence its upstream sections as traffic congestion propagates backwards via shock waves, a phenomenon known since the 1950s [10]. These two basic processes are constantly present on any road network, evolving continuously in response to the changes in traffic conditions. In the urban setting, the presence of intersections and traffic signals further influences the dynamics of traffic patterns and complicates the prediction problem.

Traffic prediction can be performed at different levels, from the behaviours of individual vehicles to the traffic states of entire districts. The prediction methods vary between the different levels. At the individual vehicle level, the predictor models the behaviour of each vehicle to predict future trajectories. Meanwhile, at the link level, the predictor focuses on the macroscopic properties such as traffic speeds, flows,

and densities on each road and predicting their evolution over time. For this thesis, we define the prediction problem as link-level prediction and the task of predicting individual vehicle properties is considered out of scope. Furthermore, when discussing applications of graph neural networks in traffic prediction, we may also refer to road links as nodes, and connections between road links (such as road intersections) as edges.

We can represent the traffic properties on each link with a variety of variables, including:

- Flow (q): number of vehicles passing a point or the link per unit time
- Speed (v): distance covered per unit time
- Density (ρ): number of vehicles per unit distance at a given instance in time
- Travel time (TT): time to traverse a given link
- Occupancy (occ): percentage of time a detector is occupied by vehicles
- Headway: average distance or time between vehicles

The above quantities are related to one another in the equations below.

$$q = \rho \cdot v \tag{2.1}$$

$$\text{TT} = \text{link distance} \cdot v^{-1} \tag{2.2}$$

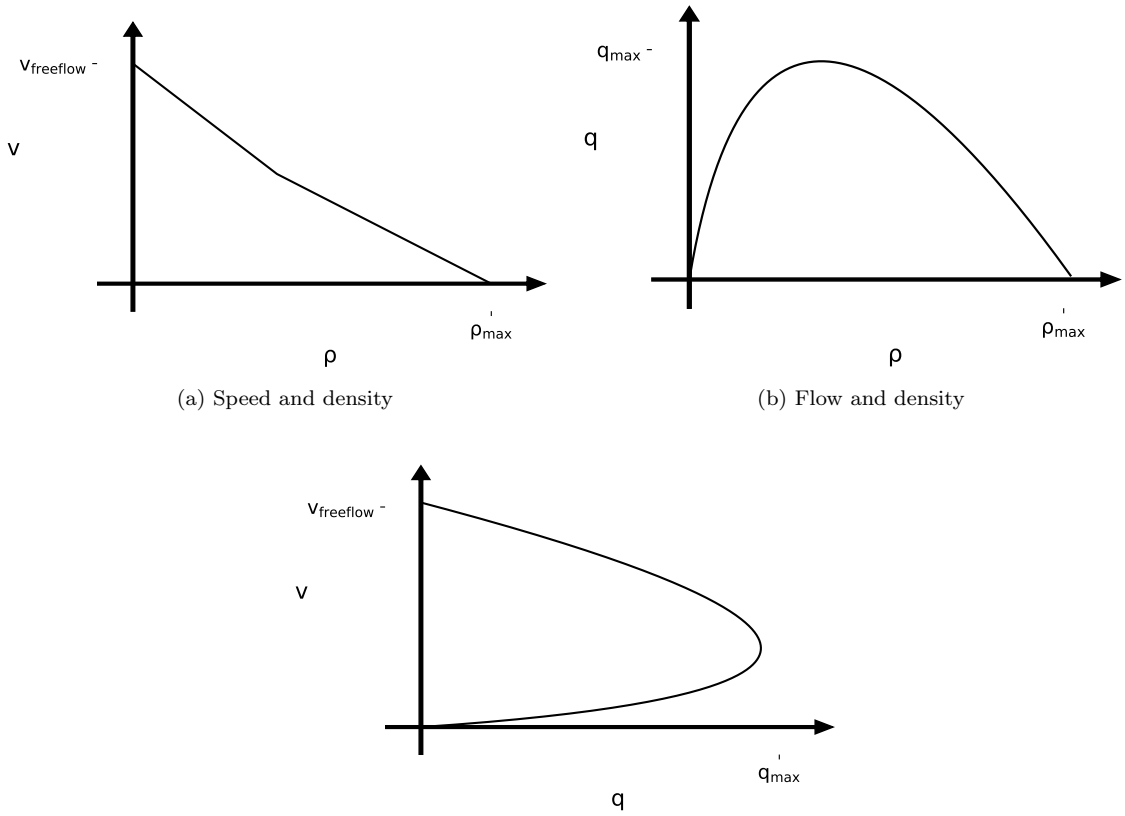
$$\text{headway} = q^{-1} \tag{2.3}$$

$$q = c \cdot \text{occ} \cdot v \tag{2.4}$$

where c is a constant of proportionality that relates the two quantities.

Furthermore, the relationship between flow, speed, and density can be described using the macroscopic fundamental diagram shown in Figure 2.1. The speed on a link when a road is empty is known as its free-flow speed, $v_{freeflow}$. As the density increases, the speed decreases until the maximum density, ρ_{max}, is reached. At this point, no additional vehicle can be packed onto the road. Using these values, the corresponding flow as well as the link capacity, q_{max}, can then be calculated with Equation (2.1). Figure 2.1 shows the relationship between speed and density as linear; however, the relationship is much more complicated in practice. A more detailed description of this topic can be found in [11]. It is important to note that

the diagram is specific to each road because values such as free-flow speed and max density are highly dependent on road characteristics.



(a) Speed and density

(b) Flow and density

(c) Speed and flow

Figure 2.1: The macroscopic fundamental diagram of traffic

Typically, only a small subset of these properties are available to the transportation agency depending on the sensors installed. Consequently, traffic prediction typically only predicts flow or speed depending on the source of data. Loop detectors installed in the road can produce accurate vehicle count data that allows us to calculate flow by counting the number of vehicles passing through in a unit time; however, it is impossible to obtain information such as speed or density with a single loop detector since they require observation over a distance rather than a single point. Meanwhile, GPS or Bluetooth data contains locations and timestamps that allow us to calculate average vehicle speed by dividing distance traversed over time elapsed; yet flow and density information from this data can be inaccurate since some vehicles may lack the technology and would be unaccounted for in the data. In practice, other factors such as weather and road design also influence traffic patterns; however, similar to other works on this topic, this thesis only uses the past observations and the graph

structure of the road network as inputs to our prediction models.

The time horizons of the predicted values depend heavily on the time granularity of the available data (they are the same in the majority of cases). Even though traffic patterns evolve continuously on a road network, data are usually aggregated and discretized over time as they are collected. This is done to reduce the size of the data set and to reduce variability in the data by increasing the sample size of each period. Consequently, the time horizons of the prediction model need to be a multiple of the time granularity of the dataset; otherwise, we cannot easily verify the accuracy of the prediction model using the available data.

The following notations are used throughout this thesis:

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: The directed graph which describes the road network. $\mathcal{V}$ is the set of nodes, which represents the links, while $|\mathcal{V}| = N$. $\mathcal{E}$ is the set of edges, which represents the intersections of the road network.
- $\mathcal{N}(i)$: The set of nodes in the neighbourhood of node i. This is not restricted to the immediate neighbours of node i, and also includes node i itself.
- $\mathbf{x}_i^{(t)}$: A vector with length d that represents the observation of node i at time t.
- $\mathbf{X}^{(t)}$: A matrix with size $(N \times d)$ that represents the observation of the entire road network at time t.
- $\hat{\mathbf{x}}_i^{(t)}$: A vector with length d that represents the prediction of node i at time t.
- $\hat{\mathbf{X}}^{(t)}$: A matrix with size $(N \times d)$ that represents the prediction of the entire road network at time t.
- H: The prediction horizon.

Using the above notation, we can define the prediction problem as learning a function f that maps the past observations to predictions using the graph $\mathcal{G}$ as follows:

$$\hat{\mathbf{X}}^{(t+1)}, \hat{\mathbf{X}}^{(t+2)}, ..., \hat{\mathbf{X}}^{(t+H)} = f\left(\mathbf{X}^{(t)}, \mathbf{X}^{(t-1)}, ..., \mathcal{G}\right) \tag{2.5}$$

2.3 Evaluation Metrics

The end objective of a traffic prediction method is to generate accurate predictions of future traffic states by employing currently available information. This information may include past observations, road network characteristics, as well as any useful external information such as weather and the presence of traffic events. The output

of the model is the predicted state values of each location in the road network over each time slice in the prediction horizon.

The goal of a predictive model is to minimize the difference between the predicted state values and the actual state values. To quantify this numerically, researchers typically use a combination of the following time series regression metrics to assess the performance of predictive models: mean absolute error (MAE), mean absolute percentage error (MAPE), mean-square error (MSE), and root-mean-square error (RMSE). MAE is the average of the absolute error across all predictions while MAPE is the average of the absolute relative error that emphasizes lower values. Meanwhile, MSE is the averaged squared error, which also represents the variance of prediction errors. Lastly, RMSE is the square root of the average squared errors, which is also the standard deviation of prediction errors. For a prediction horizon H, given predictions $\hat{\mathbf{x}}_i^{(t+1)}, \hat{\mathbf{x}}_i^{(t+2)}, ..., \hat{\mathbf{x}}_i^{(t+H)}$ and the actual observed value $\mathbf{x}_i^{(t+1)}, \mathbf{x}_i^{(t+2)}, ..., \mathbf{x}_i^{(t+H)}$ for N different samples, we can calculate the four metrics using the equations below:

$$\text{MAE} = \frac{1}{N}\frac{1}{|\mathcal{V}|}\frac{1}{H}\sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{V}|}\sum_{k=1}^{H}\left\|\mathbf{x}_{i,j}^{(t+k)} - \hat{\mathbf{x}}_{i,j}^{(t+k)}\right\|_1 \tag{2.6}$$

$$\text{MAPE} = \frac{1}{N}\frac{1}{|\mathcal{V}|}\frac{1}{H}\sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{V}|}\sum_{k=1}^{H}\left\|\frac{\mathbf{x}_{i,j}^{(t+k)} - \hat{\mathbf{x}}_{i,j}^{(t+k)}}{\mathbf{x}_{i,j}^{(t+k)}}\right\|_1 \tag{2.7}$$

$$\text{MSE} = \frac{1}{N}\frac{1}{|\mathcal{V}|}\frac{1}{H}\sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{V}|}\sum_{k=1}^{H}\left\|\mathbf{x}_{i,j}^{(t+k)} - \hat{\mathbf{x}}_{i,j}^{(t+k)}\right\|_2 \tag{2.8}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\frac{1}{|\mathcal{V}|}\frac{1}{H}\sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{V}|}\sum_{k=1}^{H}\left\|\mathbf{x}_{i,j}^{(t+k)} - \hat{\mathbf{x}}_{i,j}^{(t+k)}\right\|_2} \tag{2.9}$$

2.4 Prediction with Dynamic Traffic Simulation Models

Traffic prediction and modelling have been studied since the 1950s. This section covers the traditional methods which model traffic dynamics explicitly using the observed behaviours of traffic. In contrast, the newer methods adopt data-driven techniques such as time series analysis, regression analysis, or deep learning. This thesis focuses on data-driven techniques and does not experiment with the methods described in this section; however, we include their discussion in this chapter due to their significant and historical contribution to this field. These methods can be roughly divided into two categories: the macroscopic approach and the microscopic approach.

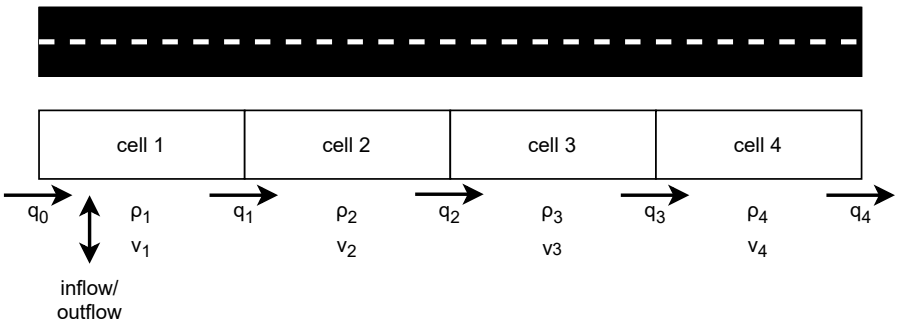


Figure 2.2: The cell transmission model

2.4.1 Macroscopic Approach

The macroscopic approach to traffic modelling treats the stream of traffic on a road link as a collective, homogeneous entity evolving over time due to influences from upstream and downstream links. In 1956, Lighthill and Whitham [4] pioneered this approach by creating the kinematic wave model. This model observes the phenomenon of traffic waves propagating slowly along a road as similar to the kinematic waves in fluid dynamics. Consequently, this model applies the kinematic wave equations to describe the change in flow, density, and speed along a road. The notion of traffic shock waves augments this model by explaining the cause of congestion accumulation and dissipation on a road [10].

The simplicity of the kinematic wave model allows a numerical solution to be described, known as the cell transmission model (CTM) [12]. The CTM works by discretizing a road into cells, as shown by Figure 2.2, where the length of each cell is equal to the distance travelled in one time step at the free-flow speed. Given the speed v or density ρ of each cell as well as any inflows and outflows from the corridor, we can compute the flow q between consecutive cells using the fundamental diagram in Figure 2.1. Subsequently, based on conservation of inflows and outflows at each cell, we can calculate the density of each cell at the next time step. Therefore, if the initial conditions of the road are known, we can iteratively update the cells and predict future traffic conditions along a road corridor. A more detailed description of the cell transmission model can be found in [12]. Extensions to the CTM include applying CTM to the analysis of complex road networks [13], and optimizing the CTM by reducing the number of cells [14]. Moreover, the works of [15] and [16] show that CTM theory can also be applied to traffic signal control.

One limitation of the first-order kinematic wave model is that it assumes instantaneous change in speed with corresponding changes in density, which implies infinite acceleration. Therefore, in the 1970s, there were attempts to improve the first-order

kinematic wave model by approximation with a second-order fluid dynamics model [17], [18]. The second-order model introduces a momentum equation to capture the finer dynamics of the fluid-like behaviour of traffic. However, the fundamental differences between traffic and fluids cause second or higher-order fluid dynamics models to produce unrealistic results [19]. Nevertheless, there is ongoing research in improving second-order models by relaxing the assumptions and constraints applicable only to fluids [20].

2.4.2 Microscopic Approach

The microscopic approach to traffic modelling considers the collective behaviour of a traffic stream stems from the behaviour of individual vehicles within the stream. A car-following model achieves this by describing how vehicles act relative to their preceding vehicle. Mainly, each vehicle will maintain a gap with its preceding vehicle by changing its speed based on the behaviour of the leading car. [21]–[23] are examples of this model, and the modelling results are in agreement with the macroscopic models discussed in Section 2.4.1 [24]. This model is more descriptive than the kinematic wave model since it relates the macroscopic traffic properties to the behaviour of individual vehicles. However, this also increases model complexity significantly, and it is difficult to tune the model parameters to achieve a realistic driver behaviour. Today, car-following models are commonly found in traffic simulation software such as Aimsun Live [9] and SUMO [25]. Figure 2.3 is a screenshot of the SUMO traffic simulator, where each triangle represents a vehicle and interactions among vehicles are described with a car-following model.

2.5 Time Series Models

As data collection and processing technology improved over the years, a new class of traffic prediction models emerged. Researchers began to view the evolving macroscopic traffic properties on a road as time series and applied time series analysis to predict traffic in the 1970s. In contrast to the methods described in Section 2.4, these methods do not model traffic dynamics explicitly. Instead, these methods use the available data to estimate the parameters of the model.

2.5.1 ARIMA Models

The ARIMA (autoregressive integrated moving average) model is a class of time series models that is extensively researched in traffic prediction [26], [27]. This type

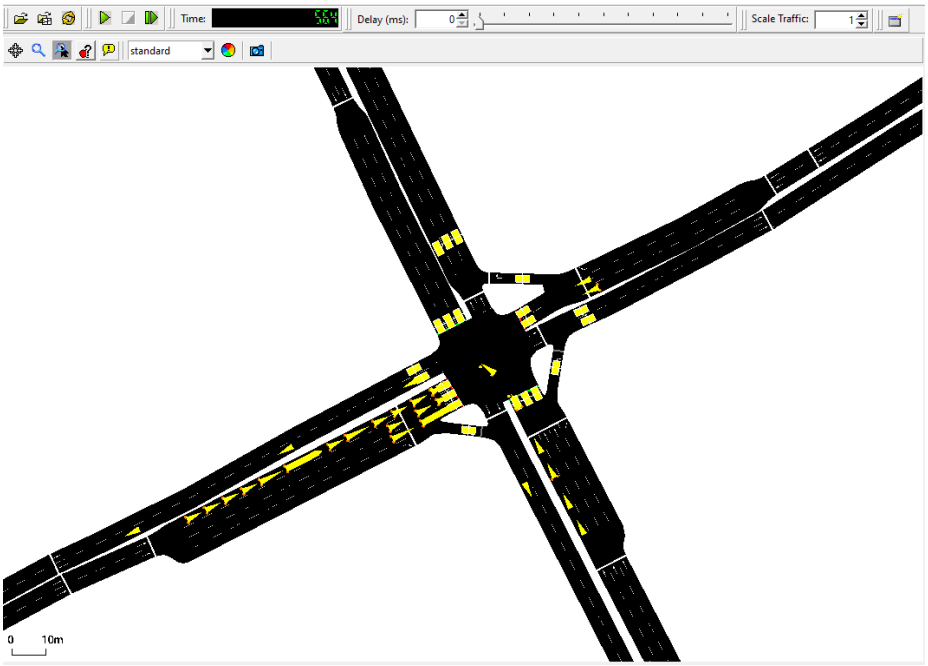


Figure 2.3: The SUMO traffic simulator

of model considers each observation to be correlated with its preceding observations. We can decompose the ARIMA model into its 3 components: autoregression (AR), differencing (also known as integration, I), and moving average (MA).

The autoregressive component assumes that the observation for link i at time t is related to the linear combination of its past p observations. Meanwhile, the moving average component additionally assumes that the observation is related to the linear combination of past q prediction errors, where error refers to the difference between the predicted and observed values. The ARIMA model without the differencing component is shown in Equation (2.10), where ψ_{ij} and ω_{ik} are respectively the parameters to the autoregressive and the moving average components, while c and ϵ are respectively the constant and the error terms.

$$\hat{x_i}^{(t+1)} = c + \sum_{j=0}^{p} \psi_{ij} x_i^{(t-j)} + \sum_{k=0}^{q} \omega_{ik} \epsilon_i^{(t-k)} + \epsilon_i^{(t+1)} \tag{2.10}$$

Finally, the full ARIMA model adds a differencing component, which transforms the initial time series into a series of differences between consecutive observations in the time series. The differencing is performed before fitting the model in Equation (2.10) in order to transform a non-stationary time series to become stationary, which means that the properties of the time series are constant over time. Based on this description, an ARIMA model can be described using a set of 3 numbers (p, d, q),

where p is the order (number of time lags) of the autoregressive component, d is the number of times first-order differencing is applied, and q is the order of the moving average component. The procedure of determining the orders and calculating the model parameters are specified by [28]. It should be noted that the ARIMA model is a univariate model, thus road segments would be decoupled from one another and analyzed individually through time in traffic prediction applications. In this thesis, we implement the ARIMA models using the pmdarima [29] and statsmodels [30] Python packages.

