Bias Estimation of Spatiotemporal Traffic Sensor Data with Physics-informed Deep Learning Techniques

By

HAN YANG

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Civil Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Yueyue Fan, Chair

Michael H. Zhang

Miguel Jaller

Committee in Charge

2022

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Efficient operations of intelligent transportation systems rely on high-quality traffic data. Infrastructure-based traffic sensors, though providing major data sources for ITS, are subject to data quality issues. Existing studies have attended to identifying malfunctioning sensors or recovering missing data. Nevertheless, critical gaps remain to be addressed. Firstly, most studies only attempt to label sensors as either 'good' or 'bad'. In this way, any useful information contained in the partially 'bad' sensors is always discarded while the potentially erroneous information given by the partially 'good' sensors is always preserved. Secondly, the traffic dynamics attributes have not been effectively exploited when examining the sensor data, which is a missed opportunity for utilizing valuable information. This dissertation will try to fill these research gaps. In the dissertation, we first construct three networked sensor error correction models using transportation domain knowledge and Physics-Informed Deep Learning (PIDL) techniques based on fully-connected Neural Network, Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM), to extract features from the spatiotemporal sensor data and quantify the traffic flow measurement biases for all the sensors in the network. In addition, narrowing the scope down to a traffic segment with only two sensors, we develop another measurement error correction model using the Physics-Informed LSTM neural network combined with prior knowledge of macroscope traffic models. With minimum data requirements, this LSTM-based PIDL model is able to correct the measurement biases for traffic flux and occupancy simultaneously for both sensors. Overall, experimental results demonstrate the merits of combining machine learning techniques with domain knowledge of the physics of traffic flows in the context of sensor health monitoring and error estimation. The sensitivity analyses demonstrate the reliability and robustness of our results with respect to different testing scenarios.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Yueyue Fan, for all her unrelenting guidance and both technical and mental support during my entire Ph.D. studies. Writing this thesis was a long and uncertain journey that wouldn't have been possible without her. It is extremely fortunate for me to have a supervisor like her who cares so much about the students.

I would also like to extend the thanks to my dissertation committee members, Professor Michael Zhang and Professor Miguel Jaller, for serving as my committee and offering thoughtful comments and suggestions. It is the knowledge obtained from their courses that builds the foundation of my dissertation.

Special thanks are given to Dr. Yudi Yang, my fellow student and 15-year friend. I am always benefiting from his wisdom, and the work with him also helped form the initial idea of this dissertation.

Finally, I want to thank my family, for being a constant source of love, encouragement, and support.

Chapter 1 Introduction

1.1 Background

An intelligent transportation system (ITS) is a highly integrated system, designed to automatically collect networked data, analyze data and provide people with human-friendly information to optimize traffic management. As transportation demand continues to increase while physical roadway capacity is limited, smart transportation operations enabled by ITS become even more critical. The performance of ITS depends heavily on the quality of the data used by the system (Brumbaugh et al., 2018, Hawkins, 1980). For example, inconsistency and missing information hinder the accurate interpretation of data, while the overabundant information impedes effective downstream applications that utilize those data. In the age of Big Data, traffic network data are generated in massive amounts and from vast sources. Appropriate data pre-treatment is critical for effective and accurate downstream applications including monitoring, estimation, and decision. However, fundamental research focusing on understanding the data itself is relatively thin compared to the wide spectrum of applications that use the data as input. This research will focus on the quality of networked data, aiming to develop systematic mathematical approaches for data error estimation and proper data selection.

1.2 Traffic Detecting Sensors

Albeit the development of the Internet of Things (IoT) allows ITS to obtain traffic data via various mobile traffic sensors, like smartphones and smart vehicles, the penetration rate of mobile data is still low and the infrastructure-based traffic sensor systems are still by far the main input component for ITS. Also, due to privacy or commercial confidential concern, most of the mobile

data collected by commercial companies will not be accessible to the government or the public. Therefore, infrastructure-based sensors are still the major data source for public traffic management. There are mainly two types of infrastructure-based sensors, over-road sensors, such as ultrasonic radar, infrared sensors, and video cameras, and in-road sensors, including magnetic sensors and inductive loop sensors. Among these traffic sensors, inductive loop sensors are the most frequently implemented. A loop sensor can directly measure traffic volume (number of vehicles passing), occupancy (the proportion of time covered by a car) and indirectly infer traffic speed, characterizing the most important properties of traffic flow. The measurement data from all these sensors should be able to provide the traffic managers, the public, and the researchers with a good understanding of the traffic system.

In this study, the traffic volume and occupancy data are of our main interest. However, these traffic data are not always reliable. Empirical case studies have shown inconsistency and missing information from a large portion of network data. For example, it was reported that about one-third of the freeway sensors in California were not working correctly in PeMS data (Brumbaugh et al., 2018). On the other hand, for effective traveler information applications and traffic management applications, the traffic data errors should not be greater than 20% and 10%, respectively (Zhang et al., 2011). Quality control for archived data management systems (ADMS) has also been identified as a high-priority task recommended to the Federal Highway Administration (Shladover, 1993). Despite the importance of network data health monitoring, the research effort devoted to this subject is rather thin compared to studies that rely on data as input.

1.3 Unhealthy Sensor Data Identification

Among network data health monitoring literature, most studies focused on identifying 'bad' sensors so that data collected from those sensors can be filtered. However, an equally important question, how to estimate sensor data error (i.e., to find out 'how bad' a sensor is), has not been well addressed. In most practices, data imputation is applied to complete traffic data records after malfunctioned sensors have been identified using the aforementioned quality control procedure and the erroneous data are removed. To this end, there is a large body of literature using statistical learning techniques, including K-cluster, PPCA (probabilistic principal component analysis), and SVD (single value decomposition) (Chen et al., 2003, Weijermars & Van Berkum, 2006). However, those approaches neither fully utilize domain knowledge for transportation networks nor delve into statistical modeling of sensor errors. Simply relying on the data itself without assimilating transportation expertise produces the poor result of imputation when certain sensors are malfunctioning for an extended period of time.