The ARIMA model can be augmented by incorporating additional terms that describe the seasonality of a time series, known as the seasonal ARIMA model. The seasonal part of the seasonal ARIMA model also contains autoregression, differencing, and moving average components. However, instead of past observations, seasonal differencing computes the differences between an observation and the observation from previous seasons. Similarly, the observations from previous seasons are used to compute the AR and the MA components in the seasonal part of a seasonal ARIMA model. The full seasonal ARIMA model can be described as $(p, d, q)(P, D, Q)_m$, where the lowercase letters denote the non-seasonal part while the uppercase letters denote the seasonal part of the model, and m denotes the number of observations in a season. In traffic, there is clear daily and weekly seasonal pattern that can be exploited with this type of model; we can cite [31], [32] as applications of seasonal ARIMA in traffic prediction.

Other extensions to the ARIMA model also exist in the literature. The ARIMAX model (ARIMA model with eXogenous variables) allows ARIMA to model time series using exogenous variables, such as incorporating data from neighbouring links to generate predictions [33]. Similarly, the ARIMA model can also be extended to model the evolution of multiple endogenous time series simultaneously using vector autoregression and space-time ARIMA, which can capture the correlations among multiple roads [34], [35]. The fitting procedure of these models is very tedious and time-consuming, which makes them unsuitable for large-scale prediction. Therefore, in this thesis, we only experiment with the ARIMA and seasonal ARIMA models described above.

2.5.2 Exponential Smoothing

Exponential smoothing is another class of classical time series forecasting methods. In exponential smoothing methods, each observation is modelled as a weighted average of past observations, with the weights decaying exponentially as observations get older. The simple exponential smoothing method consists of two components, a forecasting

equation and a smoothing equation that models the level of the time series. This is shown below in Equations (2.11) and (2.12), where γ denotes the smoothing factor while l and ϵ respectively represent the level and the error terms.

- Forecasting equation:

$$\hat{x_i}^{(t+1)} = l_i^{(t)} \tag{2.11}$$

- Smoothing equation:

$$\begin{aligned} l_i^{(t)} &= \gamma x_i^{(t)} + (1-\gamma) l_i^{(t-1)} \\ &= l_i^{(t-1)} + \gamma (x_i^{(t)} - l_i^{(t-1)}) \\ &= l_i^{(t-1)} + \gamma \epsilon_i^{(t)} \end{aligned} \tag{2.12}$$

We can also extend this method to capture changes in trend and seasonality in the time series by adding additional component equations, which were first described by Holt [36] and Winters [37]. The smoothing equation for the level also describes the observation error (2.12); therefore, this type of model is also known as the ETS (error, trend, seasonal) model in the literature.

The convention is to use three letters to describe the error, trend, and seasonal components of an ETS model. The error component can be either additive (A) or multiplicative (M). Meanwhile, the trend component can be none (N) or additive (A). Lastly, the seasonal component can be none (N), additive, (A), or multiplicative (M).

2.5.3 Facebook Prophet

Prophet is a forecasting algorithm developed by Facebook Open Source [38] that models time series data with seasonality. Instead of explicitly modelling temporal dependence like the time series methods discussed earlier, Prophet uses a curve-fitting procedure on the data which allows for variable time steps and missing data. Inside the model, the Prophet algorithm decomposes the time series into 3 components plus an error term. The first component is the non-periodic trend, which is modelled using growth models with changepoints that can be automatically detected or user-specified. The second component is the seasonality, which is modelled by a Fourier series with user-specified or predefined periods such as weekly or yearly. The last component is non-periodic events or holidays, which includes holidays such as Thanksgiving or Labour day that falls on a different date every year. Prophet has a predefined list of holidays in different countries, which can also be augmented by the user. This process is illustrated in Figure 2.4.

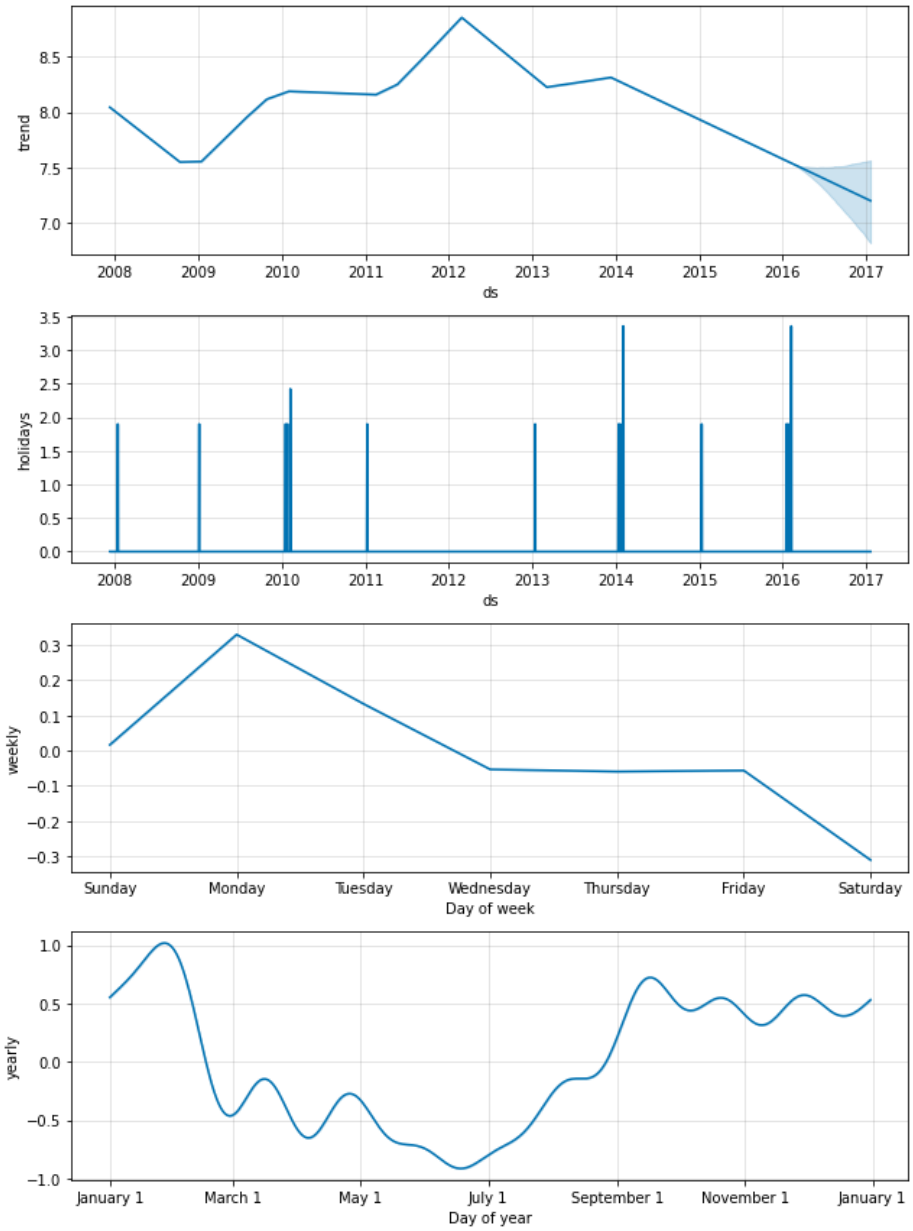


Figure 2.4: A time series decomposed using Facebook Prophet

In this example, the time series is decomposed into trend, holidays, as well as the weekly and yearly seasonal components. Image adapted from [38]

2.6 Classical Machine Learning

Classical machine learning models predict future observations by performing regression analysis on the past observation. Traffic propagation within a brief period is limited in space since the traffic state at a given link is independent of recent traffic states of faraway links. Consequently, we can predict short-term traffic at a given link by performing regression analysis on the recent observations within its neighbourhood. In contrast with the time series models, regression analysis assumes that each observation is independent of one another. In this thesis, we implement the methods described in this section using the scikit-learn library in Python [39].

2.6.1 Linear Regression

In the simplest case of linear regression, we can compute future state values as a linear combination of multiple explanatory features, such as past observations of neighbouring links. Variations of linear regression have been listed in the traffic prediction literature as baseline models to compare performance with other models [40]. Using the notations defined in Section 2.2, a linear regression model for traffic prediction can be defined as follows:

$$\mathbf{x} = \Big\Vert_{j \in \mathcal{N}(i)} \Big\Vert_{k=1}^{p} \mathbf{x}_j^{(t-k)} \tag{2.13}$$

$$\hat{\mathbf{x}}_i^{(t)} = \mathbf{w}^\top \mathbf{x} + \mathbf{b} \tag{2.14}$$

where $\mathbf{x}$ denotes the input features to the regression model, $\|$ denotes the vector concatenation operation, p denotes the number of past time steps to consider, while $\mathbf{w}$ and $\mathbf{b}$ are parameters of the model. In other words, the prediction for link i at time t is a linear combination of the features in the past p observations of the links in the neighbourhood of i plus a bias term. The objective is the find the best set of model parameters $\mathbf{w}$ and $\mathbf{b}$ that minimizes the sum of the squares of prediction error over T training observations. This objective is defined as:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{t=1}^{T} \left\| \mathbf{x}_i^{(t)} - \left(\mathbf{w}^\top \mathbf{x} + \mathbf{b} \right) \right\|_2 + \lambda R(\mathbf{w}) \tag{2.15}$$

The final term, $\lambda R(\mathbf{w})$, is a regularizer over the model parameters that prevents overfitting. The regularizer contains a regularization strength parameter $\lambda > 0$ and regularization function $R(\mathbf{w})$. There are two common regularization functions, L_1 and L_2, which are defined as follows:

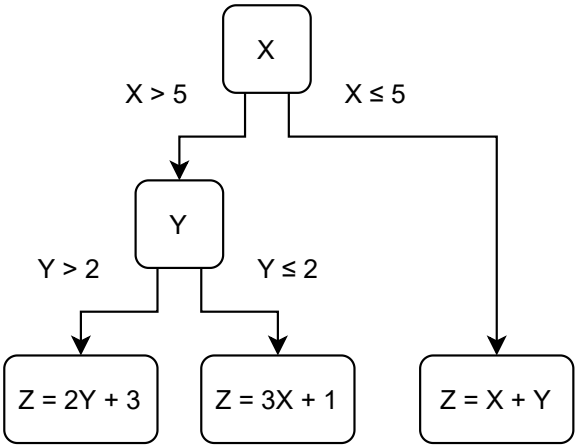


Figure 2.5: A simple regression tree

In this example, X and Y are the features while Z is the target variable. In practice, regression trees can also support categorical and quantitative features as well as multivariate outputs.

- L_1 regularization: $R(\mathbf{w}) = \|\mathbf{w}\|_1$, also known as lasso regularization
- L_2 regularization: $R(\mathbf{w}) = \|\mathbf{w}\|_2$, also known as ridge regression

2.6.2 Ensemble Regression Tree

Linear regression assumes a linear relationship between the input features and the target variable, which may not be true for traffic prediction. Therefore, we also explore the use of decision tree learning, a simple yet very powerful non-linear regression method.

Decision trees with regression targets are also known as regression trees. A regression tree splits the input samples recursively into a tree-like decision diagram until it reaches the desired number of depth or leaf nodes. Each internal node of the tree contains a rule that splits the samples according to the value of some features and passes each split to the corresponding children node. Each leaf node of the tree contains a simple model that describes only the samples within its split. During prediction, we traverse the tree based on the features until we reach a leaf node, then the model within the node can generate the predicted target. With a large number of nodes, this approach can approximate complex functions with relatively simple models; however, a large number of nodes can also lead to overfitting. Therefore, it is common to construct multiple regression trees via ensemble learning to mitigate overfitting. An example of a simple regression tree is shown below in Figure 2.5. Although regression trees can split the samples according to more general inequalities that contain multiple features, the implementation in this thesis selects a single feature to perform each split.

There are two common techniques of ensemble regression trees: bagging (bootstrap aggregating) and boosting. In bagging, each regression tree in the ensemble is constructed only using a subsample of training data (with replacement) and predictions are generated by averaging the predicted values from all trees. Random forest [41] is a typical example of bagged regression trees. In contrast, boosting [42] builds the regression trees sequentially and each subsequent tree emphasizes samples that previous trees fail to predict accurately. Similar to linear regression, the input to an ensemble regression tree model for traffic prediction is the recent observations of nearby links shown in Equation (2.13). The training methodologies of these models are specified in the respective papers [41], [42] and we omit the detailed mathematical formulation in this discussion due to their complexity. Overall, ensemble learning creates a robust regression model and its prowess in traffic prediction has been cited in [7], [40].

Other classical machine learning methods also exist in the traffic prediction literature, such as k-nearest neighbour [43]–[45] and support vector regression [46], [47]. However, their presence in the recent literature is minimal since they are largely superseded by artificial neural networks, which we introduce in the next section. Therefore, while it is important to mention the existence of these works, they do not relate significantly to our contributions and are omitted in this thesis.

2.7 Deep Neural Networks

Since the beginning of the 21st century, the rise of artificial neural networks and deep learning gave researchers a new tool for traffic prediction. This section discusses the various deep learning methods and their application in traffic prediction. Similar to the time series and classical machine learning methods discussed previously, these methods represent traffic dynamics implicitly. Deep neural networks are universal function approximators that can approximate any continuous function in Euclidean space [48]. Consequently, we can use deep neural networks to capture the complex relationship of traffic properties evolving over space and time. In this thesis, we implement all deep neural networks using PyTorch [49] with the MSE loss function defined in Equation (2.8).

2.7.1 Multilayer Perceptron

Multilayer perceptron (MLP) is the first and simplest type of artificial neural network, consisting of one input layer, a series of one or more hidden layers, and one

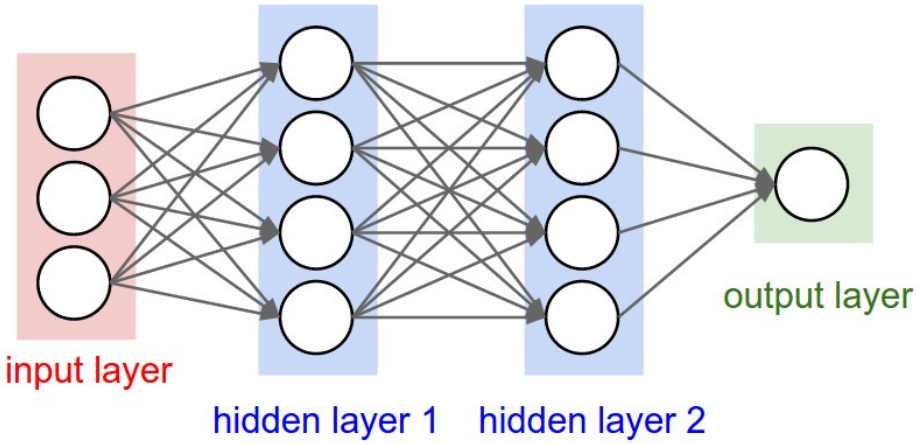


Figure 2.6: A multilayer perceptron with two hidden layers

In this example, the dimension of the input is 3, the dimension of the hidden layers is 4, and dimension of the output is 1. Each layer is fully connected to the previous layer. Image adapted from [50]

output layer. The input layer accepts the input vector $\mathbf{x}$ and the information travels through the layers sequentially, ending at the output layer. After the input layer, each subsequent layer receives information from the previous layer and applies a linear transformation along with a non-linear activation function. The size of the transformed vector after each layer is also known as its dimension. Formally, in an MLP with $L-1$ hidden layers, we can define the outputs $\mathbf{h}$ of each layer recursively as follows:

$$\begin{aligned} \mathbf{h}^{(0)} &= \mathbf{x} \\ \mathbf{h}^{(l)} &= \sigma\left(\mathbf{W}^{(l)\top}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right) \quad 1 \leq l \leq L \end{aligned} \tag{2.16}$$

where σ denotes the activation function while $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ respectively denote the model weights and bias of layer l. The final vector at the output layer, $\mathbf{h}^{(L)}$, is the model prediction. During training, we provide the model with labelled data and iteratively optimize the model parameters via gradient descent and backpropagation to minimize the specified loss function. Figure 2.6 is an example of a multilayer perceptron.

In the context of traffic prediction, the input and output features can be the same as the regression analysis methods introduced in Section 2.6, which corresponds to a model capable of predicting for a single link. Alternatively, the model can also be configured to predict for the entire road network by using an input feature that consists of recent observation across the entire network. We examined both configurations in our experiments, which are discussed in Chapter 3. In our work, we

experimented with three common activation functions: rectified linear unit (ReLU), sigmoid, and hyperbolic tangent (tanh), which are defined below.

- Rectified linear unit:
$$\text{ReLU}(z) = \max(z, 0) \tag{2.17}$$
- Sigmoid:
$$S(z) = \frac{1}{1 + e^{-z}} \tag{2.18}$$
- Hyperbolic tangent:
$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.19}$$

In the traffic prediction literature, this type of artificial neural network appeared in the late 1990s and early adopters of this model include [51]–[54]. Over time, training neural networks became more efficient due to advances in training techniques [55], [56] as well as increased computation power. This allowed for neural networks with increasingly many layers and prompted the development of more complex neural network architectures discussed in the later sections.

2.7.2 Recurrent Neural networks

Recurrent neural networks (RNNs) are the predominant deep learning model for analyzing sequential data, consisting of repeated cells that form a temporal sequence. In contrast to an MLP where each layer contains distinct model weights, the model parameters in an RNN are shared across time steps. Moreover, an RNN accepts input data sequentially at the corresponding cells of each time step. Consequently, by varying the number of repeated cells, this architecture can process sequential data of different lengths. There are two common RNN cell architectures: long short-term memory (LSTM) [57] and gated recurrent unit (GRU) [58], which are defined below in Equations (2.20) and (2.21).

- Long short-term memory:

$$\begin{aligned}
\mathbf{k}_1^{(t)} &= S\left(\mathbf{W}_1\mathbf{x}^{(t)} + \mathbf{U}_1\mathbf{h}^{(t-1)} + \mathbf{b}_1\right) \\
\mathbf{k}_2^{(t)} &= S\left(\mathbf{W}_2\mathbf{x}^{(t)} + \mathbf{U}_2\mathbf{h}^{(t-1)} + \mathbf{b}_2\right) \\
\mathbf{k}_3^{(t)} &= \tanh\left(\mathbf{W}_3\mathbf{x}^{(t)} + \mathbf{U}_3\mathbf{h}^{(t-1)} + \mathbf{b}_3\right) \\
\mathbf{k}_4^{(t)} &= S\left(\mathbf{W}_4\mathbf{x}^{(t)} + +\mathbf{U}_4\mathbf{h}^{(t-1)} + \mathbf{b}_4\right) \\
\mathbf{c}^{(t)} &= \mathbf{k}_1^{(t)} * \mathbf{c}^{(t-1)} + \mathbf{k}_2^{(t)} * \mathbf{k}_3^{(t)} \\
\mathbf{h}^{(t)} &= \mathbf{k}_4^{(t)} * \tanh\left(\mathbf{c}^{(t)}\right) \\
\mathbf{o}^{(t)} &= \sigma\left(\mathbf{W}_o\mathbf{h}^{(t)} + \mathbf{b}_o\right)
\end{aligned} \tag{2.20}$$

- Gated recurrent unit:

$$\begin{aligned}
\mathbf{k}_1^{(t)} &= S\left(\mathbf{W}_1\mathbf{x}^{(t)} + \mathbf{U}_1\mathbf{h}^{(t-1)} + \mathbf{b}_1\right) \\
\mathbf{k}_2^{(t)} &= S\left(\mathbf{W}_2\mathbf{x}^{(t)} + \mathbf{U}_2\mathbf{h}^{(t-1)} + \mathbf{b}_2\right) \\
\mathbf{k}_3^{(t)} &= \tanh\left(\mathbf{W}_3\mathbf{x}^{(t)} + \mathbf{U}_3\left(\mathbf{k}_1^{(t)} * \mathbf{h}^{(t-1)}\right) + \mathbf{b}_3\right) \\
\mathbf{h}^{(t)} &= \left(1 - \mathbf{k}_2^{(t)}\right) * \mathbf{h}^{(t-1)} + \mathbf{k}_2^{(t)} * \mathbf{k}_3^{(t)} \\
\mathbf{o}^{(t)} &= \sigma\left(\mathbf{W}_o\mathbf{h}^{(t)} + \mathbf{b}_o\right)
\end{aligned} \tag{2.21}$$

In this formulation, $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are the model parameters $\mathbf{k}$ are the intermediate outputs of the model, $\mathbf{h}^{(t)}$ and $\mathbf{o}^{(t)}$ are respectively the hidden state and the output generated by the model at time t, while $*$ denotes the Hadamard product. Furthermore, the LSTM model additional contains a cell state denoted by $\mathbf{c}$. Similar to MLPs, we can train RNNs using the backpropagation through time algorithm. It should be noted that the above equations are the standard configurations of the LSTM and the GRU presented in their respective original papers, but variations also exist in the literature. Figure 2.7 illustrates how RNN cells can be configured in different applications.