In the recent paper coauthored by the author of this dissertation (Yang et al., 2019), a novel statistical model incorporating traffic network structure and nodal flow balance law was developed. In addition to identifying malfunctioning sensors, the model also shows the ability in quantifying measurement bias and recovering traffic flow data. Despite dropping out the need for the flow conservation for the entire network, that model still assumed nodal level flow balance, which does not always hold at a heavily queued location. In addition, by ignoring the time attributes of the observations, that model missed an opportunity to take advantage of the additional information contained in the traffic flow dynamics. In this study, we will resolve this limitation by making use of the temporal relations between the sensor data.

Note that studying traffic flow dynamics often needs another important traffic flow attribute, traffic density. Although a single loop detector system does not measure traffic density directly, traffic occupancy, as one of the two direct measurements, is often used to derive the density. However, the health of traffic occupancy is rarely discussed in the literature. Part of this study will fill this gap by introducing the traffic flow ordinary differentiate equation (ODE) model which draws a relation between the traffic flux (volume) and traffic density (occupancy) and using the model to examine the health conditions of both sensor readings.

While incorporating the time attribute, we will need to look at multi-dimensional time-series data containing complex non-linear feature information. Conventional analytical models are difficult, if not impossible, to suit such a need. Therefore, in this study, we will take advantage of the Physics-Informed Deep Learning (PIDL) techniques and combine neural networks with transportation engineering domain knowledge, to extract high-level features from traffic data and learn a non-parametric mapping to predict measurement biases in network sensors.

This work mainly consists of two parts:

First, we will look at the sensors in a network scope. We will develop physics-informed deep learning models based on the fully connected neural network, convolutional neural network (CNN), and long short-term memory (LSTM) to capture the spatial and temporal correlations in the traffic volume data and to estimate the traffic volume measurement biases of the networked sensors.

In the second part, we narrow down the scope into a single traffic segment with two sensors and will construct a traffic ODE model which draws an implicit relation between the traffic volume and the traffic occupancy and their measurement biases. We then combine the traffic model with

deep learning techniques and use the PIDL model to learn this implicit relation and simultaneously identify both volume and occupancy measurement biases.

Chapter 2 Literature reviews

2.1 Traffic Network Data

The ‘traffic data’ is the key to the modern traffic management system. Only correct, healthy, and integrated input data can support the effectual decision output.

Highway traffic sensor systems are still the primary information source for traffic management. Generally, there are two types of traffic sensors, over-road sensors like ultrasonic radar, infrared sensors, and video cameras, and in-road sensors like magnetic and inductive loop sensors. Among these traffic sensors, inductive loop sensors are the most frequently implemented. A loop sensor can measure directly traffic volume (number of vehicles passing), occupancy (the proportion of time it is covered by a car), and indirectly the traffic speed, characterizing the most important properties of traffic flow.

In the U.S., several states' Department of Transportation (DoT) built their own traffic performance monitoring system, like the Performance Measurement System (PeMS) of Caltrans. The operating of these systems requires input data from traffic detecting sensors, such as inductive loop sensors, magnetometers, and video cameras. Though studies have been conducted on replacing the inductive loop with other techniques like infrared sensors, among all the traffic detecting sensors in the current DoT traffic performance monitoring system, inductive loop sensors still play critical roles (Chen, 2003) and account for 95% of the collected data (Bickel et al., 2007).

By characterizing traffic volume, occupancy, and speed, the inductive loop sensors support traffic management in many aspects. Ramp metering, which controls the entrance gate of freeways,

operates based on the mainline flow density. The incident detecting system must identify and allocate incidents by extracting all the real-time traffic speed, volume, and occupancy data. Short-term congestion prediction acts similar to incident detecting but also requires historical data. Traffic models such as the car-following model and driver's behavior model often have numerous parameters, the calibrating and estimating of which also needs the network data from traffic sensors (Bickel et al., 2007).

2.2 Network Data Errors Estimation

2.2.1 Sensor Data Errors

A significant challenge associated with sensor data is its quality assessment. According to Margiotta (2002), for effective traveler information and traffic management applications, traffic data errors should not be greater than 20% and 10%, respectively. However, inductive loop detectors are found to be vulnerable. Several external impacts, including poor pavement condition, improper installation, and heavy vehicles, may lead to degrading the measurement performance (Klein et al., 2006). Empirical case studies have shown inconsistency and missing data from a large portion of road sensors. For example, it was reported that about one-third of the freeway sensors in California were not working correctly in PeMS data (Varaiya, 2001). Quality control for archived data management systems (ADMS) has also been identified as a high-priority task recommended to the Federal Highway Administration (Turner, 2007). Despite the importance of sensor health monitoring, the research effort devoted to this subject is relatively thin.

2.2.2 Malfunctioning Sensor Detection

Unhealthy sensors may lead to bad estimates of traffic volume and speed. Among sensor health monitoring literature, most studies focused on identifying malfunctioned sensors whose data should be discarded. A large group of previous quality control methods in this direction analyzed data from individual sensors separately. By setting reasonable thresholds for observed values and checking consistency among volumes, occupancy, and speeds, these studies could identify malfunctioned sensors (Payne et al., 1976). When the monitored variables are not within the acceptable ranges, for instance, too large traffic flow (e.g., greater than 3000 vehicles per hour per lane) or speed (e.g., greater than 160 km per hour) measurement is considered to be unreasonable, thus is dropped (Jagadeesh et al., 2014). Washington Algorithm (Jacobson et al., 1990) formulated a region in the volume-occupancy plane and rejected all the measuring data points outside the region. Rajagopal and Varaiya (Rajagopal & Varaiya, 2009) aggregated the ten diagnostic states provided by PeMS.

The basic threshold criteria are not always effective in finding malfunctioning sensors. Weijermars & Van Berkum (2006) found that the principle of flow conservation was a useful addition to the threshold criteria. Chen et al. designed a statistical algorithm to identify sensor error from California daily PeMS data and introduced the use of entropy scores to reflect the fluctuation of observed values (Chen et al., 2003). Though still looking at individual nodes, Waller et al. (2008) introduced an index of network consistency, which can be used to evaluate observed traffic flows.

This category represents the early endeavor in traffic data quality control and has evolved into a much more complicated validity criteria combination through the years. It is still prevailing in practice nowadays due to the convenient automation in a database environment. However,

although these simple methods could identify some obvious outliers, they may be challenging to find out small biases.

Instead of focusing on a single sensor's observed values at the one-time interval, another group of research efforts, emphasizes using the spatial and temporal correlation between nearby sensors and adjacent time intervals to detect sensor failure (Turner, 2007). Rajagopal et al., (2008) designed an algorithm with correlation statistics of neighboring sensors to detect multiple sensor failures in a network. Using neighboring lane correlation and time series from a traffic path, Dailey proposed a Kalman filter-based statistical algorithm to make an optimal volume and speed value prediction for inductive loops (Dailey, 1993). Sensor error is detected if the actual observation evidently deviates from the predicted value. In addition to those correlations, Nihan (1997) noticed that upstream and downstream traffic counts should be consistent, and the difference between spatially close detectors' counts cannot exceed the storage rate if all detectors work perfectly. Following the same vein, Vanajakshi & Rilett (2004) expressed this conservation principle using a partial differential equation based on macroscopic traffic flow theory and derived a nonlinearly constrained optimization problem to identify the malfunctioned sensors and estimate the measurement error. The spatial correlation of the above approaches is limited to neighboring sensors on a corridor. A more generic approach, simply based on the distance between sensors, was proposed to detect sensor mislabeling errors. Still, this work has not really utilized the network structure of a road traffic system (Kwon et al., 2004).

Recently, Sun et al. pointed out the limitation of those previous studies which did not consider spatial correlations among network sensors, and proposed a new approach for identifying bad sensors in a network setting. By enumerating all possible bases of a linear system consisting of the

network flow balancing equations, these researchers were able to identify broken sensors as those whose data are most frequently found inconsistent against data of the basis sets.

2.2.3 Erroneous Sensor Data Imputation

In most practices, data imputation is applied to complete traffic data records after malfunctioned sensors have been identified using the aforementioned quality control procedure and the erroneous data are removed. Li et al.(2014) divide these techniques into three types: interpolation, prediction, and statistical learning methods.

The interpolation can be done from either the spatial dimension or the temporal dimension. For spatial dimension, the missing or erroneous data can be filled using the average of the spatially neighboring data (Chen et al., 2003), or the K-Nearest Neighbors method (Liu et al., 2008). Similarly, temporal dimension interpolation uses the average of the temporally neighboring data. However, these methods have a strong assumption that the neighboring data has to be very similar in dimension, which cannot always hold. On the other hand, other than using similarity, these models don't consider the data correlation in either spatial or temporal dimensions and therefore, waste much useful information in the data.

The traffic data has long been treated as time series data. Taking use of the correlation and variance in the temporal dimension, one can predict the missing data using some time series models, such as the autoregressive moving average (ARMA) model (Nihan & Holmesland, 1980) and autoregressive integrated moving average (ARIMA) model (Williams et al., 1998). However, due to the complexity of the traffic flow data and the various traffic conditions that may affect the traffic, the conventional time series models don't fully capture the traffic flow dynamics.

Meanwhile, the models discard any data after the potentially erroneous observations which we want to predict, therefore, losing the information within that disposed data.

As the aforementioned prediction and interpolation approaches do not fully utilize the stochastic information of the networked traffic data, recent studies have attempted to use statistical learning methods to address this problem. Statistical learning models have been developed decades ago, but they are not prevailing until the most recent decade after the exponential growth in computing power. Methods like Probabilistic Principal Component Analysis and Markov Chain Monte Carlo multiple imputation method start to be used in the traffic data imputation (Choi et al., 2019, Farhan & Fwa, 2013).

Nevertheless, all the studies above pay no attention to the erroneous data itself. This may generate two significant problems: Firstly, if a bad sensor is identified, its data is just filtered and replaced with the new data imputed from other 'related' sensors. By doing so, we are abandoning the piece of information that may have the highest relationship with the truth. Secondly, if a slightly malfunctioning sensor is not identified as bad because its data quality still lies under an acceptable threshold, its data is simply kept and used to infer other sensors. Given this possibility, the entire data imputation process becomes unreliable.

In reality, different levels of sensor problems exist, from the most severe and manifest, zero call or constant call, to the modest but more undetectable ones, such as unbalanced sensitivity. Obviously, the former does not provide any useful information at all and should be tossed anyway, but the observation from the latter could still be potentially valuable in reconstructing traffic operation data. In many cases, sensor data are systematically deviated from the actual traffic volume due to causes such as counting neighboring lane traffic, missing motorcycles, and double

counting long vehicles. Albeit the data collected under those conditions significantly differ from what actually happened on the road, the systematic error part can be modeled mathematically and inferred with other sensors' counts in the traffic network, not limited to neighboring ones. With this being considered, instead of asking whether a sensor is good or bad, it leads to an equally important question, how to estimate sensor data error (i.e., 'how bad' a sensor is)?

To address these problems, Dr. Yang, Dr. Fan, and the author of this dissertation developed a Generalized Method of Moments (GMM) based sensor error estimation model (Yang et al., 2019). This model takes use of the network structure information and can quantify the sensor measurement bias and recover real traffic flow data accordingly (the more detailed contribution of this study will be discussed in Chapter 3). Nevertheless, this GMM model still has two limitations. Firstly, it cannot take advantage of the information in the temporal dimension of the traffic observation, because the model treats each observation as an independent sample. Secondly, in the spatial dimension, the model still requires a strong flow conservation assumption, which, however, does not always hold.

2.3 Deep Learning Techniques

With the development of ITS, there has been a data explosion in transportation engineering. Due to the complexity of spatial-temporal dependencies in traffic networks, conventional statistical learning techniques, like Bayesian networks, linear/logistic regression, and decision trees, lack the power to fully capture the features in the data (Nguyen et al., 2018). Over the recent decades, researchers have been using deep learning approaches to overcome this barrier. Deep learning is a type of machine learning and artificial intelligence that uses artificial neural networks with multiple layers to imitate the structure of the human brain. The word deep comes from the fact that

these artificial neural network models usually have multiple hidden (deep) layers. It is these hidden layers that increase the deep learning techniques' ability to process more complex and higher dimensional data.