With a recurrent neural network, we can predict future traffic states by feeding the past observations sequentially into the model. Similar to the MLP model, an RNN can be configured to predict a single link or the entire road network by modifying the input features. Furthermore, an RNN can generate a sequence of predictions by feeding the predicted values back into the network as the input of the next time step. Recurrent neural networks have been actively used in traffic prediction research to capture the temporal dynamics of evolving traffic since the work of [59] in 2015. Later improvements to this model include stacking multiple RNNs [60] and incorporating the spatial structure of the road network in the RNN [61].

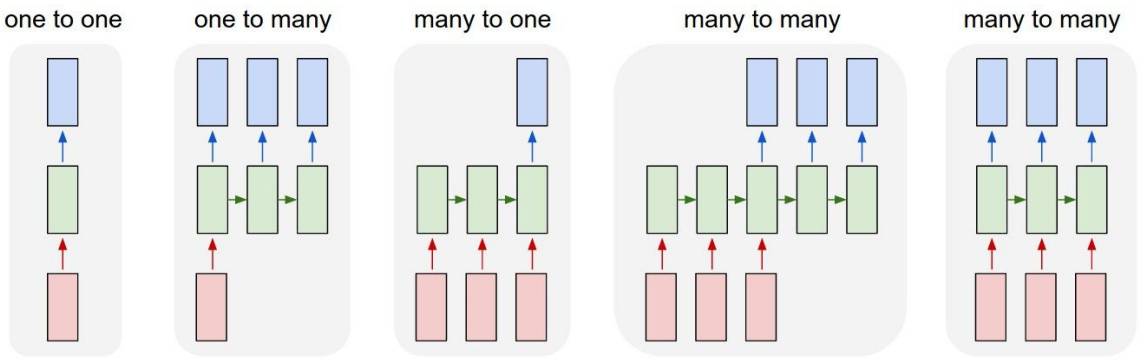


Figure 2.7: The various configurations of recurrent neural networks

In this example, the red rectangles represent the input vectors, the green rectangles represent the RNN cells, and the blue rectangles represent the RNN outputs. Image adapted from [50]

2.7.3 Graph Neural Networks

Graph neural networks (GNNs) are artificial neural networks designed to process graph-structured data, primarily through applying the graph convolution operation. Graph convolution extends the notion of the convolution operation, which is commonly applied to analyzing visual imagery with a grid-like structure, to an operation that can be applied to graphs with arbitrary structures. Therefore, a GNN is capable of extracting information using the spatial correlations between nodes in a graph and lends itself well to capturing the complex patterns needed for short-term traffic prediction. We can cite [5] as the first application of graph neural network in short-term traffic prediction, and there have been many subsequent works that expand upon this idea.

The first GNN framework is founded in the convolution theorem and the field of graph signal processing [62]. The convolution theorem states that the convolution of two signals in the point-wise product of their Fourier transforms and graph signal processing defines the Fourier transform of a graph based on the normalized Laplacian matrix $\mathbf{L}$. The definition of the normalized Laplacian matrix is shown in Equation (2.22) below:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{2.22}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{A}$ is the adjacency matrix of the graph, and $\mathbf{D}$ is the diagonal degree matrix defined by $\mathbf{D}_{ii} = \Sigma_j \mathbf{A}_{ij}$. Subsequently, we can perform eigendecomposition on the Laplacian matrix to obtain a matrix of eigenvectors $\mathbf{U}$ and a diagonal matrix of the corresponding eigenvalues $\mathbf{\Lambda}$. The Fourier transform and the inverse Fourier transform of a graph signal $\mathbf{x} \in \mathbb{R}^{|\mathcal{V}|\times 1}$ is then respectively defined in Equations (2.23) and (2.24).

$$\tilde{\mathbf{x}} = \mathbf{U}^\top \mathbf{x} \tag{2.23}$$

$$\mathbf{x} = \mathbf{U}\tilde{\mathbf{x}} \tag{2.24}$$

Finally, we can perform convolution on a graph signal by applying the Fourier transform, multiplying by the convolutional kernel, then applying the inverse Fourier transform [63]. This is shown in Equation (2.25), where $\mathbf{y} \in \mathbb{R}^{|\mathcal{V}|\times 1}$ is the output of the convolution and diagonal matrix $\mathbf{\Theta}$ is the convolutional kernel.

$$\mathbf{y} = \mathbf{U}\mathbf{\Theta}\mathbf{U}^\top\mathbf{x} \tag{2.25}$$

The main problem with this approach is that the operation is not localized in space because the kernel $\mathbf{\Theta}$ is applied in the spectral domain after the Fourier transform. In other words, the output of a node contains information from the entire graph, which is undesirable for applications such as traffic where a node only exerts influence on a part of the graph. The work of [64] demonstrates that we can achieve a localized filter by restricting $\mathbf{\Theta}$ to be a polynomial of $\mathbf{\Lambda}$ and that the Chebyshev expansion is a suitable polynomial kernel approximation. The Chebyshev polynomial of order k, $T_k(x)$, can be written as the recurrence relation in Equation (2.26).

$$T_k(x) = 2x\,T_{k-1}(x) - T_{k-2}(x); \qquad T_0(x) = 1,\ T_1(x) = x \tag{2.26}$$

Using this method, $\mathbf{\Theta}$ is approximated as a Chebyshev expansion of order $K-1$, where K is the maximum radius of convolution [65]. This is shown in Equation (2.27), where $\theta \in \mathbb{R}^{K+1}$ is a vector of Chebyshev coefficients and λ_{max} is the maximum eigenvalue of $\mathbf{L}$.

$$\begin{aligned} \mathbf{\Theta} &= \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{\Lambda}}) \\ \tilde{\mathbf{\Lambda}} &= \frac{2\mathbf{\Lambda}}{\lambda_{max}} - \mathbf{I} \end{aligned} \tag{2.27}$$

The full convolution operation can then be written as:

$$\begin{aligned} \mathbf{y} &= \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{L}})\mathbf{x} \\ \tilde{\mathbf{L}} &= \frac{2\mathbf{L}}{\lambda_{max}} - \mathbf{I} \end{aligned} \tag{2.28}$$

Lastly, for the case where $K = 1$, [66] proposes that λ_{max} can be approximated as 2 to achieve a linear model with respect to $\mathbf{L}$. Furthermore, the two parameters, θ_0 and θ_1, can be combined into a single parameter θ by setting $\theta = \theta_0 = -\theta_1$. This is

shown in Equation (2.29) below.

$$\begin{aligned}\mathbf{y} &= \theta_0\mathbf{x} + \theta_1(\mathbf{L}-\mathbf{I})\mathbf{x} \\ &= \theta\left(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{x}\end{aligned} \tag{2.29}$$

So far, this section only discussed the case of single dimensional inputs and outputs at each node. Using the linear model, we can generalize Equation (2.29) to multidimensional inputs and outputs as well as introduce a bias term and activation function. This is shown in Equation (2.30) below, where the model parameters are defined by a matrix $\mathbf{W}$ with dimensions equal to the number of input features and the number of output features.

$$\mathbf{Y} = \sigma\left(\left(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{X}\mathbf{W} + \mathbf{b}\right) \tag{2.30}$$

With localized convolution, every node has a defined neighbourhood around it based on the structure of the graph. Using the formulation introduced by [67], the operations of a GNN can be divided into two phases, a message-passing phase and a readout phase. For a given node i, the message passing phase aggregates the information within its neighbourhood $\mathcal{N}(i)$, then the subsequent readout phase applies the model parameters to create the output. In traffic, the predominant form of GNN is the aforementioned linear model and its extensions. In this framework, the message passing phase can be interpreted as taking a weighted sum of the inputs within the neighbourhood, where the coefficients a are determined by graph properties such as the adjacency matrix or the Laplacian matrix. Meanwhile, the readout phase is defined as a linear transformation with an activation function. Using the notation introduced earlier in Section 2.2, this type of GNN can be formulated as follows:

$$\mathbf{h}_i = \sigma\left(\sum_{j\in\mathcal{N}(i)} a_{ij}\mathbf{W}\mathbf{x}_j + \mathbf{b}\right) \tag{2.31}$$

where $\mathbf{h}_i$ is the output representation for node i, $\mathbf{x}_j$ is the graph input for node j, a_{ij} is the influence from node j to node i that is defined by the aggregation matrix, $\mathbf{W}$ and $\mathbf{b}$ are respectively the weight and bias terms of the model that transforms the input to hidden dimension, and $\sigma(\cdot)$ denotes the activation function. In the case of the linear model, Equations (2.30) and (2.31) are equivalent as $\mathcal{N}(i)$ corresponds to node i and its one-hop neighbours while the coefficients a correspond to entries in the matrix $\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ in Equation (2.30).

In the more recent graph attention networks [68], a_{in} are instead produced by

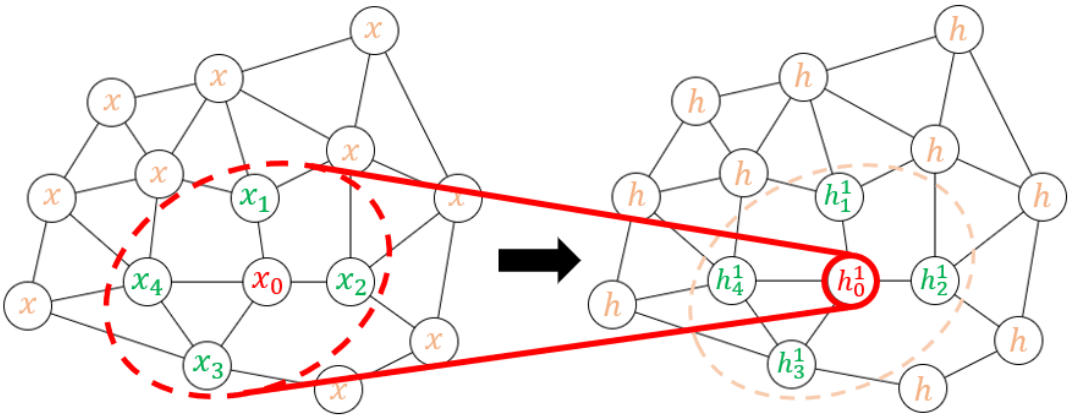


Figure 2.8: A graph neural network

In this example, the target node is x_0 and its neighbouring nodes are x_1 to x_4. The neighbourhood here is defined as the nodes within one hop from the target, which is represented by the dotted circle.

an additional module that learns the relationship between every pair of nodes to assign weights for the aggregation. This can be achieved with a variety of attention mechanisms that exist in the literature such as the works of [69], [70], the mechanism cited in [68] is as follows:

$$
\begin{aligned}
a_{ij} &= \frac{\exp(\text{LeakyReLU}\left(\boldsymbol{\alpha}^\top[\mathbf{W}\mathbf{x}_i \parallel \mathbf{W}\mathbf{x}_j]\right))}{\sum_{k\in\mathcal{N}(i)} \exp(\text{LeakyReLU}\left(\boldsymbol{\alpha}^\top[\mathbf{W}\mathbf{x}_i \parallel \mathbf{W}\mathbf{x}_k]\right))} \\
\mathbf{h}_i &= \sigma\left(\sum_{j\in\mathcal{N}(i)} a_{ij}\mathbf{W}\mathbf{x}_j + \mathbf{b}\right)
\end{aligned}
\tag{2.32}
$$

where

$$
\text{LeakyReLU}(z) = \begin{cases} z & z > 0 \\ 0.2\,z & z \leq 0 \end{cases} \tag{2.33}
$$

In the above formulation, $\boldsymbol{\alpha}$ defines the linear layer that calculates the attention value between two nodes, and $\parallel$ denotes the concatenation operator. It is important to note that there is only one set of model weight $\mathbf{W}$ and bias $\mathbf{b}$ that is applied to all nodes in both formulations.

A short-term traffic prediction model can use GNNs to capture the spatial correlations between different nodes of a road network; however, the model also needs to account for the changing dynamic of traffic through time. This is commonly achieved

in the literature through the use of RNNs with GRU cells as exemplified by [5], [7], [8], where the matrix multiplications in the GRU is replaced with a GNN operation. For example, (2.21) can be modified as follows:

$$\begin{aligned}
\mathbf{k}_{1,i}^{(t)} &= S\left(\sum_{j\in\mathcal{N}(i)} a_{ij}\left(\mathbf{W}_1\mathbf{x}_j^{(t)} + \mathbf{U}_1\mathbf{h}_j^{(t-1)}\right) + \mathbf{b}_1\right) \\
\mathbf{k}_{2,i}^{(t)} &= S\left(\sum_{j\in\mathcal{N}(i)} a_{ij}\left(\mathbf{W}_2\mathbf{x}_j^{(t)} + \mathbf{U}_2\mathbf{h}_j^{(t-1)}\right) + \mathbf{b}_2\right) \\
\mathbf{k}_{3,i}^{(t)} &= \tanh\left(\sum_{j\in\mathcal{N}(i)} a_{ij}\left(\mathbf{W}_3\mathbf{x}_j^{(t)} + \mathbf{U}_3\left(\mathbf{k}_{1,i}^{(t)} * \mathbf{h}_j^{(t-1)}\right)\right) + \mathbf{b}_3\right) \\
\mathbf{h}_i^{(t)} &= \left(1 - \mathbf{k}_{2,i}^{(t)}\right) * \mathbf{h}_i^{(t-1)} + \mathbf{k}_{2,i}^{(t)} * \mathbf{k}_{3,i}^{(t)} \\
\mathbf{o}_i^{(t)} &= \sigma\left(\mathbf{W}_o\mathbf{h}_i^{(t)} + \mathbf{b}_o\right)
\end{aligned} \tag{2.34}$$

There are also works in the literature that apply other types of deep neural networks in traffic prediction, the most notable being convolutional neural networks [40], [71], [72]. However, the attempts at employing convolutional neural networks require adjustments that dismantle the topological structure of the road network [72]. As a result, this type of framework cannot generate predictions on individual links and we omit them in this thesis.

Chapter 3

Evaluation of Classical and Current Machine Learning Methods

3.1 Introduction

In the previous chapter, we introduced the traffic prediction problem and a wide selection of solutions that currently exist in the literature. We found that recent works mainly focus on innovating new models, especially in the graph neural network area. Although the older methods are sometimes listed as baseline models for comparison, there lacks a comprehensive evaluation of the different methods under different settings in the literature. This type of evaluation is important because it allows us to draw insight into the constitution of an accurate predictive method and incorporate proven techniques when developing new models.

In this chapter, we address this deficiency by assessing the performance of a large selection of models in both urban and highway scenarios. We evaluate the models using data from traffic simulations which avoids the problem of missing data due to sensor issues common in real-world data sets. In addition, the simulation environment allows us to easily adjust demands and road conditions to create a variety of benchmarks for the analysis.

3.2 Methodology

3.2.1 Data Source

We procured two sets of data to represent the highway and urban traffic conditions; both sets are created using the Aimsun Next [9] simulation software. For the highway data, we simulated Queen Elizabeth Way, a highway in Ontario with 56 links on the

Figure 3.1: Map of the highway region chosen for this study.

eastbound direction of travel. For the urban data, simulated a 167-link region in downtown Toronto. The maps of the highway and urban regions are respectively shown in Figure 3.1 and Figure 3.2.

The travel demand was collected from survey data in 2016 [73], then the simulation model was calibrated using measurements from the loop detectors installed along the roads [74]. We built the simulation model using morning peak-hour travel demands, and each simulation is for the four-hour period between 6:00 and 10:00 AM. The speed (distance traveled per unit time) and flow (number of vehicles per unit time) for every link are extracted from the simulations in 1-minute intervals. Speed and flow are selected because they are the most common form of data in the real-world measured using loop detectors and GPS. To augment the data, the simulation is run 50 times and each simulation uses the original travel demands multiplied by a random scalar factor between 0.5 and 1.5.

3.2.2 Evaluation

In this study, we assessed the models based on the predicted speed values in 5 different settings. For both the urban and highway dataset, we used a prediction horizon of 5 minutes as a base case. Missing data due to faulty sensors on the road is very common, thus we additionally included a scenario to evaluate the model performance under the presence of missing data in the highway setting. We chose 5% as the probability of missing data, evaluated using the same 5-minute prediction horizon. Meanwhile, we

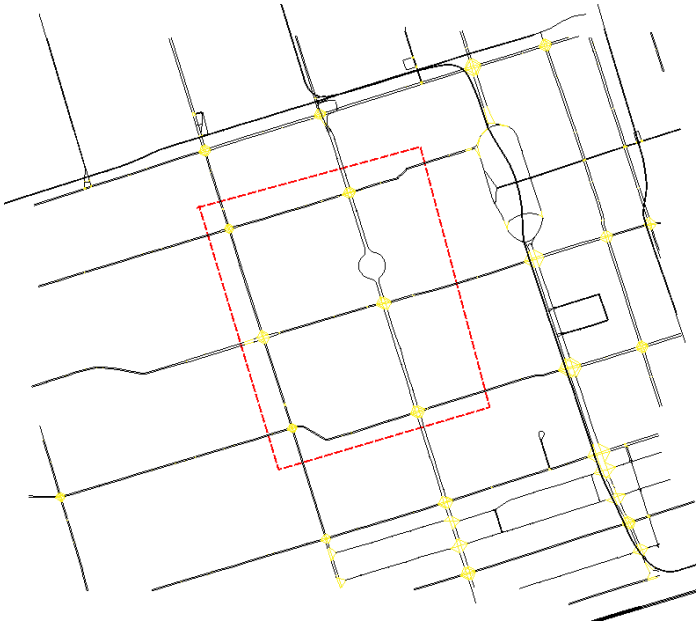


Figure 3.2: Map of the urban region chosen for this study.

included a scenario with a 1-minute prediction horizon for the urban dataset, which is useful for adaptive traffic signal control.

The last scenario evaluates the generalizability of each model. We generated an artificial dataset from the highway simulation model by blocking a lane in an arbitrarily chosen section while using the same travel demands, which simulates traffic conditions during a traffic accident. We train the models on the regular highway dataset, then apply them to the unseen artificial dataset to evaluate the performance metrics. Across all scenarios, we report the MAE, MAPE, RMSE of the prediction models, their definitions are outlined in Section 2.3.