Deep learning techniques have also demonstrated their powers in transportation engineering research. In the traffic prediction field, Polson & Sokolov, (2017) develop a deep learning model for short-term traffic flow prediction. After comparing the performance with a single-layer neural network model, they find that there is 14% less MSE. Because the deep learning approach can extract features from multi-dimensional data, they can have better predictions with the help of broader information. Zhao et al., (2017) construct an LSTM mode to predict traffic flow, and the LSTM model is found to outperform traditional time series models such as ARIMA, especially in long-term prediction. Moreover, deep learning can also be used for traffic condition perdition. For instance, Ranjan et al., (2020) build a hybrid CNN and LSTM model to perform city-wide congestion prediction and prove that the model can effectively capture from the data the relationship between both spatial and temporal dimensions. Cao et al., (2020) propose a CLM deep learning model for traffic speed prediction and show that this model has a better prognosis than SVR, MLP, and Lasso.

Researchers also attempt to use different types of deep learning methods to extract the features embedded in the high-dimensional traffic flow data and infer the missing information. Zhuang et al., (2019) treat the traffic volume data as spatial-temporal images and missing data as the blank region in the image and propose a CNN-based model to reconstruct the spatial-temporal data matrix. The model is able to impute missing data from missing ratios up to 50%. Cui et al., (2020) construct a two-layer bi-directional LSTM network that can capture forward and backward

temporal dependencies in spatiotemporal data to predict the network-wide traffic with missing values. This model can perform excellent missing data imputation, especially when training data contains multiple patterns of missing values.

Overall, the capability of capturing features from spatial-temporal dimensions makes deep learning an excellent approach to analyzing traffic data, traffic prediction, and missing data imputation. However, similar to the previous discussion, most studies assume that the data is either good or bad, or missing. The 'good' data is used to infer the 'bad' or 'missing' data without being verified. The 'bad' data is considered useless and discarded completely. Similar to the objective of the previous paper (Yang et al., 2019), this study aims to numerically quantify the sensor reading biases but using deep learning approaches rather than purely relying on analytical models.

Conventional deep learning approaches are purely data-driven, and the predictions made by these complete data-driven models may be physically inconsistent or infeasible (Karniadakis et al., 2021). The concept of physics-informed deep learning (PIDL) helps resolve this problem (Raissi et al., 2017). It can be considered a combination of data-driven and model-driven neural network models, so the neural network model has some prior knowledge of physical law (Figure 2.1). Initially, it was designed for solving ordinary and partial differential equations (Lagaris et al., 1998) and used mainly in physics. Still, the concept of physics information machine learning is quickly transcending to many engineering disciplines, including transportation engineering. Huang & Agarwal (2020) incorporate the macroscopic traffic conservation law with the physics-informed neural network. With prior knowledge of the traffic model, the PIDL can perform reasonable traffic state estimation with minimal traffic data. Mo et al., (2021) develop a family of neural network-based car-following models informed by the Intelligent Driving Model, Optimal Velocity Model,

and Full Velocity Difference Model and prove the better performance of neural networks informed by physics over those without.

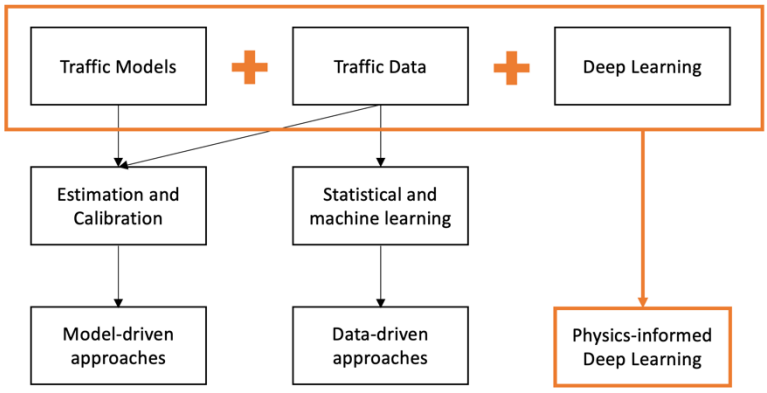


Figure 2.1 Physics-informed deep learning (Huang & Agarwal, 2020)

To extend the initial concept of PIDL, as shown in Figure 2.1, the physics informing can take place in any part of the learning process, e.g., training data preparation (Han et al., 2022), neural network structure design (Mo et al., 2021), and loss function design (Huang & Agarwal, 2020). With the aid of transportation engineering knowledge, PIDL has demonstrated a more efficient and effective learning process with less data demand. Therefore, in this research, we will take advantage of these attributes of PIDL and construct sensor bias estimation models based on it.

Chapter 3 Previous Work: Networked Sensor Data Error Estimation in Static Framework

In the previous study (Yang et al, 2019), by integrating a sensor measurement error model and a transportation network model, we propose a Generalized Method of Moments (GMM) based estimation approach to determine the parameters of systematic and random errors of traffic sensors in a road network. The roles and functionalities of the problem discussed in that paper are illustrated in Figure 3.1 and highlighted in blue. Steps 1 and 2 are the detection of entirely and partially malfunctioning sensors, respectively. Step 3 represents the standard denoising procedure. Step 4 is to correct systematically erroneous data. Step 5 is to impute missing data.

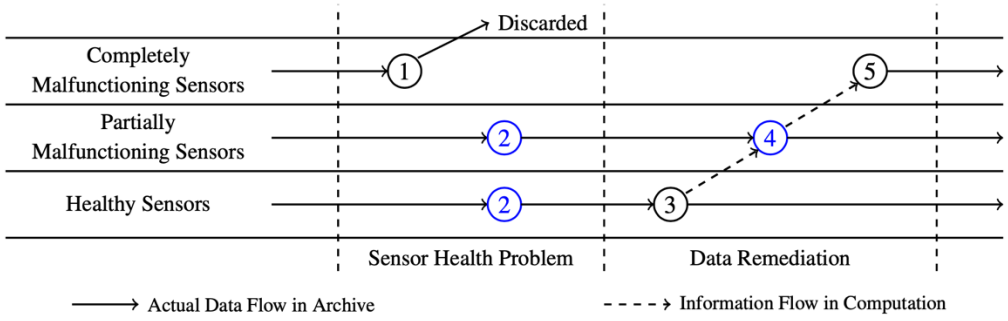


Figure 3.1 Sensor Data Processing Chart

A single traffic loop detects sensor typically generates two traffic measurements: traffic volume (flux) and occupancy. We group the traffic measurement errors into two categories:

Random error: This type of error is natural to any type of measurement. Even in a perfectly functioning sensor, the recorded traffic counts or occupancy can be ostensibly different from the true value due to inherently unpredictable fluctuation of the readings. With only the presence of random error, the measurement error has a zero mean and small variance.

Systematic error (bias): This type of error affects the observed data in a predictable direction. Traffic sensors may be subject to several types of systematic errors, but in this study we integrate all these types of errors into a single systematic error. We call a sensor malfunctioned if the systematic error is present. The main focus of this study is to identify those sensors and correct their observed data using systematic error estimates.

Traffic measurements given by the traffic sensors are often aggregated into certain time intervals (e.g., 10s, 1min). Mathematically, we use Z_a^t to denote the actual traffic volume or occupancy occurring at the sensor a during t^{th} time interval, V^t to denote the traffic reading given by this sensor. Then,

$$V_a^t = Z_a^t + W_a^t \tag{3.1}$$

Where, $W_a^t = V_a^t - Z_a^t$ is the statistics of random measurement error.

$E(W_a^t|Z_a^t)$, the expectation of the measurement error W^t, is the systematic measurement error we defined, and it is a function depending on Z,

$$E(W_a^t|Z_a^t) = f(Z_a^t) \tag{3.2}$$

For simplicity, in this study, we assume that $f(Z_a^t) = \beta Z_a^t$, namely a linear relationship between the real traffic and the systematic error, and we call β the measurement bias, which is not time dependent. Now we can convert equation to,

$$V_a^t = \beta Z_a^t + \varepsilon_a^t \tag{3.3}$$

Here ε_a^t can be treated as the white noise for the measurement, with $E(\varepsilon_a^t|Z_a^t) = 0$ and constant variance σ_a^2.

In this paper, our objective was to estimate the sensor measurement biases in a static framework, therefore, ignoring the temporal dimension, Equation (3.1) can be expressed as,

$$V_a = Z_a + W_a \tag{3.4}$$

Similarly, $W_a = V_a - Z_a$ is the statistics of random measurement error. $E(W_a) = \mu_a Z_a$ is the systematic measurement error. If we use $U_a = W_a - E(W_a)$, then $E(U_a) = 0$. The main focus of this paper is to estimate the systematic error by using the fact that traffic data are spatially correlated.

Then the next step is to a construct a network abstracting model converting a general traffic network into a directed graph, which follows the assumption of flow conservation, satisfying

$$\sum_{a \in \mathcal{A}^+(i)} Z_a - \sum_{a \in \mathcal{A}^-(i)} Z_a = 0, \ \ \forall i \in \mathcal{I} \tag{3.5}$$

Here $\mathcal{I}$ is the sets of all intermediate nodes in this abstracted network. $\mathcal{A}^+(i)$ and $\mathcal{A}^-(i)$ are the the incoming and outgoing links of the intermediate node i. Using $\boldsymbol{P}$ to denote the node-link adjacency matrix, the matrix form can be expressed as,

$$\boldsymbol{PZ} = 0 \tag{3.6}$$

Combining the flow conservation equation with measurement error model, we obtain,

$$\sum_{a\in\mathcal{A}^+(i)}\frac{V_a}{1+\mu_a}-\sum_{a\in\mathcal{A}^-(i)}\frac{V_a}{1+\mu_a}=\sum_{a\in\mathcal{A}^+(i)}\frac{U_a}{1+\mu_a}-\sum_{a\in\mathcal{A}^-(i)}\frac{U_a}{1+\mu_a},\qquad \forall i\in\mathcal{I} \tag{3.7}$$

Let $\boldsymbol{\beta}=[\beta_a]=[1/(1+\mu_a)]$, then the matrix form is

$$\boldsymbol{P}(\boldsymbol{V}\circ\boldsymbol{\beta})=\boldsymbol{P}(\boldsymbol{U}\circ\boldsymbol{\beta}) \tag{3.8}$$

Where ° stands for the Hadamard product.

Defining a zero-mean stochastic function of $\boldsymbol{\beta}$, $g(\boldsymbol{\beta})$, and using Generalized Method of Moment, we can estimate $\boldsymbol{\beta}$ by minimizing the Euclidean distance of $g(\boldsymbol{\beta})$,

$$\min_{\boldsymbol{\beta}>\mathbf{0}} g(\boldsymbol{\beta})^T\boldsymbol{W}\,g(\boldsymbol{\beta})^T \tag{3.9}$$

Assuming that the entire measurement timeframe is T and each measurement interval is t, we can define a zero-mean stochastic function of $\boldsymbol{\beta}$, e.g.,

$$g(\boldsymbol{\beta})=\boldsymbol{P}(\frac{1}{T}\sum_{t}^{T}\boldsymbol{V^t}\circ\boldsymbol{\beta}) \tag{3.10}$$

By defining $g(\boldsymbol{\beta})$ using (3.7), we are aggregating all measurements of one sensor into just one observation. This is a typical method of moment approach. However, doing so, we are losing a large portion of information contained in each individual measurement. Therefore, generalized method of moment gives more flexible way to aggregate the observations. Using GMM, we can define $g(\boldsymbol{\beta})$ as

$$g(\boldsymbol{\beta}) = \left[\left(P\left(\frac{1}{|\Psi(1)|} \sum_{t \in \Psi(1)} \boldsymbol{V}^{t\circ}\boldsymbol{\beta} \right) \right)^{T}, \ldots, \left(P\left(\frac{1}{|\Psi(\mathrm{K})|} \sum_{t \in \Psi(\mathrm{K})} \boldsymbol{V}^{t\circ}\boldsymbol{\beta} \right) \right)^{T} \right]^{T} \tag{3.11}$$

Instead of aggregating all the observations into only one, GMM allows the data to be grouped into flexible K groups. Here $\Psi(\mathrm{k})$ is the k^{th}set observations, containing all the observations that are grouped into this set. With all these setups, we can then solve the (3.6) minimization problem, which gives us the estimation of $\boldsymbol{\beta}$.