3.2.3 Model Selection

For this study, we selected 8 candidate models from the methods discussed in the previous chapter.

- **ARIMA**: as described in Section 2.5.1
- **Linear regression**: as described in Section 2.6.1
- **Random forest regression**: as described in Section 2.6.2
- **Local MLP**: as described in Section 2.7.1 with same input and output features as the regression models
- **Global MLP**: as described in Section 2.7.1 with network-wide input and predictions

- **Local RNN**: as described in Section 2.7.2 with same input and output features as the regression models
- **Global RNN**: as described in Section 2.7.2 with network-wide input and predictions
- **GRNN** [7]: This model is similar to what is described in Equation (2.34) in Section 2.7.3 with the following modifications. Here β is a hyperparameter of the model.

$$
\begin{aligned}
a_{ij} &= \begin{cases} 1 & i = j \\ \beta & (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \\
\mathbf{k}_{1,i}^{(t)} &= S\left(\mathbf{W}_1 \mathbf{x}_i^{(t)} + \sum_{j \in \mathcal{N}(i)} a_{ij} \mathbf{U}_1 \mathbf{h}_j^{(t-1)} + \mathbf{b}_1\right) \\
\mathbf{k}_{2,i}^{(t)} &= S\left(\mathbf{W}_2 \mathbf{x}_i^{(t)} + \sum_{j \in \mathcal{N}(i)} a_{ij} \mathbf{U}_2 \mathbf{h}_j^{(t-1)} + \mathbf{b}_2\right) \\
\mathbf{k}_{3,i}^{(t)} &= \tanh\left(\mathbf{W}_3 \mathbf{x}_i^{(t)} + \sum_{j \in \mathcal{N}(i)} a_{ij} \mathbf{U}_3 \left(\mathbf{k}_{1,i}^{(t)} * \mathbf{h}_j^{(t-1)}\right) + \mathbf{b}_3\right) \\
\mathbf{h}_i^{(t)} &= \left(1 - \mathbf{k}_{2,i}^{(t)}\right) * \mathbf{h}_i^{(t-1)} + \mathbf{k}_{2,i}^{(t)} * \mathbf{k}_{3,i}^{(t)} \\
\mathbf{o}_i^{(t)} &= \sigma\left(\mathbf{W}_o \mathbf{h}_i^{(t)} + \mathbf{b}_o\right)
\end{aligned}
\tag{3.1}
$$

In addition, we include two simple models to serve as baselines: a model that always predicts the arithmetic mean of past observed values (Constant), and a model that uses the most recent observation as the prediction (Previous Interval). The hyperparameters summarized in **Table 3.1** are tuned by coordinate descent. In coordinate descent, the hyperparameters are tuned one at a time. We repeatedly this process until an additional pass of coordinate descent over all hyperparameters no longer yields a different set of hyperparameters.

3.3 Results

The results of the models under the five scenarios are shown in the tables below. The mean and the confidence interval for each metric are calculated using 5-fold cross-validation. In 5-fold cross-validation, the full dataset is split into 5 partitions with

Table 3.1: Hyperparameter selection

Name	Range	Applicable Models
Neighborhood size (number of steps to target link)	{2, 3, 4, 5, 6}	Linear, Random Forest, Local MLP
Number of past time steps	{2, 3, ..., 8}	ARIMA, Linear, Random Forest, Local MLP, Global MLP, Local RNN, Global RNN, GRNN
Number of hidden features (global models)	{64, 128, 256, 512}	Global MLP, Global RNN
Number of hidden features (other models)	{32, 64, 128}	Local MLP, Local RNN, GRNN
Batch size	{32, 64, 128}	Local MLP, Global MLP, Local RNN, Global RNN, GRNN
L2 regularization strength (linear regression)	{0, 0.01, 0.1, ..., 100}	Linear
L2 regularization strength (neural network)	{1e-5, 1e-4, 1e-3}	Local MLP, Global MLP, Local RNN, Global RNN, GRNN
Number of training epochs	{50, 100, ..., 2000}	Local MLP, Global MLP, Local RNN, Global RNN, GRNN
Number of differences	{0, 1, 2}	ARIMA
Number of trees	{100, 150, 200}	Random Forest
Tree depth	{5, 10, 15, 20}	Random Forest
Dropout	{0, 0.2, 0.5}	Local MLP, Global MLP
β (defined in **Equation** (3.1))	{0.2, 0.4, 0.6}	Local RNN, GRNN

equal sizes. In each iteration, one partition is reserved for testing while the remaining partitions are used for training; the process is repeated until each of the 5 partitions has been used as the test set. We did not perform 5-fold cross-validation on the scenario with unseen artificial data since the test dataset is completely separated from training and validation datasets.

Except for the unseen test set scenario, the baseline models of constant and previous interval performed worse than every other model. However, the simple linear regression baseline proves itself to be comparable in performance with the deep learning architectures. The only scenario where the linear regression model had poor performance is in the presence of missing data. In general, the ARIMA model performed worse than the linear regression and random forest regression models. Overall, the random forest regression model performed the best in all scenarios and metrics.

Across all experiments, the local versions of MLP and RNN performed better than their global counterparts. In addition, the performance of the global RNN model is unstable as evidenced by the large spread of 95% confidence intervals. The local RNN performed better than its MLP counterpart for highway data, while their performances are relatively similar on the urban data. However, the MLP models can generalize better to unseen artificial test data. Finally, the GRNN model performed competitively across all experiment scenarios, but it is consistently outperformed by

Table 3.2: Results for 5-minute prediction horizon on the highway dataset

		Constant	Previous Interval	Linear	ARIMA	Random Forest	Local MLP	Global MLP	Local RNN	Global RNN	GRNN
MAE (km/h)	Mean	33.16	4.56	3.29	4.41	**2.76**	3.41	3.80	2.86	7.11	3.30
	95% CI	32.34	4.24	3.02	4.12	2.52	3.27	3.41	2.66	4.28	2.77
		33.98	4.87	3.55	4.70	3.01	3.55	4.19	3.06	9.98	3.82
MAPE (%)	Mean	109.55	12.32	9.72	11.96	**8.02**	9.95	11.54	8.36	21.25	9.89
	95% CI	88.06	10.32	8.17	10.05	6.74	8.40	9.95	7.05	12.21	7.46
		131.04	14.32	11.28	13.87	9.31	11.50	13.13	9.67	30.29	12.33
RMSE (km/h)	Mean	35.46	9.34	5.12	9.13	**4.74**	5.37	5.76	4.83	11.81	5.37
	95% CI	34.72	8.72	4.62	8.53	4.15	5.01	5.05	4.33	7.56	4.44
		36.19	9.95	5.61	9.73	5.32	5.72	6.48	5.33	16.06	6.30

Table 3.3: Results for 5-minute prediction horizon on the highway dataset with 5% missing data

		Constant	Previous Interval	Linear	ARIMA	Random Forest	Local MLP	Global MLP	Local RNN	Global RNN	GRNN
MAE (km/h)	Mean	33.16	4.57	4.76	4.43	**2.85**	4.49	4.64	3.04	5.82	3.61
	95% CI	32.34	4.26	4.42	4.14	2.59	4.19	4.06	2.78	3.79	3.37
		33.98	4.88	5.10	4.72	3.11	4.78	5.21	3.30	7.85	3.84
MAPE (%)	Mean	109.55	12.36	12.55	12.00	**8.26**	12.29	13.39	8.88	16.78	10.91
	95% CI	88.06	10.37	10.89	10.09	6.96	10.45	11.61	7.35	11.73	8.77
		131.04	14.35	14.21	13.92	9.56	14.13	15.16	10.41	21.82	13.06
RMSE (km/h)	Mean	35.46	9.37	6.92	9.17	**4.84**	6.52	7.04	5.08	10.08	5.83
	95% CI	34.72	8.77	6.28	8.58	4.31	5.99	6.08	4.59	6.84	5.37
		36.19	9.97	7.56	9.77	5.38	7.05	8.01	5.58	13.32	6.29

Table 3.4: Results for 5-minute prediction horizon on the downtown dataset

		Constant	Previous Interval	Linear	ARIMA	Random Forest	Local MLP	Global MLP	Local RNN	Global RNN	GRNN
MAE (km/h)	Mean	6.68	6.53	4.30	4.87	**3.94**	4.42	4.43	4.18	4.27	4.35
	95% CI	6.64	6.52	4.28	4.83	3.91	4.38	4.20	4.11	4.15	4.32
		6.72	6.55	4.32	4.90	3.97	4.46	4.67	4.25	4.40	4.39
MAPE (%)	Mean	46.71	36.65	24.21	30.98	**22.37**	24.00	24.40	22.98	24.41	25.32
	95% CI	44.32	36.21	23.72	30.46	21.93	23.31	23.41	22.49	22.31	24.22
		49.11	37.09	24.69	31.50	22.80	24.68	25.40	23.46	26.52	26.42
RMSE (km/h)	Mean	10.04	10.88	6.61	7.73	**6.30**	6.92	6.87	6.77	6.82	7.01
	95% CI	9.91	10.85	6.56	7.69	6.26	6.81	6.47	6.56	6.63	6.94
		10.16	10.92	6.65	7.76	6.34	7.02	7.28	6.98	7.02	7.09

Table 3.5: Results for 1-minute prediction horizon on the downtown dataset

		Constant	Previous Interval	Linear	ARIMA	Random Forest	Local MLP	Global MLP	Local RNN	Global RNN	GRNN
MAE (km/h)	Mean	6.68	6.78	3.79	4.53	**3.40**	3.86	4.13	3.89	4.37	4.40
	95% CI	6.64	6.74	3.77	4.49	3.38	3.78	4.01	3.83	4.14	4.32
		6.72	6.81	3.80	4.57	3.42	3.93	4.24	3.94	4.59	4.47
MAPE (%)	Mean	46.71	38.05	21.48	28.19	**19.14**	20.10	22.38	21.18	26.60	26.58
	95% CI	44.32	37.74	21.12	27.70	18.90	19.26	22.02	20.82	24.33	25.27
		49.11	38.36	21.83	28.68	19.38	20.93	22.73	21.53	28.87	27.89
RMSE (km/h)	Mean	10.04	11.31	5.80	7.13	**5.47**	6.02	6.33	6.24	6.78	6.94
	95% CI	9.91	11.24	5.77	7.07	5.43	5.84	6.14	6.03	6.39	6.86
		10.16	11.38	5.83	7.18	5.50	6.20	6.52	6.45	7.16	7.02

Table 3.6: Results for 5-minute prediction horizon on the unseen artificial test set

	MAE (km/h)	MAPE (%)	RMSE (km/h)
Constant	31.01	96.08	33.75
Previous Interval	6.15	16.54	11.00
Linear	5.64	16.71	8.31
ARIMA	6.10	16.92	10.87
Random Forest	**5.28**	**15.58**	**8.36**
Local MLP	6.46	18.78	9.51
Global MLP	12.38	39.34	15.99
Local RNN	6.69	27.89	12.10
Global RNN	22.43	72.74	32.21
GRNN	5.70	15.61	8.99

the random forest model and the local RNN model.

Curiously, on the urban dataset, a few models can predict 5 minutes in advance better than 1 minute. Similarly, on the highway data, the global RNN model performed better under the presence of missing data.

3.4 Discussion

The ARIMA model is commonly used as a baseline model for comparison in the traffic prediction literature. However, our experiment results show that ARIMA performs only marginally better than simply predicting the last observation. In addition, since there exists known influences among nearby roads, the univariate nature of ARIMA hampers its performance. This is especially evident as the prediction horizon increases, where evolving traffic patterns cause nearby information to become more important than local history. Overall, a good traffic prediction model should contain a more regional perspective rather than decoupling the roads from one another.

In the case of feedforward and recurrent neural networks, the local models perform better than the network-wide models. We speculate that this is because traffic dynamics are local, and using separate models for each location allows the parameters to be learned more easily. The global model has access to the recent history of the entire road network; therefore, the model can extract the relevant features to create accurate predictions for every location. Meanwhile, each local model only has access to the recent history of a smaller nearby region. This restriction forces the local models to use features with known influences on the output, which guides parameter learning and creates more accurate predictions. Nevertheless, it is uncertain whether global models can achieve the same performance given access to more data.

Graph neural networks are compact due to the shared parameters across different links, and our experiment results show that this creates a competitive model. How-

ever, removing parameter sharing can further improve performance as evidenced by the better performance of the local RNN models. We believe this can be attributed to the different local dynamics of traffic, and having a custom updating operation for each location is essential in generating accurate predictions. Nevertheless, the reduced number of parameters of the GRNN helps prevent overfitting and generalize to unseen data, where it outperformed all other deep learning models in our experiment. One interesting research direction would be designing a graph neural network with a dynamic updating operation that accounts for the changing traffic dynamics between different roads and different times of the day. This would allow the overall model to remain relatively compact while being expressive enough to represent all traffic patterns.

Across all scenarios, the random forest model generates the most accurate predictions. The ensemble nature of random forest makes it very robust to overfitting, and the model also generalizes well to unseen data (**Table 3.6**). However, it is not a compact model, requiring more memory and storage compared to other models. It is difficult to quantify the exact number of parameters in a random forest model (since tree structures are semi-parametric), but the random forest for the 5-minute highway prediction scenario contains over 83 million combined nodes in the regression trees. In contrast, the local MLP and the GRNN model contain 516,512 and 50,817 parameters respectively. Finally, since the regression features are different for every location, separate models need to be trained for every link, which slows model training and reduces scalability to larger networks.

Evaluation metrics such as MAE, MAPE, and RMSE demonstrate model performance in general; however, they cannot illustrate the finer trends of model performance. Therefore, we examined the model predictions at each location and identified two trends. On the highway dataset, the error on a link is negatively correlated with its length (Figure 3.3). We speculate that this is because shorter links are usually present at interchanges and exits, which influence traffic on the highway and increase the difficulty of prediction. Similarly, on the urban dataset, the error on a link is positively correlated to its number of upstream and downstream connections (Figure 3.4), which signifies that links close to intersections have higher prediction errors. We speculate that this is because traffic signals have a greater influence on the conditions of these links rather than traffic propagation. Therefore, it would be interesting to include information on traffic signals into predictive models and assess their performance. Lastly, we examined where prediction errors occur, which is illustrated in Figure 3.5. Evidently, high prediction error coincide with areas where there is a large change in speed and the prediction error is much lower when the speed is relatively

constant. However, prediction accuracy is arguably more important in unstable traffic conditions than relatively constant traffic conditions, which suggests that mean error metrics such as MAE, MAPE, and RMSE may not be the best evaluation metrics for this problem. Future research should consider reassessing the evaluation metrics and focusing on predicting changes in the traffic condition.

3.5 Conclusion

In this chapter, we compared classical and modern methods of forecasting short-term traffic in both highway and urban settings using data from traffic simulation software. The methods included in this evaluation are ARIMA, random forest regression, multilayer perceptrons, recurrent neural networks, and graph recurrent neural networks. Using a traffic model of the Greater Toronto Area, we generated a highway and an urban benchmark dataset. Finally, we created scenarios such as different prediction horizons and the presence of missing data to achieve a comprehensive comparison.

In our experiments, we demonstrated that deep learning methods of traffic prediction, including graph neural networks, are effective for traffic prediction. Graph neural networks with shared parameters were very compact, resistant to overfitting, and performed well in all of our experiments. However, ensemble methods such as random forest regression can generate more accurate predictions at the cost of being a larger model, which may require considerably more storage capacity for large transportation networks. In addition, we highlight the importance of incorporating nearby observations in achieving more accurate predictions. However, it is also essential to limit the model inputs to only nearby observations since too many inputs obstruct model learning and deteriorate system performance. Moreover, we also demonstrated the need for individual parameters for each prediction location due to the uniqueness of local traffic patterns. Finally, we identified that existing models cannot predict well in the presence of intersections. Overall, these facets should be carefully considered when developing new models for traffic prediction.

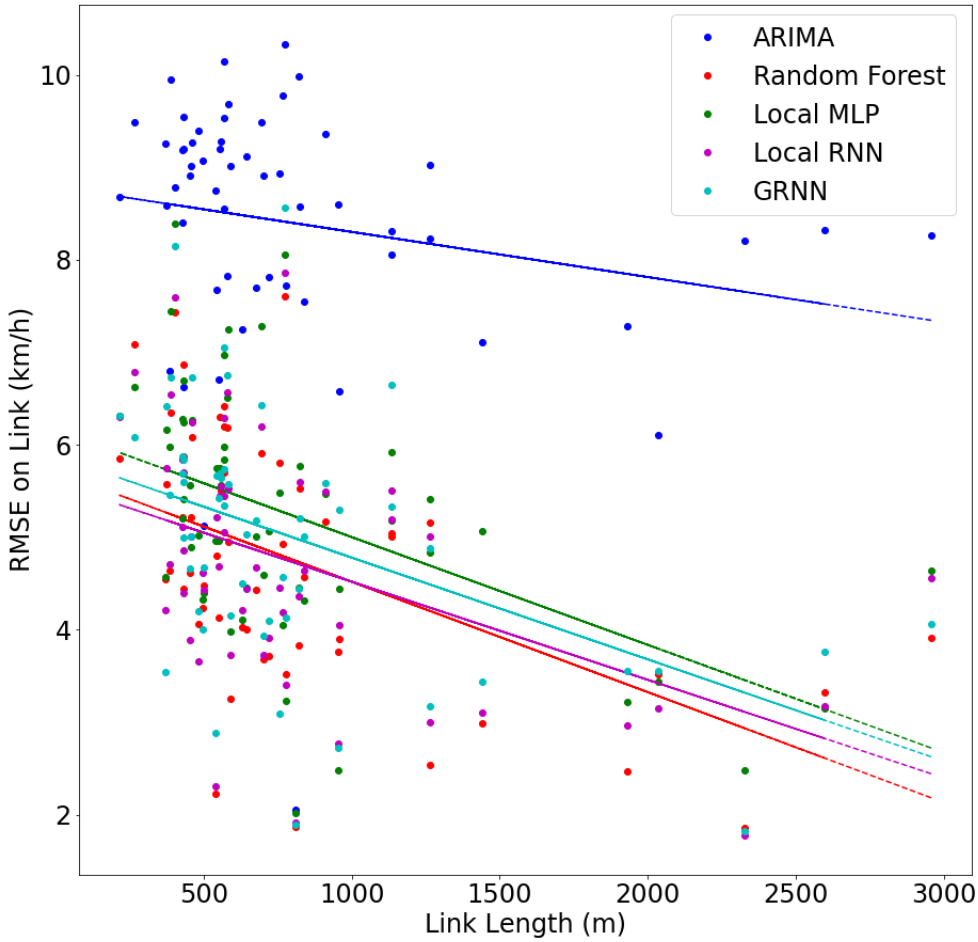


Figure 3.3: The RMSE of each link and the line of best fit for each model on the highway dataset.

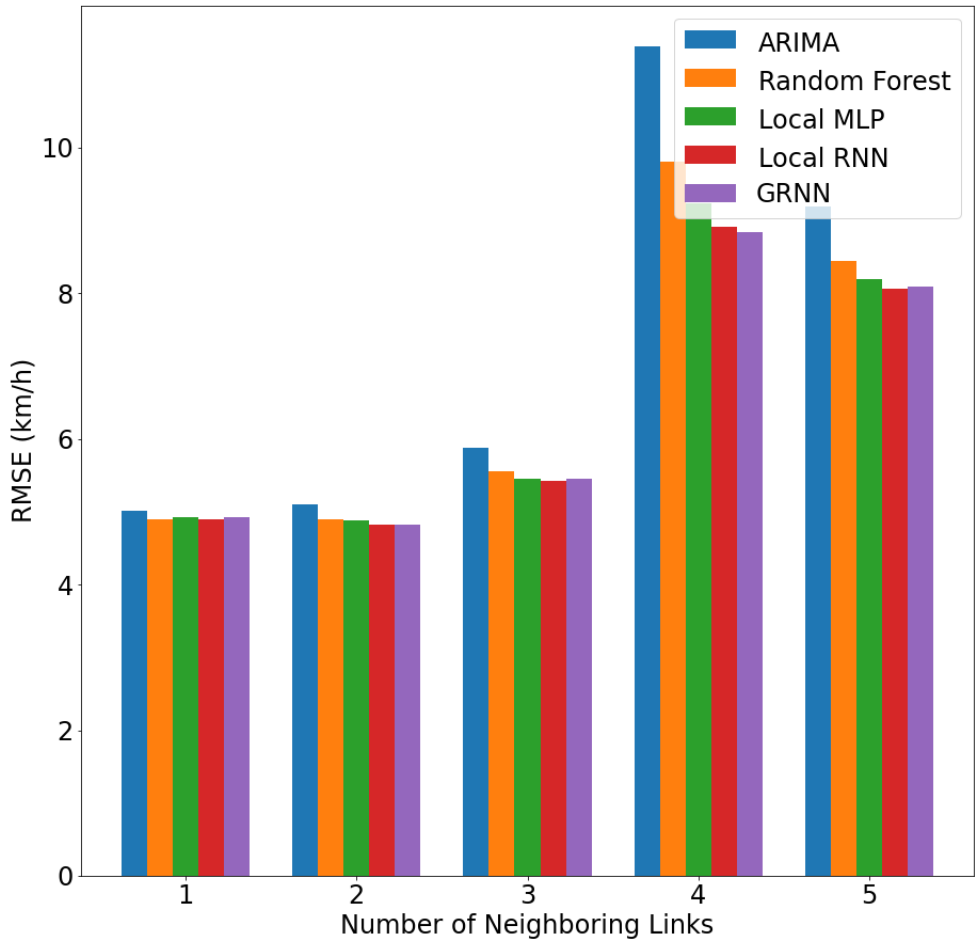
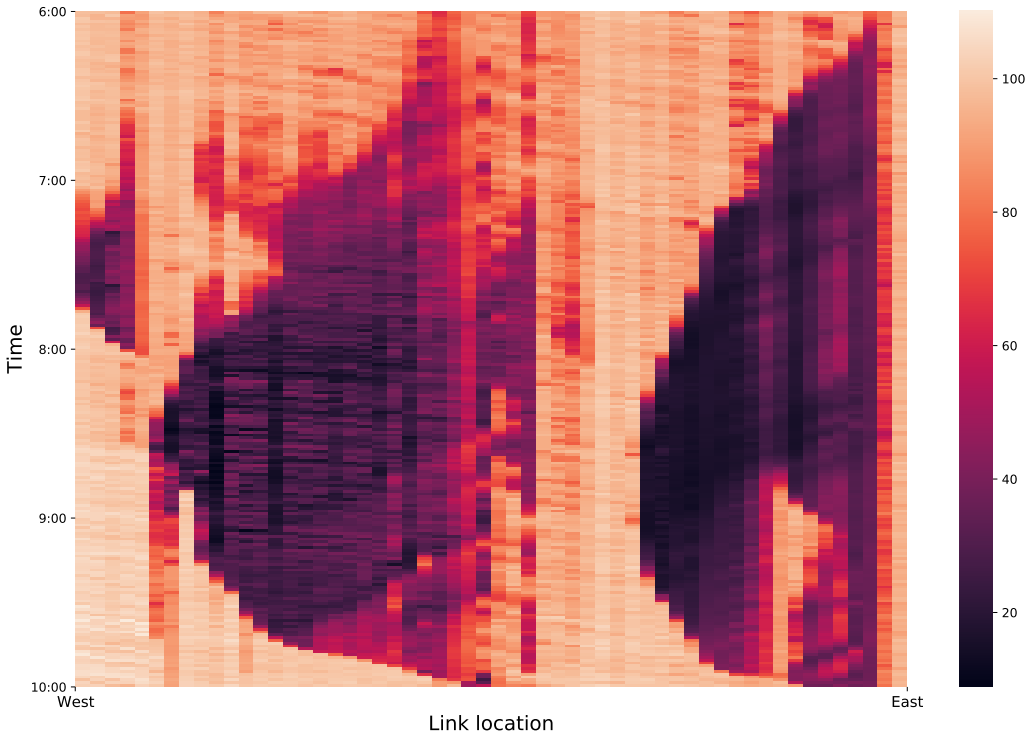
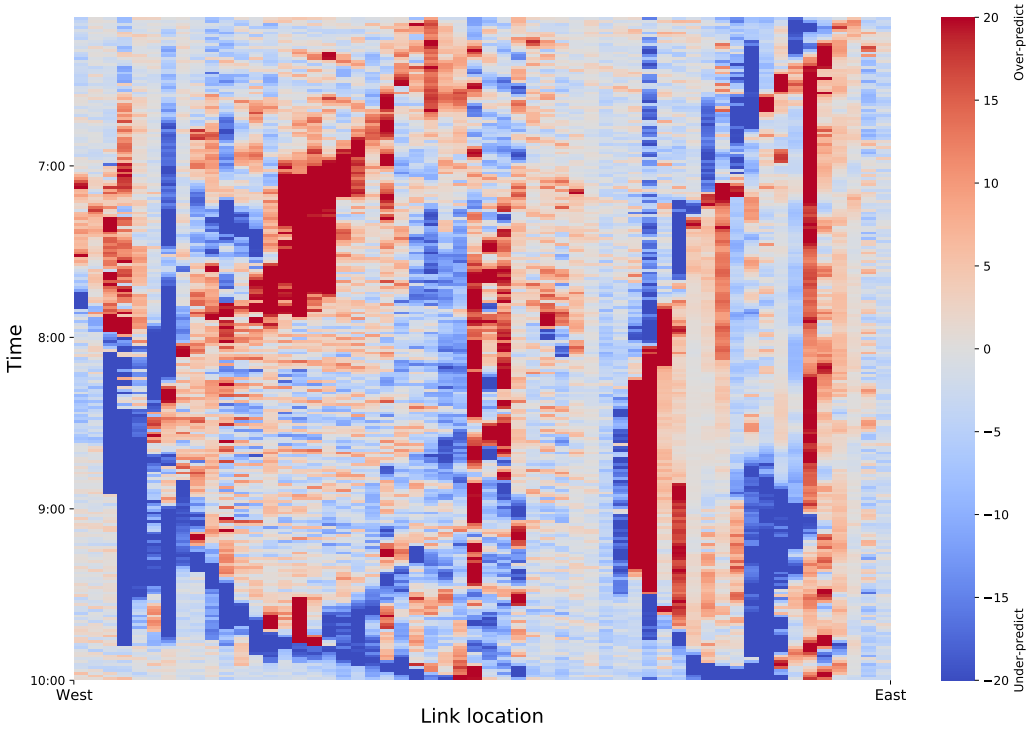


Figure 3.4: The RMSE of each model categorized by the number of neighboring links on the urban dataset.



(a) The true speed of the entire corridor



(b) The prediction error of the random forest model

Figure 3.5: The error of the random forest model for 1 simulation of the highway data

Chapter 4

Comparative Evaluation of Graph Neural Network Variants

4.1 Introduction

The previous chapter evaluated GNNs and other deep neural networks against the classical machine learning methods. However, the literature on GNNs has exploded over the past 4 years and the GRNN model may not be an accurate representation of all GNNs. Therefore, we felt the need to expand the selection of GNNs to create a more objective analysis.

In addition, each work in the literature claims to improve upon its predecessors in the evaluation metrics; however, the improvements are very marginal. In a recent example from 2020 [8], the authors compare their proposed model against the works of [5], [6], which are both published in 2017. The proposed model demonstrates a 0.9% reduction in MAPE for flow prediction, which corresponds to 0.8 vehicles per 5 minutes in MAE [8]. Similar results can also be found in [75]–[77]. This serves as additional motivation for us to investigate these innovations and determine if there is still room for advancement.

This chapter first describes the graph convolution perspective and then develops a taxonomy of GNN short-term traffic prediction models based on their components. Afterwards, we explore different variations on these components and eventually arrive at a variant that is similar to a traditional recurrent neural network. Finally, we also revisit the regression view of short-term traffic prediction using random forest regression, which we found to be a powerful prediction method in the previous chapter.

4.2 Methodology

4.2.1 Components of GNN Traffic Prediction Models

As mentioned in Section 2.7.3, a short-term traffic prediction model can use a GNN to capture the spatial correlations between different nodes of a road network while using an RNN to capture the changing dynamic of traffic through time. Typically, the two components are combined by replacing the matrix multiplications in the GRU with a GNN operation shown in Equation (2.34). Equation (2.34) is written again in this section as (4.1) to improve readability.

$$
\begin{aligned}
\mathbf{k}_{1,i}^{(t)} &= S\left(\sum_{j\in\mathcal{N}(i)} a_{ij}\left(\mathbf{W}_1\mathbf{x}_j^{(t)} + \mathbf{U}_1\mathbf{h}_j^{(t-1)}\right) + \mathbf{b}_1\right) \\
\mathbf{k}_{2,i}^{(t)} &= S\left(\sum_{j\in\mathcal{N}(i)} a_{ij}\left(\mathbf{W}_2\mathbf{x}_j^{(t)} + \mathbf{U}_2\mathbf{h}_j^{(t-1)}\right) + \mathbf{b}_2\right) \\
\mathbf{k}_{3,i}^{(t)} &= \tanh\left(\sum_{j\in\mathcal{N}(i)} a_{ij}\left(\mathbf{W}_3\mathbf{x}_j^{(t)} + \mathbf{U}_3\left(\mathbf{k}_{1,i}^{(t)} * \mathbf{h}_j^{(t-1)}\right)\right) + \mathbf{b}_3\right) \\
\mathbf{h}_i^{(t)} &= \left(1 - \mathbf{k}_{2,i}^{(t)}\right) * \mathbf{h}_i^{(t-1)} + \mathbf{k}_{2,i}^{(t)} * \mathbf{k}_{3,i}^{(t)} \\
\mathbf{o}_i^{(t)} &= \sigma\left(\mathbf{W}_o\mathbf{h}_i^{(t)} + \mathbf{b}_o\right)
\end{aligned}
\tag{4.1}
$$

With this formulation, we can categorize GNN short-term traffic prediction models by their 3 components. The first 2 components are the operation concerning the input $\mathbf{x}_i^{(t)}$ and the last hidden state $\mathbf{h}_i^{(t-1)}$, which can be a standard matrix multiplication or a GNN variant. The third component is the model weights $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$, which can be either shared among nodes or independent. We explore variations on these components in the next section. It is important to note that some works use other mechanisms to capture the temporal dynamics of traffic; however, this investigation is focused on RNN-based models due to their prevalence.

4.2.2 Variations on GNN Components

We begin with the GRNN model [7] introduced in the previous chapter, which uses a standard matrix multiplication for input, convolution for the last hidden state, and shared model weights among nodes. This formulation transforms the first equation

of (2.21) to the following:

$$\mathbf{k}_{1,i}^{(t)} = S\left(\mathbf{W}_1\mathbf{x}_i^{(t)} + \sum_{j\in\mathcal{N}(i)} a_{ij}\mathbf{U}_1\mathbf{h}_j^{(t-1)} + \mathbf{b}_z\right) \tag{4.2}$$

As in (4.1), the remaining equations in (2.21) can be transformed likewise and are omitted for brevity in this section.

We then experiment with changing the graph convolution operation to graph attention shown in (2.32) and examine the different combinations of applying the attention operation to input and hidden states. For this investigation, we call this type of model the graph attention gated recurrent unit (GA-GRU) as it combines the concepts of graph attention networks and gated recurrent units. Equation 4.3 is one configuration where attention is only applied to the last hidden state; i.e., GA-GRU (hidden).

$$\begin{aligned} a_{ij} &= \frac{\exp(\text{LeakyReLU}\left(\boldsymbol{\alpha}^\top[\mathbf{W}\mathbf{x}_i \parallel \mathbf{W}\mathbf{x}_j]\right))}{\sum_{k\in\mathcal{N}(i)} \exp(\text{LeakyReLU}\left(\boldsymbol{\alpha}^\top[\mathbf{W}\mathbf{x}_i \parallel \mathbf{W}\mathbf{x}_k]\right))} \\ \mathbf{k}_{1,i}^{(t)} &= S\left(\mathbf{W}_1\mathbf{x}_i^{(t)} + \sum_{j\in\mathcal{N}(i)} a_{ij}\mathbf{U}_1\mathbf{h}_j^{(t-1)} + \mathbf{b}_1\right) \end{aligned} \tag{4.3}$$

Afterwards, we explore the removal of shared weights among different nodes by designating a unique set of model weights $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ for each node. In order to remove all weight sharing across nodes, we also replaced the shared attention layer in (4.3) with a trainable attention matrix. For this investigation, we call this type of model the attentional graph recurrent neural network; i.e., AGRNN (hidden). In this framework, (4.3) is transformed to the following:

$$\mathbf{k}_{1,i}^{(t)} = S\left(\mathbf{W}_{1,i}\mathbf{x}_i^{(t)} + \sum_{j\in\mathcal{N}(i)} a_{ij}\mathbf{U}_{1,i}\mathbf{h}_j^{(t-1)} + \mathbf{b}_{1,i}\right) \tag{4.4}$$

In contrast with (4.3), the subscript i in all weights and biases signifies that each node contains its own model parameters, and the a in this framework are learnable weights.

We also highlight the input-only attention variant of the AGRNN; i.e., AGRNN

Table 4.1: The GNN variants explored
Categorized along the 3 component discussed in Section 4.2.1

Model	Input	Hidden	Weights
GRNN [7]	Multiplication	Convolution	Shared
GA-GRU (input)	Attention	Multiplication	Shared
GA-GRU (hidden)	Multiplication	Attention	Shared
GA-GRU (both)	Attention	Attention	Shared
AGRNN (input)	Attention	Multiplication	Independent
AGRNN (hidden)	Multiplication	Attention	Independent
AGRNN (both)	Attention	Attention	Independent
AGCRN [8]	Attention	Attention	Factorized

(input), the equation is defined below:

$$\mathbf{k}_{1,i}^{(t)} = S\left(\sum_{j \in \mathcal{N}(i)} a_{ij}\mathbf{W}_{1,i}\mathbf{x}_j^{(t)} + \mathbf{U}_{1,i}\mathbf{h}_i^{(t-1)} + \mathbf{b}_{1,i}\right) \tag{4.5}$$

Since there is no convolution or attention applied to the previous hidden states, the hidden states of different nodes do not influence one another. Combined with the independent model weights, this structure is akin to traditional recurrent neural networks with added location context and models of different nodes can be trained independently.

Lastly, we also include the AGCRN model in this comparison. The AGCRN model is an example in the literature where the independent model weights are factorized according to the spatial correlations among nodes [8]. In this framework, there is a shared pool of weights and each node has a latent state representation that is used to access this pool of weights differently, which results in a structure that shares weights between nodes yet applies a distinct model to every node. In addition, the latent state representation can capture nodes with similar behaviour and apply similar model weights. Compared to node-independent weights, this structure also has the benefit of reducing the number of parameters and the factorization of model weights allows them to be optimized more easily [78].

Table 4.1 summarizes the different variations of GNN discussed in this section.

4.2.3 Datasets

For this study, we employ the same datasets from our evaluation in the previous chapter. In addition, to facilitate comparison with the recent literature on GNNs, we additionally include the California datasets described below.

- **Toronto datasets**: the same data that was used in the previous chapter, de-

scribed in Section 3.2.1

- **California datasets**: The PeMS04 and PeMS08 datasets are collected from the Caltrans Performance Measurement System (PeMS) of districts in California. The interval is 5 minutes per time step, which corresponds to 288 data points per day. The adjacency matrix is defined according to road distance and connectivity. We follow the evaluation procedure established by other papers [8], [79], which uses the last 12 observations to predict the next 12 time steps, i.e., use the past hour of traffic data to predict that of the next hour.

4.2.4 Model Selection

We evaluated our models against a selection of different methods that are listed below, including other graph neural networks as well as time series analysis methods. Additionally, similar to Chapter 3, we tuned the hyperparameters summarized in Table 4.2 using the same coordinate descent procedure outlined in Chapter 3.

- **Historical Average**: A time series model that predicts the average of observations from the same time of day in previous weeks. It is not applicable to simulated datasets since the simulation model simulates only one day and has no long-term time series.
- **ARIMA**: as described in Section 2.5.1
- **Random Forest**: as described in Section 2.6.2
- **GCN** [66]: This model is a Graph Convolutional Network with 1 hidden layer. The output layer is connected with a fully-connected layer to predict traffic states.

Table 4.2: Hyperparameter selection

Name	Range	Applicable Models
Neighborhood size (number of steps to target node)	{1, 2, 3, 4, 5, 6}	All models except ARIMA
Number of historical time steps	{2, 3, ..., 9}	All models on simulation datasets
Number of historical time steps	12	All models on PeMS datasets
Number of hidden features	{32, 64, 128}	Neural network models
Batch size	{32, 64, 128}	Neural network models
Learning rate	{1e-5, 1e-4, 1e-3}	Neural network models
L2 regularization strength	{1e-5, 1e-4, 1e-3}	Neural network models
Number of training epochs	{50, 100, ..., 2000}	Neural network models
Number of differences	{0, 1, 2}	ARIMA
Number of trees	{100, 150, 200}	Random forest
Tree depth	{5, 10, 15, 20}	Random forest

Table 4.3: Performance comparison of traffic speed prediction models for simulated highway dataset

	5-minute horizon			15-minute horizon		
Model	MAE (km/h)	MAPE (%)	RMSE (km/h)	MAE (km/h)	MAPE (%)	RMSE (km/h)
Historical Average	Not applicable			Not applicable		
ARIMA	4.41	11.96	9.13	7.52	20.73	15.47
GCN	3.72	9.05	5.95	5.24	13.08	8.99
GRNN	<u>3.24</u>	9.66	<u>5.31</u>	5.18	14.90	9.00
GA-GRU (input)	4.99	15.30	7.72	13.09	38.09	16.83
GA-GRU (hidden)	3.38	10.30	5.35	5.00	15.31	8.55
GA-GRU (both)	4.08	12.55	6.32	13.48	47.70	16.95
AGRNN (input)	3.28	9.71	5.46	4.90	14.85	8.63
AGRNN (hidden)	3.92	9.14	6.00	4.99	12.13	8.12
AGRNN (both)	3.53	8.69	6.17	<u>3.84</u>	**9.52**	<u>6.86</u>
AGCRN	3.41	**7.70**	7.38	4.28	<u>10.00</u>	8.84
Random forest	**2.77**	<u>8.02</u>	**4.73**	**3.60**	10.31	**6.51**

4.2.5 Evaluation Metrics

In this study, we report the MAE, MAPE, RMSE of the prediction models, their definitions are outlined in Section 2.3. For the Toronto datasets, we used both 5th minute and 15th minute as the prediction horizon and computed error using only the prediction at the horizon $\hat{\mathbf{x}}_i^{(t+H)}$. Meanwhile, on the California datasets, we followed the convention of [8], [79] and computed error using all predictions up to the prediction horizon H; i.e., $\hat{\mathbf{x}}_i^{(t+1)}, \hat{\mathbf{x}}_i^{(t+2)}, ..., \hat{\mathbf{x}}_i^{(t+H)}$.