In this paper, we translate nodal flow balance law into structural equations, whose first moments are employed to estimate the systematic error ratio of sensors. The proposed framework allows a flexible data aggregation strategy, for which the traditional MOM and GLS are extreme cases. With such strategy, it is possible, without knowing random error ratios, to improve parameter identification by separating observations to more groups or to amend estimator consistency by clustering observations to fewer groups. The method in that study is capable of evaluating the level of data issue and correcting traffic flow data in addition to identifying malfunctioning sensors, while most previous sensor health studies concern only the latter. Second, it utilizes network structure of traffic monitoring system, while many previous studies that focus on spatial relation gave attention only to those immediately neighboring sensors on a corridor. Compared to the works in Sun et al. (2016) and Yin et al. (2017), which also exploits the network feature, our method lessens their requirement of flow balance on the entire network, which may take several hours to establish.

The estimation method in this paper is somewhat exemplary in the sense that it provides a conservative statistical approach to a novel problem. It only considers the most well-examined data type, traffic counts as well as probably most commonly accepted measurement error and

network models. In practice, there are multiple types of sensor data available, such as flow and density. Besides the static flow balance law based on network graph, other useful transportation domain knowledge like temporal dependency of traffic volumes, speed-density relationship and macroscopic traffic flow models can certainly provide additional information. This dissertation will attempt to incorporating this information into the error identification model and estimating the biases of sensor measurements by taking advantage of the fact that traffic data collected from sensors at different locations over a network and at different time over a timeline is indeed inherently correlated.

Chapter 4 Networked Sensor Biases Correction in Dynamic Framework

4.1 Introduction

In the previous work, the sensor error estimation model only looks at the spatial correlation between the networked sensors while ignoring the temporal information in the data. In this study, we will construct sensor bias correction models in a dynamic framework. Because the networked traffic sensor data are correlated from both spatial and temporal dimensions, it may be difficult for traditional statistical approaches to incorporate domain knowledge with the data and fully exploit its complexity. Machine learning techniques, like Physics-Informed Deep Learning (PIDL), however, well overcome this issue. Combining domain knowledge with deep learning techniques, PIDL has demonstrated the capability to efficiently extract features from complex engineering data and perform consistent predictions. In this study, we will construct three PIDL models (fully connected neural network-based, CNN-based and LSTM-based) to estimate the network sensor measurement bias, reconstruct erroneous traffic flow, and compare the difference in performance.

4.2 Methodology

4.2.1 Time Series Model Capturing Traffic Dynamics

Our abstracted traffic network consists of links, each of which has one sensor. Any network that does not comply with this setting in its original form can be processed to match this requirement, following the network abstraction algorithm [obj]. $\mathcal{G}$[obj] denote the network, $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$, where $\mathcal{N}$ and $\mathcal{A}$ are the set of nodes and links in the network. Let $s = |\mathcal{A}|$ denote the number of links.

For any node i, $\mathcal{A}_{(i)}^{+}$ is the set of links that has traffic flow coming in and $\mathcal{A}_{(i)}^{-}$ is the set of links that has traffic flow going out. We use $\mathcal{I}$ to denote the set of intermediate nodes and $n = |\mathcal{I}|$, the number of intermediate nodes.

Several macroscopic traffic variables, including traffic volume, speed, and occupancy, are naturally time series data. Time series analysis with historical traffic observations allows not only a short-term traffic forecasting model to predict the future traffic state but also the traffic data quality validation. Conventionally, the traffic data time series analyses tend to be model-driven, with many strong assumptions (Zheng & Huang, 2020) and these models are usually struggling in dealing with nonlinearity (Ma et al., 2017). Since the start of the big data era, data-driven models have well addressed these issues, especially with machine learning techniques, which are excellent at extracting complex and nonlinear attributes from data. On the other hand, the choice of models and the structuring of the input data significantly affect the model performance. A solely data-driven model without a good understanding of the data itself may not even converge. To this point, the domain knowledge does present a high value in helping construct a physical-informed machine learning model which some prior knowledge of the studying data.

Since we know that the traffic volume data are time series data, we will start looking at it using the conventional time series analysis. Let Z_a^t denote the true traffic volume for link a at t^{th} time interval. Following the autoregressive time series model (AR) of order p, we may write,

$$Z_a^t = \phi_a^1 Z_a^{t-1} + \phi_a^2 Z_a^{t-2} + \cdots + \phi_a^p Z_a^{t-p} + \epsilon_a^t \tag{4.1}$$

This is called an AR(p) model, where ϵ^t is the zero mean random error. The AR model forecasts the t^{th} traffic volume using a linear combination of t^{th} to $(t-p)^{th}$ traffic volume.

Combine it with the error model $V_a^t = \beta_a Z_a^t + \varepsilon_a^t$,

$$\begin{aligned} Z_a^t &= \phi_a^1 \frac{V_a^{t-1} - \varepsilon_a^{t-1}}{\beta_a} + \phi_a^2 \frac{V_a^{t-2} - \varepsilon_a^{t-2}}{\beta_a} + \cdots + \phi_a^p \frac{V_a^{t-p} - \varepsilon_a^{t-p}}{\beta_a} + \epsilon_a^t \\ &= \frac{\phi_a^1 Z_a^{t-1} + \phi_a^2 Z_a^{t-2} + \cdots + \phi_a^p Z_a^{t-p}}{\beta_a} + \gamma_a^t \end{aligned} \tag{4.2}$$

where, $\gamma_a^t = \frac{\beta_a \epsilon_a^t - \left(\varepsilon_a^{t-1} + \varepsilon_a^{t-2} + \cdots + \varepsilon_a^{t-p}\right)}{\beta_a}$ and $E(\gamma_a^t) = 0$.