Although the above metrics conveniently produce numerical values for easy comparison across models, they are unable to represent all model aspects. Therefore, we also measured the complexity of each model for a more well-rounded comparison as shown in Table 4.6.

4.3 Results

We performed the evaluation using a data split of 60% training, 20% validation, and 20% testing for each dataset. We then recorded each metric to produce results shown in Tables 4.3, 4.4, and 4.5 below, where the bolded number is the lowest error and the underlined number is the second lowest error. In addition, we also report the model complexity for the 5-minute prediction horizon on the highway dataset in Table 4.6.

In most experiments, performances improve from that of GRNN, GA-GRUs, to AGRNNs. First, this indicates that using an attention mechanism to learn spatial correlations is better than using a fixed adjacency matrix. Besides, the node-specific convolutional weights can capture distinct traffic patterns in each node and improve accuracy. Moreover, the AGRNN (input) model is competitive with other GNNs

Table 4.4: Performance comparison of traffic speed prediction models for simulated urban dataset

	5-minute horizon			15-minute horizon		
Model	MAE (km/h)	MAPE (%)	RMSE (km/h)	MAE (km/h)	MAPE (%)	RMSE (km/h)
Historical Average	Not applicable			Not applicable		
ARIMA	4.87	30.98	7.73	5.43	35.12	8.58
GCN	4.30	24.33	6.53	4.62	25.21	6.94
GRNN	4.37	24.88	7.06	4.94	29.32	7.78
GA-GRU (input)	4.85	31.06	7.65	5.31	33.45	8.39
GA-GRU (hidden)	4.24	24.04	6.91	4.83	28.48	7.98
GA-GRU (both)	4.62	29.27	7.40	4.96	31.18	7.90
AGRNN (input)	4.11	22.49	6.88	4.31	24.44	7.21
AGRNN (hidden)	4.23	24.55	6.71	4.43	25.77	7.02
AGRNN (both)	4.08	22.35	6.62	4.27	**23.88**	7.02
AGCRN	4.02	23.59	7.02	4.49	28.40	7.78
Random forest	**3.89**	**22.18**	**6.21**	**4.17**	24.37	**6.66**

Table 4.5: Performance comparison of traffic flow prediction models on PeMS04 and PeMS08 dataset

	PeMS04			PeMS08		
Model	MAE (veh)	MAPE (%)	RMSE (veh)	MAE (veh)	MAPE (%)	RMSE (veh)
Historical Average	24.99	16.07	41.84	21.21	13.72	36.73
ARIMA	27.53	20.55	42.44	22.67	14.92	35.08
GCN	23.72	17.92	37.47	21.09	14.42	31.45
GRNN	30.66	25.02	46.06	26.09	21.92	39.07
GA-GRU (input)	32.78	30.24	46.65	27.73	42.20	38.78
GA-GRU (hidden)	24.73	17.17	38.18	19.89	14.01	30.73
GA-GRU (both)	29.35	25.23	42.74	23.53	19.96	34.46
AGRNN (input)	24.01	17.37	37.82	21.04	14.57	31.19
AGRNN (hidden)	22.97	16.32	37.25	20.31	13.48	30.80
AGRNN (both)	23.77	16.54	38.53	22.04	14.33	34.16
AGCRN	**19.86**	**13.06**	**32.57**	**16.08**	**10.40**	**25.55**
Random forest	20.74	13.76	35.03	16.64	10.95	26.95

Table 4.6: Model complexity measured in number of parameters

Model	Complexity
Historical Average	48960
ARIMA	1035
GCN	66587944
AGCRN	150112
GRNN	3393
GA-GRU (input)	1025
GA-GRU (hidden)	1025
GA-GRU (both)	1121
AGRNN (input)	8602510
AGRNN (hidden)	605710
AGRNN (both)	634610
Random forest[a]	171195284

[a] It is difficult to quantify the exact number of parameters in a random forest model (since tree structures are semi-parametric), therefore we instead report the combined number of nodes across all trees. The actual number of parameters is at least several times this value.

according to all error metrics; which signifies that the propagation of hidden states among nodes between consecutive RNN time steps is not essential in achieving accurate prediction. It should be noted that although our experiments keep the other components of the model constant among GRNN, GA-GRUs, and AGRNNs, the findings of this work may not generalize to other architectures, such as multiple graph convolutional layers or different RNN configurations.

4.4 Discussion

The AGRNN with input-only attention is an extreme version of separating GNN weights to create independent models; meanwhile, the factorized weights in AGCRN can be viewed as a trade-off between having completely shared weights of GNNs and completely independent weights. The superior results of the AGCRN model in our experiments suggest that this trade-off approach is worthy of further investigation.

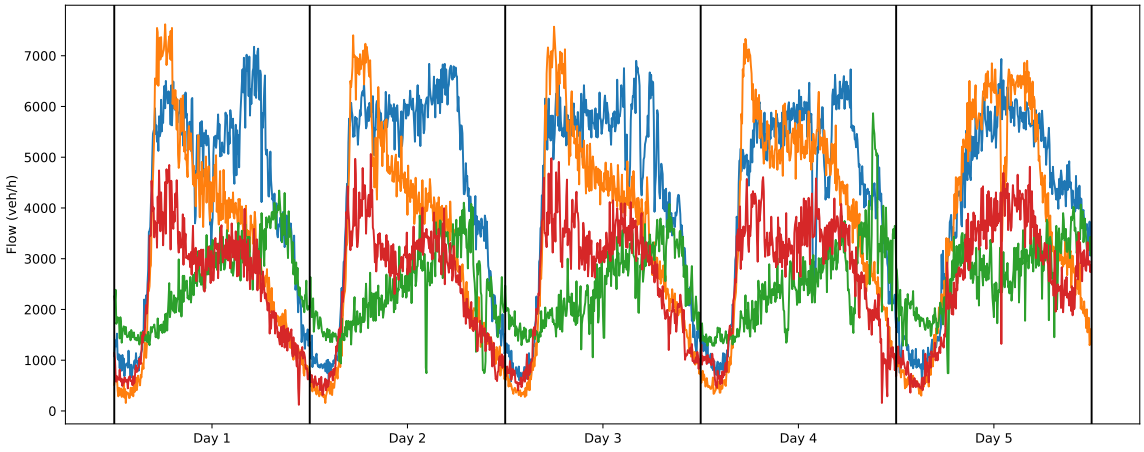
The experiment results show that the AGCRN performs noticeably worse in the simulated datasets relative to other models. We believe this is because real-world data are largely repetitive with very few day-to-day variations. In contrast, the simulated datasets allow for randomized demand and the traffic patterns of different runs are completely different from one another. This trend is illustrated in Figure 4.1. Overall, we believe the variation provides a more objective analysis of the predictive capacity of the models. Although simulation software contains its problems such as potential errors in the network and calibration procedures, its ability to freely adjust demands and create more varied datasets should not be overlooked.

The experiment results also show that the random forest model exhibits the lowest

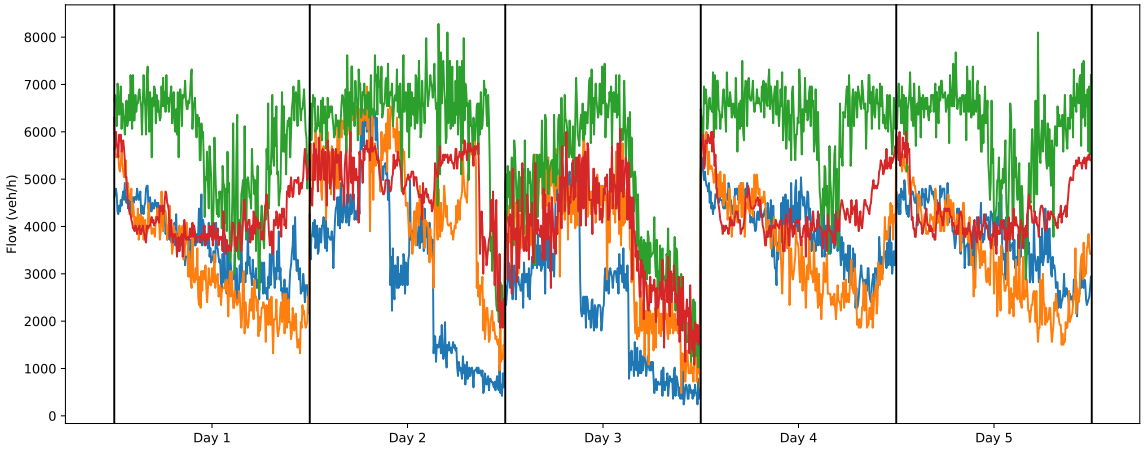
error on the simulated datasets while being narrowly outperformed by the AGCRN model on the California datasets. This supports the hypothesis that short-term traffic prediction can be formed as a regression problem and further highlights that sharing model weights and latent states are inconsequential in attaining model accuracy. Although we do not report the exact training time of each model, except the historical average, all models examined in this chapter displayed similar training time in the order of several hours. However, the random forest model contains by far the largest number of parameters across all experiments, which may be impracticable for deployment in real-world environments due to storage requirements. Overall, the result suggests that while GNNs can be more compact, random forest regression remains competitive and should not be overlooked in short-term traffic prediction.

The errors we report for the GNN models are largely in line with recent publications and it is unlikely that further hyperparameter tuning would yield any significant reduction in prediction error [8]. Furthermore, similar to other studies that use the California PeMS datasets [8], [75], the MAE and RMSE in Table 4.5 are measured in number of vehicles per time step (5 minutes). We see that the best GNNs improve upon the baseline historical average model by around 10 vehicles per time step, which corresponds to 120 vehicles per hour. To put this into perspective, the capacity of a 3-lane highway is in the range of 5000 to 6000 vehicles per hour [80]. Therefore, the best GNNs can predict better than historical average by about 2 percent of the capacity. It is unclear whether this error reduction can result in tangible benefits for downstream applications of traffic prediction.

Although the location-independent AGRNN and random forest models contain more parameters compared to their graph convolutional counterparts, it also presents several major advantages. First of all, this framework is highly flexible and modular to the number of prediction locations and new modules can be added as new sensors are installed. Secondly, this framework allows us to easily identify and remedy the source of error because each location has an independent model that can be updated as new data becomes available. Lastly, since each prediction location contains its own model that can be trained separately, the overall framework scales linearly with respect to the number of prediction locations, which allows this model to be deployed for large cities with relatively low memory consumption. We generated synthetic data with a variable number of locations and trained the models on a system with a GeForce RTX 2070 which has 8 GB of memory. Using the same default model hyperparameters, the GRNN model runs out of memory when the number of locations exceeds 850 while the location-independent models do not face this constraint.



(a) PeMS08 dataset - one working week of data



(b) Simulated highway dataset - first 5 simulation runs

Figure 4.1: Flow comparison between real-world and simulated data
For each dataset, we select 4 links and plot the flow values over 5 days.

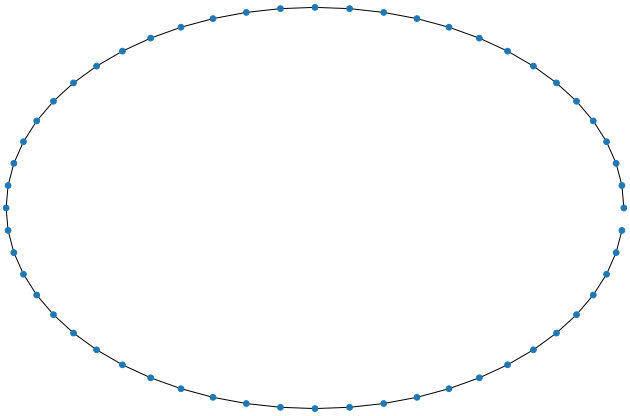
4.4.1 GNN Attention Analysis

In our introduction of GNNs in Section 2.7.3, we mentioned that the attention matrix represents the learned influence among all nodes. During the experiment, we extracted the trained attention matrix in the models to perform a sanity check on their learned influence. In the GNN variations defined in Section 4.2.2, we restrict the influence of node i to its neighbourhood $\mathcal{N}(i)$ based on our understanding that traffic propagation occurs according to the network topology. However, the AGCRN and other GNNs in the literature do not enforce this restriction and allow for network-wide influence for every node. This difference is highlighted in Figure 4.2, where we notice that the attention values learned by the AGCRN model are significantly different from the road topology. This result further indicates that the superior performance of the ACGRN may be attributed to detecting statistical correlation among faraway

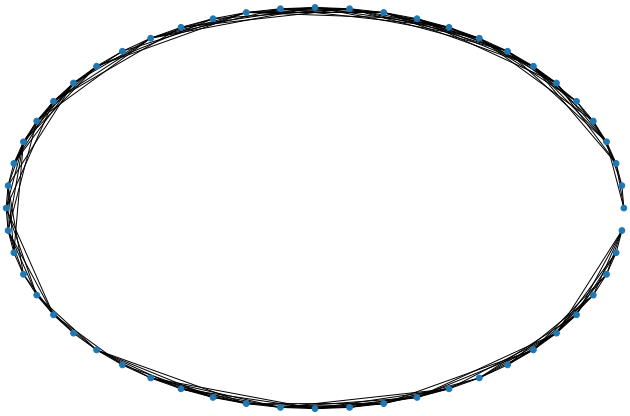
nodes as opposed to its ability to model traffic dynamics. Consequently, this suggests that models such as the AGCRN may fail to predict accurately in the event of traffic incidents since the local traffic patterns would deviate from normal and cannot be adequately explained by long-range statistical correlation.

4.5 Conclusion

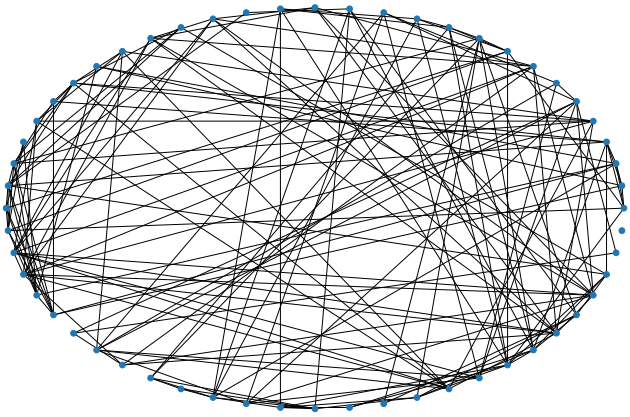
In this chapter, we compared several variants of GNNs against random forest regression using simulated data of two regions in Toronto as well as real-world sensor data from selected California highways. We found that incorporating matrix factorization, attention, and location-specific model weights either individually or collectively into GNNs can result in a better overall performance. Moreover, although random forest regression is a less compact model, it matches or exceeds the performance of all variations of GNNs in our experiments. Finally, we also highlighted two issues with current state-of-the-art GNNs: Parameter sharing across different locations limits their flexibility and scalability. Also, the GNNs are learning the incorrect traffic propagation patterns. Overall, this suggests that the current graph convolutional methods may be in incorrect approach to traffic prediction.



(a) Simulated highway dataset - node connectivity



(b) Simulated highway dataset - AGRNN attention



(c) Simulated highway dataset - AGCRN attention

Figure 4.2: Trained attention matrices of the GNN models

In this illustration, each blue dot represents a node. The locations of the dots are chosen to achieve a compact figure and do not reflect real-world locations. For the attention figures, the width of the line represents the trained attention values between the two nodes.

Chapter 5

Long-Term Traffic Prediction

5.1 Introduction

The previous chapters discussed our contributions regarding short-term traffic prediction, with horizons up to 1 hour. However, transportation agencies may require prediction further into the future to enable long-term planning. Historically, research in this sector is mostly focused on time series models such as seasonal ARIMA [31].

In this chapter, we first revisit the seasonal ARIMA model in long-term traffic prediction by following the analysis of Williams and Hoel [31]. Afterwards, we explore the exponential smoothing and the Facebook Prophet models described in Section 2.5.2 and 2.5.3, respectively. Subsequently, we also examine the potential of regression methods such as random forest and XGBoost in this task. Lastly, we attempt to define the boundary between short-term and long-term prediction by determining the horizon where the knowledge of recent traffic history no longer influences prediction accuracy.

5.2 Methodology

5.2.1 Data Source

Since this study is focused on long-term prediction, the 4-hour long simulated datasets from the previous chapters are insufficient for this analysis. Therefore, for this study, we obtained the traffic speed data of 216 links on the Gardiner Expressway in Toronto over the month of October 2019. The location of the Gardiner Expressway is shown on the map in Figure 5.1 highlighted in red. The data is collected using Global Positioning System (GPS) and aggregated over 15-minute intervals. For this analysis, the data has been divided to form a training set which consists of data from October

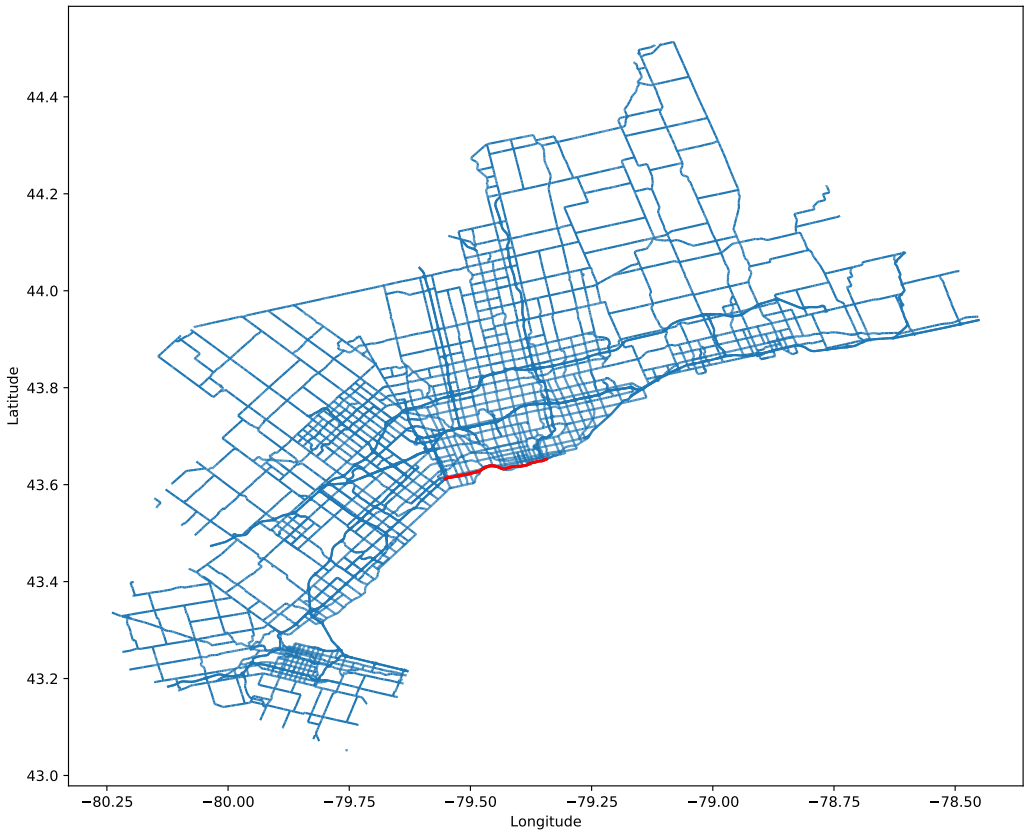


Figure 5.1: Location of the Gardiner Expressway

1st to 23rd, and a test set which consists of the remaining data from October 24th to 31st.

5.2.2 Model Selection

In this study, we compare the 6 methods listed below.

Historical Average

A historical average model is a simple model that partition the available training data into bins based on timestamp and the observed values are averaged within each bin. During prediction, the model identifies which bin the prediction timestamp belongs to and returns the stored value in that bin. For example, if the model is asked to predict for Thursday, October 31, 2019, at 9:00 AM, the model finds all instances of training data that are associated with Thursday at 9:00 AM and returns an average of those data points. Unless additional data are supplied, the model will always predict the same value for a time bin, such as every Thursday at 9:00 AM.

In this study, we create two historical average models. The first model is the

weekly model that contains separate time bins for every weekday. Since the data has a 15-minute interval, there are 96 timestamps in each day for a total of 672 time bins. Meanwhile, the second model is the daily model that only distinguishes between workweek and weekends for a total of 192 time bins. No model selection is required for this method since there is only one possible model.

Seasonal ARIMA

The seasonal ARIMA model is outlined in Section 2.5.1. Due to computational constraints, it is not feasible to exhaustively search through all possible orders for a seasonal ARIMA model. Therefore, we need to rely on various statistical tests in the model selection process. This process is well established and we follow the procedure outlined in [81] in this section. In addition, previous work has identified $(1, 0, 1)(0, 1, 1)_m$ as a set of orders that consistently performs well in the traffic prediction context [31].

First, we need to select the season differencing term (D). The autocorrelation function (ACF) is a useful test to determine whether the data is stationary. Figure 5.2 plots the the ACF for 6 selected links. As expected, the ACF plots show strong correlations at multiples of 96 lags (1 day), and some links show the strongest correlation at 672 lags (1 week). Visually, this shows that there is a clear seasonality in our data and suggests seasonal differencing. Figure 5.3 shows the ACF for the same 6 links after applying seasonal differencing once. The seasonally differenced time series exhibits a more stationary pattern, which is consistent with the findings of [31].

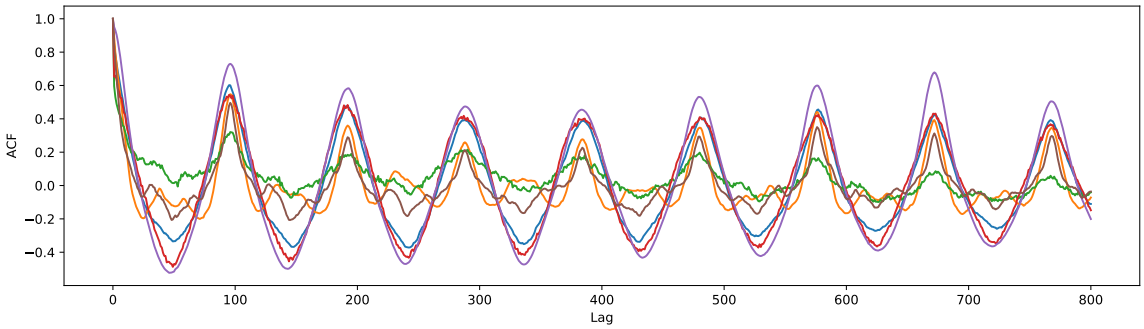


Figure 5.2: ACF of the original time series

To more objectively determine whether seasonal differencing is required, we can apply the Osborn, Chui, Smith, and Birchenhall (OCSB) test [82] to the original time series. Since seasonal ARIMA is a univariate model, the tests are separately applied to the time series of each link. Figure 5.4 illustrate the results of this test, which suggest that no seasonal differencing should be applied for both 1-day and 1-week

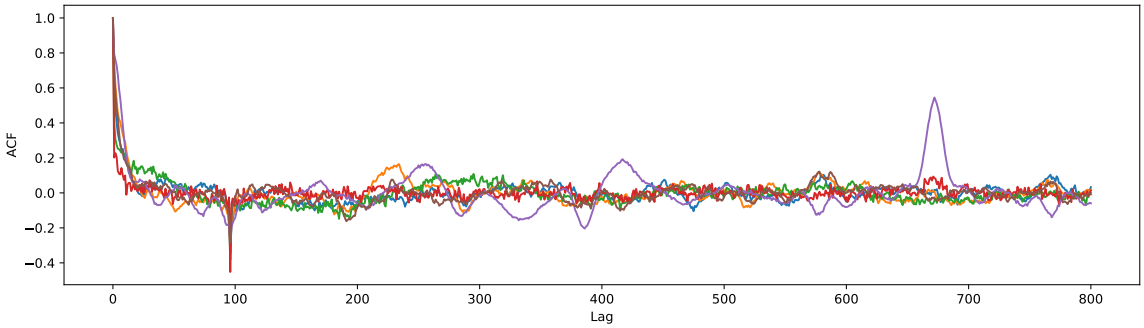


Figure 5.3: ACF of the first-order seasonally differenced time series (period = 1 day)

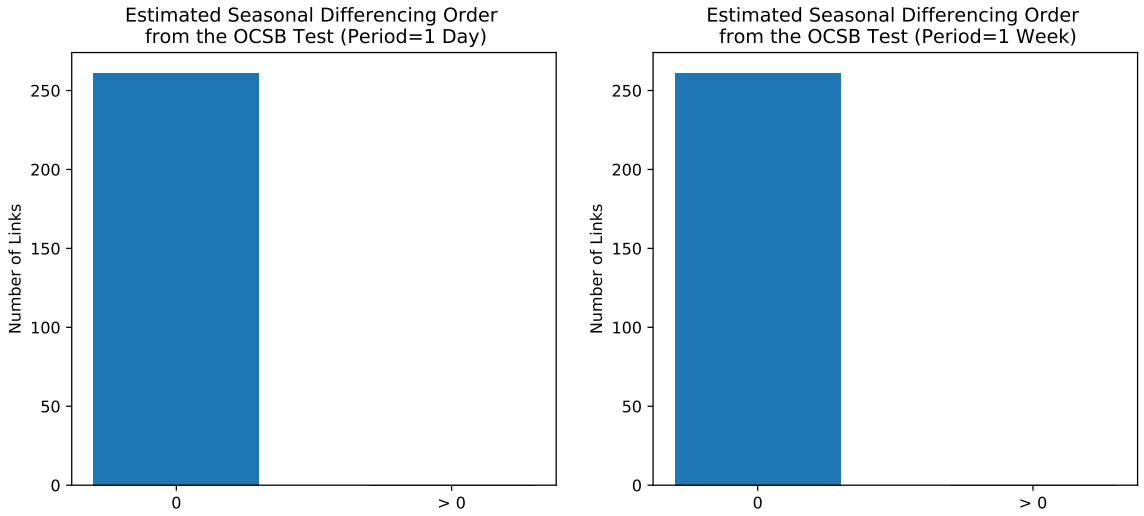
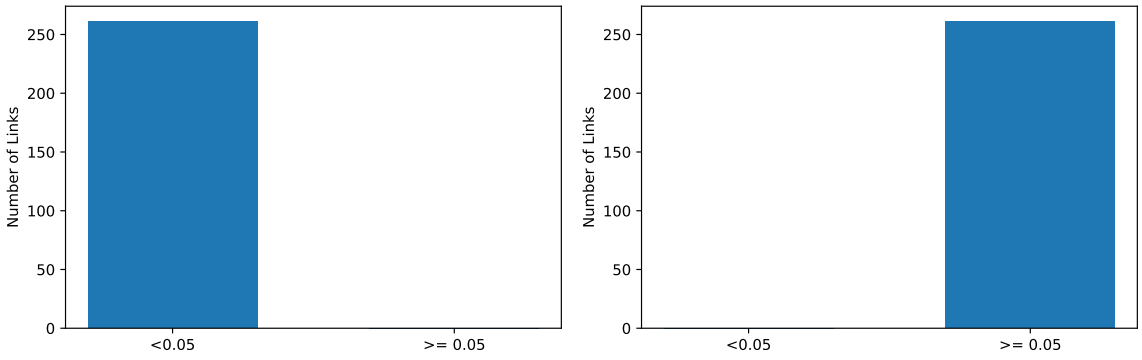


Figure 5.4: OCSB test result

seasons. This contradicts the findings of [31] and also the visual analysis above. The reason for this discrepancy could be because one month of data is not enough, or because the OCSB test was designed for financial time series and the technique is not suitable for the traffic setting. Ultimately, due to the clear seasonality shown in Figure 5.2, we decide to set the order of seasonal differencing (D) to 1 and the period (m) to 96 (1 day) for this analysis.

Once the seasonal differencing term (D) is known, we can apply a unit root test to objectively determine whether the resulting time series is stationary and decide whether further differencing is required. For this analysis, we elected to use both the Augmented Dickey Fuller (ADF) test [83] and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test [84], both commonly used in time series analysis. In the ADF test, the null hypothesis is that the time series has a unit root. If we fail to reject the null hypothesis, there is evidence that the series is non-stationary. On the other hand, the null and alternate hypotheses of the KPSS test are opposite of the ADF test, where



(a) p-values of the ADF Test

(b) p-values of the KPSS Test

Figure 5.5: Unit root test result

the null hypothesis is that the data are stationary. We use a p-value of 0.05 for the two tests and display the results in Figure 5.5. The results of the tests suggest that the seasonally differenced time series are stationary, and both tests are in agreement with one another. Therefore, no further differencing is required and the non-seasonal differencing order (d) is set to 0 for this analysis.

In the literature, the remaining terms of the seasonal ARIMA model are selected by fitting the model with different sets of orders and then select the set that minimizes an information criterion [81]. The information criterion is a model selection tool that measures the quality of the fitted model relative to other models. The most common information criterion in the literature is the Akaike information criterion (AIC) [85]; however, it is prone to overfitting when the sample size is too small. To address this, a correction formula to AIC has been developed known as corrected AIC (AICc), which we elected to use in this analysis. One important note is that model selection through minimizing AIC can only be performed using a fixed order of differencing (d, D) because differencing alters the underlying model fitting process. Therefore, d and D must be selected prior to the other orders.

There are different methods to fit a seasonal ARIMA model; however, the parameter estimation process typically takes a long time. This is especially true when the number of periods in a season (m) is large, as is the case with our data. Therefore, we perform this search using 20% of the links and begin with the baseline of $(1, 0, 1)(0, 1, 1)_{96}$ suggested by [31]. Throughout this process, we found no meaningful change to the AICc compared to the baseline. Therefore, we decide to use the model fitted using the baseline orders as the final model for comparison.

Exponential Smoothing

Based on the description in Section 2.5.2, there are a total of 12 models to consider. According to [81], the additive seasonal methods should be used when the seasonal variation is roughly constant throughout the series, while the multiplicative methods should be used when the seasonal variation is proportional to the level of the series. In traffic data, the seasonal variation is not dependent on the level; therefore, we will only consider none or additive seasonal components. To select the final model, we can once again perform parameter estimation and then find the model that minimizes the information criterion.

Similar to the previous section, we selected 20% of the links to fit the ETS models. We compared the corrected AIC of various models and found that ETS(A,N,A) produced the best result across different links. Therefore, this is the model that is selected to represent the class of exponential smoothing models.

Facebook Prophet

The Facebook Prophet algorithm is briefly described in Section 2.5.3. Since we only have 3 weeks of training data and there are no obvious seasonalities other than daily and weekly effects, we can let the Prophet algorithm automatically fit the data; therefore, no model selection procedure is needed for this method. Additionally, the Prophet model fitting process is significantly faster than both the ARIMA and the ETS models, completing in less than a second per link as opposed to seasonal ARIMA which can take over a minute.

Linear Regression

The linear regression model is outlined in Section 2.6.1. However, for long-term prediction, recent traffic history is unlikely to influence multiple days into the future. To account for this, we change the input features of the linear regression model to the following: day of the week, hour of the day, and whether it is a working day or a holiday. In addition, we find that one-hot encoding of the 3 types of features produces more accurate results; therefore, this is the linear regression model used in the experiments.

XGBoost

XGBoost [86] is an ensemble regression tree method that uses the boosting technique described in Section 2.6.2. We use the same input features as the linear regression model described above.

5.2.3 Evaluation Metrics

In this study, we fit the models using the 23-day training data then generate the predictions for the 8-day test period without supplying additional data. We then report the MAE, MAPE, RMSE of the prediction models, their definitions are outlined in Section 2.3.

5.3 Results and Discussion

The results of our analysis are summarized in Table 5.1. In addition, the violin plot in Figure 5.6 show the distribution of prediction error across links. We can see that the ETS model has the highest error, which corroborates the aggregated error metrics reported in Table 5.1. In addition, the distribution of the remaining models suggests that there are a few links that can be predicted with significantly higher accuracy compared to the other links.

The two plots in Figure 5.7 show the prediction on two different links over the last 2 days of October, which is separated from the training period by 6 days. Even with such a long prediction horizon, we can see that every model is able to capture the overall trend of the time series. In addition, we can see that the Prophet algorithm produces predictions that are much smoother compared to the other methods. One interesting thing to note is that the true observations seems to vary significantly between consecutive time steps, which may contribute to the prediction error. From the second plot, we can see the reason why weekly average is able to predict with almost no error on some links. For the most part, the weekly average prediction matches exactly with the ground truth. This suggests that the data itself is gap-filled using some weekly average method, which likely contributed significantly to its higher accuracy in our analysis.

During the model selection process, the Prophet model fitting process is significantly faster than both the ARIMA and the ETS models, completing in less than a second per link as opposed to seasonal ARIMA which can take over a minute.

Table 5.1: Performance comparison of long-term traffic speed prediction models

Model	MAE (km/h)	MAPE (%)	RMSE (km/h)
Weekly Historical Average	12.24	27.66	18.18
Daily Historical Average	11.88	27.18	17.11
Seasonal ARIMA	12.51	29.73	17.74
Exponential Smoothing	16.64	40.38	22.59
Facebook Prophet	12.83	29.36	17.78
Linear Regression	11.99	27.66	17.59
XGBoost	11.80	27.10	17.42

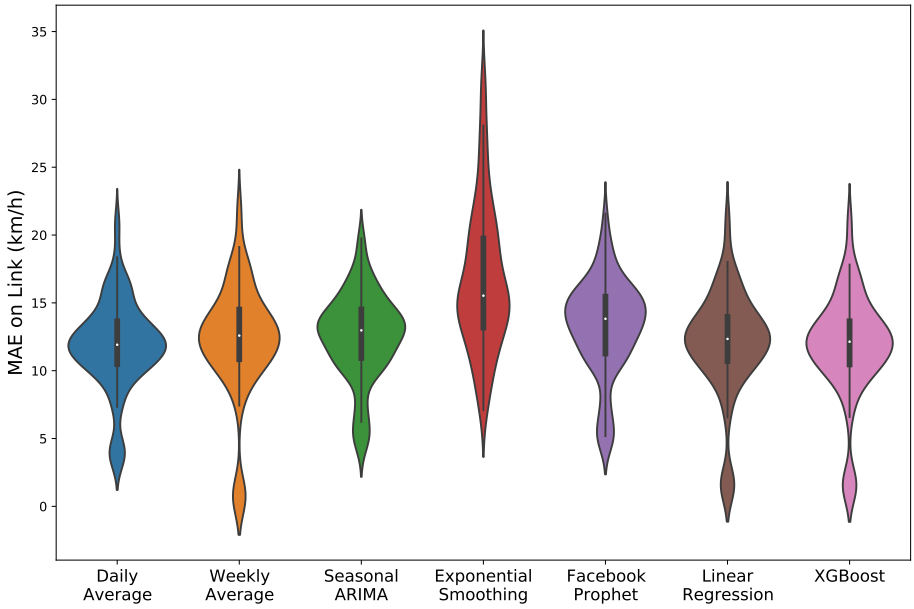
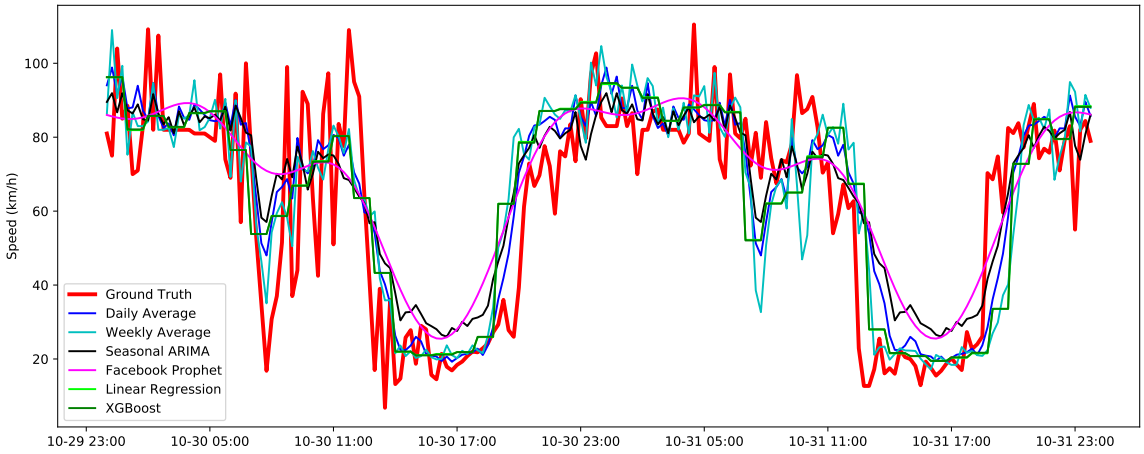


Figure 5.6: Distribution of mean error on each link

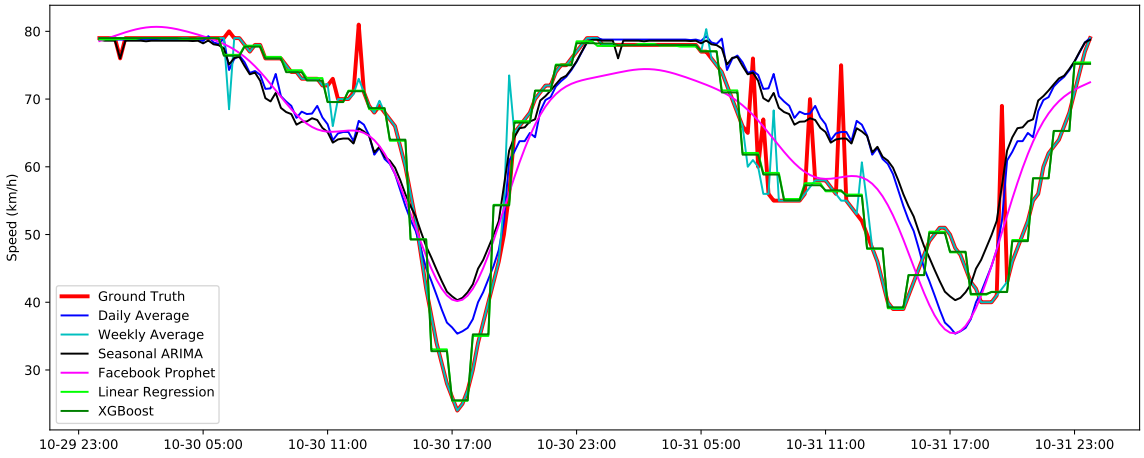
In addition, the Prophet algorithm is easy to use and applies well established time series analysis techniques. Overall, our analysis demonstrates that it can supplant traditional time series models such as seasonal ARIMA in long-term traffic prediction.