According to Yang et al., (2019), if the flow conservation law holds at link a, then,

$$\sum\nolimits_{a \in \mathcal{A}^+(i)} Z_a^t - \sum\nolimits_{a \in \mathcal{A}^-(i)} Z_a^t = 0, \qquad \forall i \in I \tag{4.3}$$

In a more general case however, due to the fluctuation of the traffic flux, the flow conservation does not always hold. This means there exists another variable η_a^t, where

$$\sum\nolimits_{a \in \mathcal{A}^+(i)} Z_a^t - \sum\nolimits_{a \in \mathcal{A}^-(i)} Z_a^t = \eta_a^t, \qquad \forall i \in I \tag{4.4}$$

Again, combine it with the error model, and we obtain,

$$\begin{aligned} &\sum\nolimits_{a \in \mathcal{A}^+(i)} \frac{\phi_a^1 V_a^{t-1} + \phi_a^2 V_a^{t-2} + \cdots + \phi_a^p V_a^{t-p}}{\beta_a} - \sum\nolimits_{a \in \mathcal{A}^-(i)} \frac{\phi_a^1 V_a^{t-1} + \phi_a^2 V_a^{t-2} + \cdots + \phi_a^p V_a^{t-p}}{\beta_a} \\ &\qquad = \eta_a^t - \sum\nolimits_{a \in \mathcal{A}^+(i)} \gamma_a^t + \sum\nolimits_{a \in \mathcal{A}^-(i)} \gamma_a^t \end{aligned} \tag{4.5}$$

As discussed, in general, η_a^t does not equal zero. However, if we treat, for example, one-day observation as a cycle, $\sum_t \eta_a^t$ should equal zero. If we look at the time series for one day, we can approximately treat η_a^t as some random variable with $E[\eta_a^t] = 0$. Therefore, the right-hand side of

the above equation has $E\left[\eta_a^t - \sum_{a\in\mathcal{A}^+(i)} \gamma_a^t + \sum_{a\in\mathcal{A}^-(i)} \gamma_a^t\right] = 0$ and apparently, we do not observe this part.

In our study, the Vs are the observed traffic volume, the measurement biases (βs) are our estimation target, while the coefficient ϕs are not our interests. For a network, we can introduce a transforming function $\mathbb{F}$,

$$\boldsymbol{\beta} = \mathbb{F}(\boldsymbol{V}) \tag{4.6}$$

where $\boldsymbol{\beta}$ is a vector, consisting of all the biases of the sensors in the network ($\boldsymbol{\beta} = \left[\beta_{a_1}, \beta_{a_2}, \dots, \beta_{a_n} \dots\right]^{\boldsymbol{T}}$ $a_1, a_2 \dots, a_n \dots \in A$). $\boldsymbol{V}$ is a matrix, containing all the sensor readings for the network of the entire day. Here, the function $\mathbb{F}$ may contain several layers of transformations, such as network structures that spatially map the observations to corresponding nodes, time series predictions that connect the measurements in the temporal dimension, and biases estimations, which input with a certain form of observations predict sensor measurement errors. Given the similarity between the structures of these virtual layers and the neural networks, we consider the deep learning models as an appropriate approach to perform the transformations for us and use the knowledge of the time series model to aid the neural networks learning process.

4.3.2 Data Preparation

The open-source microsimulation software Simulation of Urban Mobility (SUMO) (Behrisch et al., 2011) is a commonly used tool in the traffic research (Soares et al., 2013). A large number of transportation research papers are using SUMO simulation data for building and testing machine learning models (Kheterpal et al., 2018) (Chandramohan et al., 2019) (Koh et al., 2018). With

TraCI (Traffic Control Interface) (Wegener et al., 2008), it can easily communicate with the Python environment and take commands from there, giving us much flexibility.

In this study, we will use SUMO to build the traffic network and generate the training and test cases.

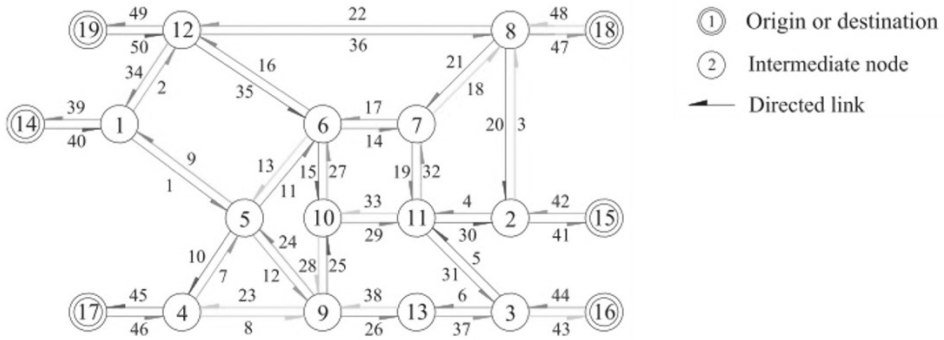


Figure 4.1 Modified Nguyen-Dupuis network

With SUMO, we built a bidirectional modified Nguyen-Dupuis network which consists of 6 origin-destination (OD) pairs, 19 intermediate nodes, and 50 directed links. Since we consider 1-day as a cycle, each simulation simulated the traffic flow for the 24-hour time. We generated random OD pairs for a day and randomly split the OD demand into hourly volume following a commonly used hourly flow pattern base (Figure 4.2) with some randomness (Figure 4.3). There are two extra considerations for the preparation of the simulation:

1. The mapping from OD pairs to link trips is non-linear, due to the fact that when the OD demand changes, certain OD pair volumes may change the path. With this consideration, we want to design the OD demands so that they can depict relatively intact mapping relations.

2. The OD demand and hourly traffic volume are designed so that some links do observe traffic congestions and shockwaves because the shockwave is one major cause of the violation of flow conservation law at intermediate nodes (intersections), and we want to test how the models handle the presence of flow imbalance.