5.3.1 Boundary Between Short-term and Long-term Prediction

To delineate the boundary between short-term and long-term prediction, we trained the random forest model from the previous chapters with the GPS speed data. In addition to data of the Gardiner Expressway, we also included data from other highway links within the municipal boundary of Toronto for this analysis. The random forest model is chosen since it consistently performs well in our analysis of short-term prediction models. We trained the model using different prediction horizons, from 15 minutes to 90 minutes. For each horizon, we performed the same hyper-parameter tuning procedure from previous chapters (Table 4.2). Furthermore, to determine whether this boundary is applicable to non-highway links, we repeated the same analysis using GPS data on the arterial roads within the municipal boundary of Toronto. The results summarized in Figure 5.8 concludes that recent traffic history starts to adversely impact prediction accuracy and long-term prediction methods are preferable for prediction horizons beyond around 1 hour. In practice, we can unify short-term and long-term prediction methods using model averaging to intelligently combine the predictions generated by different models; however, implementing this approach is left for future work.



(a) Link 1

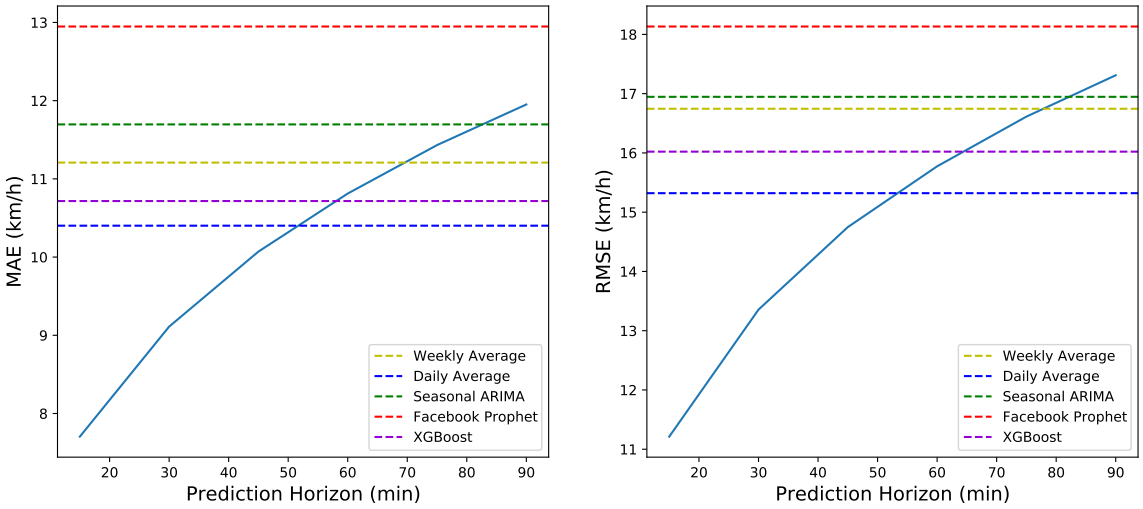


(b) Link 2

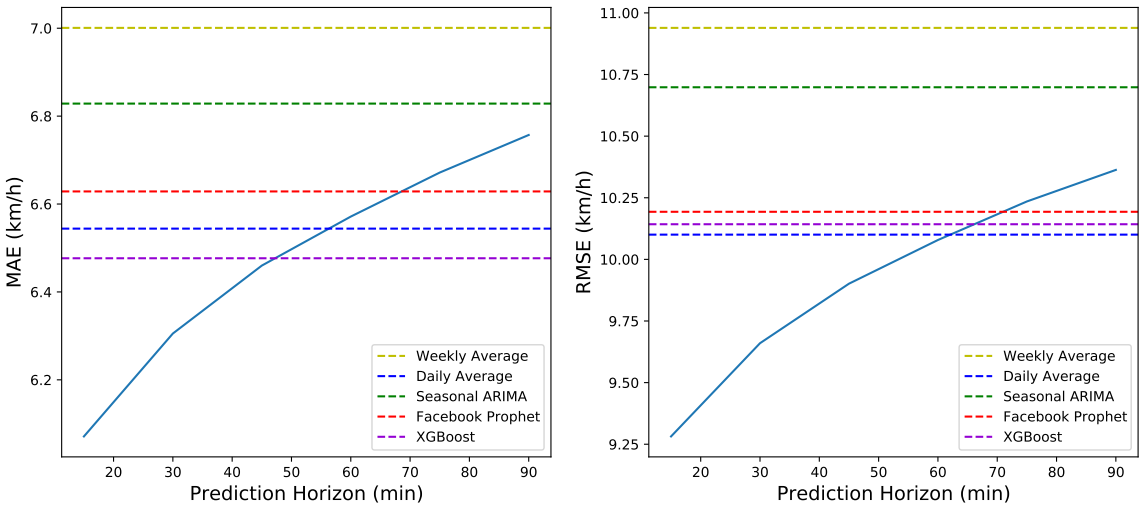
Figure 5.7: Line plot of predictions generated by various model and the ground truth on 2 selected links

5.4 Conclusion

In this analysis, we compared three different time series prediction methods, seasonal ARIMA, exponential smoothing, and Facebook's Prophet algorithm, against historical average and regression baselines. The results show that the historical average model has the lowest error while the exponential smoothing model has the highest error on our dataset. However, further analysis showed that the accurate result of the historical average model should be discounted because the data itself seems to be gap-filled using some historical average method. Nonetheless, the model performance is very close to one other and any of them could be suitably used for the long-term traffic forecasting problem. Lastly, we found that long-term methods start to overtake short-term methods for prediction horizons beyond 1 hour.



(a) Highway links



(b) Arterial links

Figure 5.8: Prediction error of short-term random forest regression compared to long-term methods

The long-term methods predict using long-term history rather than recent observations; therefore, the notion of prediction horizon is not applicable and the error is drawn as horizontal lines.

Chapter 6

Conclusion

6.1 Summary

Traffic prediction can provide tremendous aid to transportation agencies for managing traffic demand and it is an essential component in building intelligent transportation systems. Over the years, a myriad of solutions has been proposed for this challenging task, with deep learning and graph neural networks at the forefront of the current literature. This thesis is an extensive survey and a thorough evaluation of the different methods in this rapidly growing field of study. Throughout our analysis, we attempt to highlight potential issues and important considerations for deploying these solutions in the field.

In Chapter 3, we comparatively evaluated the different classes of short-term traffic prediction methods under a variety of traffic scenarios with the aid of traffic simulation software. We examined time series analysis, classical machine learning, and deep learning models. The experiment results demonstrated that random forest regression is potentially more powerful than deep learning models, at the expense of more memory and storage requirements. Furthermore, we highlighted the importance of including regional traffic history as input features to traffic prediction models. Finally, we also illustrated that individual parameters for each prediction location can improve the predictive performance of graph neural networks.

In Chapter 4, we expanded our evaluation of graph neural networks (GNNs) due to their prevalence in the recent literature. The results showcased the effectiveness of the attention mechanism in GNNs and further confirmed that location-specific parameters are beneficial to predictive performance. However, upon a closer inspection, we discovered that state-of-the-art GNN methods can potentially produce models that signifies traffic influence among faraway locations. This is inconsistent with traffic behaviour and indicates that the GNNs are detecting statistical correlations rather

than modelling traffic. Lastly, we established that while GNNs can be compact models, they are no more capable than the traditional machine learning model of random forest, which predates GNNs by over 15 years.

In Chapter 5, we examine long-term prediction methods to complete the exploration of traffic prediction. We showed that traditional time series analysis approaches are cumbersome and impractical for large-scale prediction. We then demonstrated that Facebook's Prophet, a recent forecasting algorithm, and other machine learning approaches can replace traditional time series analysis methods in the traffic prediction context. Finally, we determined that long-term prediction methods begin to eclipse short-term methods when the prediction horizon is longer than 1 hour.

6.2 Future Directions

This thesis compares different existing methods of traffic prediction using established evaluation method and metrics. Although we procured data from a variety of sources to create a range of traffic patterns, there are real-world problems that still need to be resolved before a traffic prediction system can be safely deployed. We briefly discuss four of these problems below.

- **Model Selection**: The hyperparameter optimization process used in this thesis is relatively simple and can be improved upon by using a framework similar to Ray Tune [87].
- **Multiple data sources**: The current models rely on a single source of data such as loop detectors or GPS. We need to explore alternative data sources that complement existing sensors to achieve a more comprehensive traffic view and safeguard against sensor failure. This would also raise the need for a data fusion method within the prediction system.
- **System robustness**: We need to create a prediction method that operates safely in the event of traffic incidents and sensor malfunctions. First, we need a way to reliably detect anomalies and malfunctions from sensor data. Second, we need a prediction method that can accurately predict even when traffic patterns deviate significantly from normal, such as during the presence of a blocked lane due to traffic accidents. Lastly, we need a method that can handle events where data could be missing for prolonged periods.
- **Traffic dynamics model**: The traffic prediction solutions examined in this thesis do not explicitly model traffic dynamics and our analysis in Chapter 5

illustrates that models can produce results that are inconsistent with the domain knowledge. A potential solution is to infuse a traffic dynamics model into a prediction model, which may require a more comprehensive observation than what current sensors allow and a departure from the macroscopic view of traffic prediction. Meanwhile, another direction is to incorporate sensor data into a traffic dynamics model, such as a traffic simulation software.

- **Integration into other systems**: Traffic prediction can support transportation management policies and enable adaptive control strategies. In this thesis, we compare the different models using aggregated error metrics such as mean average error; however, it is uncertain whether these metrics are appropriate for the needs of other components in intelligent transportation systems. Future work should consider tailoring prediction methods for the various adaptive traffic signal control models. One potential direction is to develop a method to identify the root cause of traffic congestion through causal analysis and improve control strategies.

6.3 Concluding Remarks

Recently, increasingly sophisticated deep learning solutions are proposed for traffic prediction. This thesis explored the major prediction methods that exist in the literature and raised some concerns regarding the state-of-the-art deep learning prediction models. For future research, it is important to consider whether these solutions can be safely deployed in the real world and whether they provide tangible improvement over existing solutions.

Bibliography

[1] HDR Corporation, "Costs of road congestion in the greater toronto and hamilton area: Impact and cost benefit analysis of the metrolinx draft regional transportation plan," Greater Toronto Transportation Authority, Tech. Rep., 2008.

[2] P. Mirchandani and L. Head, "A real-time traffic signal control system: Architecture, algorithms, and analysis," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 6, pp. 415–432, 2001.

[3] P.-W. Lin, K.-P. Kang, and G.-L. Chang, "Exploring the effectiveness of variable speed limit controls on highway work-zone operations," in *Intelligent transportation systems*, Taylor & Francis, vol. 8, 2004, pp. 155–168.

[4] M. J. Lighthill and G. B. Whitham, "On kinematic waves ii. a theory of traffic flow on long crowded roads," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 229, no. 1178, pp. 317–345, 1955.

[5] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

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

[7] X. Wang, C. Chen, Y. Min, J. He, B. Yang, and Y. Zhang, "Efficient metropolitan traffic prediction based on graph recurrent neural network," *arXiv preprint arXiv:1811.00740*, 2018.

[8] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," *arXiv preprint arXiv:2007.02842*, 2020.

[9] *Aimsun next: Your personal mobility modeling lab*. [Online]. Available: https://www.aimsun.com/aimsun-next/.

[10] P. I. Richards, "Shock waves on the highway," *Operations research*, vol. 4, no. 1, pp. 42–51, 1956.

[11] F. L. Hall, "Traffic stream characteristics," *Traffic Flow Theory. US Federal Highway Administration*, vol. 36, 1996.

[12] C. F. Daganzo, "The cell transmission model. part i: A simple dynamic representation of highway traffic," 1992.

[13] ——, "The cell transmission model, part ii: Network traffic," *Transportation Research Part B: Methodological*, vol. 29, no. 2, pp. 79–93, 1995.

[14] I. Yperman, S. Logghe, and B. Immers, "The link transmission model: An efficient implementation of the kinematic wave theory in traffic networks," in *Proceedings of the 10th EWGT Meeting*, Poznan Poland, 2005, pp. 122–127.

[15] I. Guilliard, S. Sanner, F. W. Trevizan, and B. C. Williams, "Nonhomogeneous time mixed integer linear programming formulation for traffic signal control," *Transportation Research Record*, vol. 2595, no. 1, pp. 128–138, 2016.

[16] I. Guilliard, F. Trevizan, and S. Sanner, "Mitigating the impact of light rail on urban traffic networks using mixed-integer linear programming," *IET Intelligent Transport Systems*, vol. 14, no. 6, pp. 523–533, 2020.

[17] H. J. Payne, "Model of freeway traffic and control," *Mathematical Model of Public System*, pp. 51–61, 1971.

[18] G. B. Whitham, *Linear and nonlinear waves*. John Wiley & Sons, 2011, vol. 42.

[19] C. F. Daganzo, "Requiem for second-order fluid approximations of traffic flow," *Transportation Research Part B: Methodological*, vol. 29, no. 4, pp. 277–286, 1995.

[20] A. Aw and M. Rascle, "Resurrection of" second order" models of traffic flow," *SIAM journal on applied mathematics*, vol. 60, no. 3, pp. 916–938, 2000.

[21] E. Kometani and T. Sasaki, "On the stability of traffic flow (report-i)," *J. Oper. Res. Soc. Japan*, vol. 2, no. 1, pp. 11–26, 1958.

[22] L. A. Pipes, "Car following models and the fundamental diagram of road traffic," *Transportation Research/UK/*, 1966.

[23] G. F. Newell, "A simplified car-following theory: A lower order model," *Transportation Research Part B: Methodological*, vol. 36, no. 3, pp. 195–205, 2002.

[24] ——, "Nonlinear effects in the dynamics of car following," *Operations research*, vol. 9, no. 2, pp. 209–229, 1961.

[25] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/.

[26] M. S. Ahmed and A. R. Cook, *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*, 722. 1979.

[27] C. Moorthy and B. Ratcliffe, "Short term traffic forecasting using time series methods," *Transportation planning and technology*, vol. 12, no. 1, pp. 45–56, 1988.

[28] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[29] T. G. Smith *et al.*, *pmdarima: Arima estimators for Python*, [Online]. Available from: http://www.alkaline-ml.com/pmdarima, Accessed 28th June 2020, 2017.

[30] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.

[31] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results," *Journal of transportation engineering*, vol. 129, no. 6, pp. 664–672, 2003.

[32] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 871–882, 2013.

[33] B. M. Williams, "Multivariate vehicular traffic flow prediction: Evaluation of arimax modeling," *Transportation Research Record*, vol. 1776, no. 1, pp. 194–200, 2001.

[34] Y. Kamarianakis and P. Prastacos, "Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches," *Transportation Research Record*, vol. 1857, no. 1, pp. 74–84, 2003.

[35] T. Ma, Z. Zhou, and B. Abdulhai, "Nonlinear multivariate time–space threshold vector error correction model for short term traffic state prediction," *Transportation Research Part B: Methodological*, vol. 76, pp. 27–47, 2015.

[36] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International journal of forecasting*, vol. 20, no. 1, pp. 5–10, 2004.

[37] P. R. Winters, "Forecasting sales by exponentially weighted moving averages," *Management science*, vol. 6, no. 3, pp. 324–342, 1960.

[38] *Forecasting at scale.* [Online]. Available: https://facebook.github.io/prophet/.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[40] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[41] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[42] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[43] G. A. Davis and N. L. Nihan, "Nonparametric regression and short-term freeway traffic forecasting," *Journal of Transportation Engineering*, vol. 117, no. 2, pp. 178–188, 1991.

[44] B. L. Smith and M. J. Demetsky, "Short-term traffic flow prediction models-a comparison of neural network and nonparametric regression approaches," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, IEEE, vol. 2, 1994, pp. 1706–1709.

[45] L. Zhang, Q. Liu, W. Yang, N. Wei, and D. Dong, "An improved k-nearest neighbor model for short-term traffic flow prediction," *Procedia-Social and Behavioral Sciences*, vol. 96, pp. 653–662, 2013.

[46] A. Ding, X. Zhao, and L. Jiao, "Traffic flow time series prediction based on statistics learning theory," in *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*, IEEE, 2002, pp. 727–730.

[47] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE transactions on intelligent transportation systems*, vol. 5, no. 4, pp. 276–281, 2004.

[48] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.

[50] F.-F. Li, A. Karpathy, and J. Johnson, "Cs231n: Convolutional neural networks for visual recognition," 2016. [Online]. Available: http://cs231n.stanford.edu/.

[51] M. S. Dougherty and M. R. Cobbett, "Short-term inter-urban traffic forecasts using neural networks," *International journal of forecasting*, vol. 13, no. 1, pp. 21–31, 1997.

[52] B. Park, C. J. Messer, and T. Urbanik, "Short-term freeway traffic volume forecasting using radial basis function neural network," *Transportation Research Record*, vol. 1651, no. 1, pp. 39–47, 1998.

[53] H. Dia, "An object-oriented neural network approach to short-term traffic forecasting," *European Journal of Operational Research*, vol. 131, no. 2, pp. 253–261, 2001.

[54] B. Abdulhai, H. Porwal, and W. Recker, "Short-term traffic flow prediction using neuro-genetic algorithms," *ITS Journal-Intelligent Transportation Systems Journal*, vol. 7, no. 1, pp. 3–41, 2002.

[55] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[56] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[57] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[58] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[59] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," in *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*, IEEE, 2015, pp. 153–158.

[60] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," *arXiv preprint arXiv:1801.02143*, 2018.

[61] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, "Lstm network: A deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.

[62] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[63] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[64] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[65] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, pp. 3844–3852, 2016.

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

[67] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

[68] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

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

[70] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[71] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, "Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction," in *AAAI Conference on Artificial Intelligence*, 2019.

[72] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "Dnn-based prediction model for spatio-temporal data," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, 2016, p. 92.

[73] B. Ashby, "Transportation tomorrow survey 2016: Data guide," Technical Report]. Available at http://dmg. utoronto. ca/pdf/tts/2016 . . ., Tech. Rep., 2018.

[74] I. Kamel, A. Shalaby, and B. Abdulhai, "Integrated simulation-based dynamic traffic and transit assignment model for large-scale network," *Canadian Journal of Civil Engineering*, no. 999, pp. 1–10, 2019.

[75] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 922–929.

[76] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 914–921.

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

[78] R. Ge, C. Jin, and Y. Zheng, "No spurious local minima in nonconvex low rank problems: A unified geometric analysis," in *International Conference on Machine Learning*, PMLR, 2017, pp. 1233–1242.

[79] M. Li and Z. Zhu, "Spatial-temporal fusion graph neural networks for traffic flow forecasting," *arXiv preprint arXiv:2012.09641*, 2020.

[80] H. C. Manual, "Highway capacity manual," *Washington, DC*, vol. 2, p. 1, 2000.

[81] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[82] D. Osborn, A. Chui, J. Smith, and C. Birchenhall, "Seasonality and the order of integration for consumption," *Oxford Bulletin of Economics and Statistics*, vol. 50, pp. 361–377, May 2009. DOI: 10.1111/j.1468-0084.1988.mp50004002.x.

[83] D. A. Dickey and W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root," *Journal of the American statistical association*, vol. 74, no. 366a, pp. 427–431, 1979.

[84] D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin, "Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?" *Journal of econometrics*, vol. 54, no. 1-3, pp. 159–178, 1992.

[85] H. Akaike, "A new look at the statistical model identification," *IEEE transactions on automatic control*, vol. 19, no. 6, pp. 716–723, 1974.

[86] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[87] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.