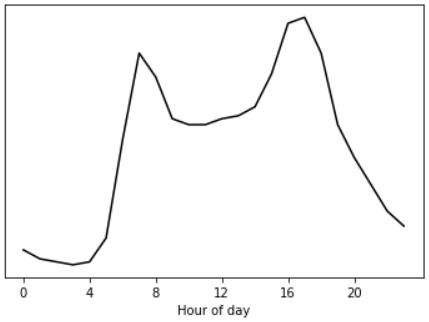


Figure 4.2 The base of average hourly flow ratio in network

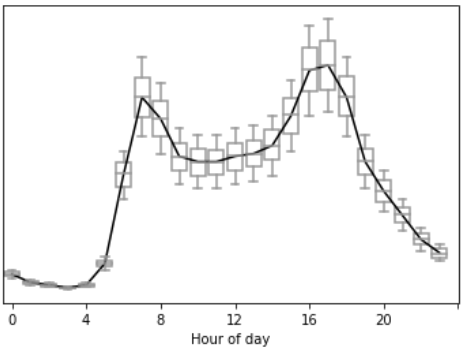


Figure 4.3 The distribution of hourly flow in network

Before starting to construct the neural network models, we first propose the ways to prepare training data that contains the proper traffic domain knowledge: 1. Given that the spatial relationship between the sensors in a network represents the network structure, to allow the model better to understand the network structure, we want to include as many sensor data as possible within the studying network. 2. Similarly, if we want the model to learn the temporal patterns of the flow correctly and broadly, the training data should also be complete, covering most of the

scenarios that could happen in the real world, e.g., covering both peak and off-peak hours, covering both stable and fluctuated flow situations. 3. Since ultimately, the model also needs to know the proper mapping from the measurements to the measurement biases, we need to manipulate the initial observations by supplying it with some manually generated random biases and train the model with the manipulated sensor observations and the 'known' biases. Since we are preparing for the training cases and the training cases serve as the ground truth, these parts of data should be unbiased sensor readings. These unbiased readings can be obtained, at the time or soon after the sensors were freshly installed or physically calibrated.

In this study, we use SUMO to simulate and collect the base traffic counts (actual traffic counts), on top of which, we will add the measurement errors (biases and white noise). Similar to the extra considerations for the simulation preparation, when adding the sensor measurement biases, we also want to ensure their representativeness so that the deep learning model can study a complete mapping process.

Typically, the traffic flow data can be treated as periodic data with each cycle of $t = 24$ hours. Considering a network of s sensors, each of which measures the traffic flow every δt time, a 24-hour measurement of the traffic flow of the network generates a $s \times n$ $(n = t/\delta t)$ matrix. Thus, a sample of d days can be expressed using a tensor of $q \times s \times n$.

To generate the training dataset from the simulated data, with the network of s sensors, we repeat the below steps for Y iterations:

Algorithm 4.1:

Step 1: At y^{th} iteration, we first creating two s-dimensional vectors $\boldsymbol{\beta_y}$ and $\boldsymbol{\sigma_y}$ representing sensor measurement biases and random error variance. Each element of $\boldsymbol{\beta_y}$ follows a uniform distribution of range $[0.5, 1.5]$ (e.g., for a^{th} sensor, $\beta_{y,a} \sim U(0.5,1.5)$), because we assume that most sensor reading biases should fall in this range. Similarly, each element of $\boldsymbol{\sigma_y}$ follows a uniform distribution of range $[0, \sigma_i]$] (e.g., for a^{th} sensor, $\sigma_{y,a} \sim U(0, \sigma_a)$).

Step 2: In reality, it is very likely that a portion of the sensors in the network are freshly installed or calibrated so that readings form these sensors should not have biases. According to Yang et al., (2019), this information may benefit the estimation of measurement biases for the rest sensors. Therefore, we also have certain amount of healthy sensors with unbiased measurements ($\beta = 1$) in our network. In our experiments with SUMO simulation data, we introduce the healthy sensors by defining that each sensor in the network has a probability of p_h to be unbiased ($\beta = 1$).

Step 3: For each iteration, we will generate D samples, hence, for the y^{th} iteration, d^{th} sample, and i^{th} sensor, its measurement bias $\beta_{y,a,d}$ follows normal distribution $N(\beta_{y,a}, 0.2\beta_{y,a})$ and accordingly, the observations of this sensor is generated following normal distribution $N(\beta_{y,a,d}\, Z_a, \sigma_{y,a}\beta_{y,a,d})$. Here, Z_a is the true traffic volume of sensor a in simulation case, or the unbiased traffic readings of sensor a if real observations are used.

In total, we generate 100 years (Y) of 365 days (D) traffic flow data, among which 80×365 groups data are training set, and 280×365 groups are validation set. For each of the 50 sensors, we have 144 observations (10 minutes aggregation each) for one day, and we generate one random

bias for that day. Hence, for the training set, we obtain traffic flow data of dimension (29200, 144, 50) and corresponding sensor biases of dimension (29200,50). For the validation set we get (7300, 144, 50) traffic flow data and (7300,50) sensor biases. We set the default value of both σ_a and p_{calib} to be 0.2.

4.3.3 Physics-Informed Neural Network Models

Fully Connected Network

After preparing the training cases, we have the manipulated sensor measurements as the neural network model input data and the real measurement biases as the expected model's output. We will construct and compare three types of deep neural network models: fully connected neural network, convolutional neural network, and Long Short-Term Memory network. Now we are starting with the fully connected neural network.

The fully connected neural networks are one group of the simplest neural networks. As its name reveals, every neuron in one layer is connected to all the other neurons of its neighbor layers. Despite its simplicity, with proper design, fully connected neural networks can still capture particular non-linearity and give good classification or prediction.

If we use x to denote the neural network input (traffic sensor measurements), $\hat{y}$ and y to denote the network output (estimated sensor biases) and target output (true sensor biases), L to denote the number of layers, w^l and b^l to denote the weights and biases between layer $l-1$ and l (w_{ij}^l and b_{ij}^l the weight and bias between i^{th} node in layer $l-1$ and j^{th} node in layer l), and f^l the activation function of layer l, then the overall mathematic expression can be written as,

$$\hat{y} = f^{L}\left(\boldsymbol{w}^{L-1} f^{L-1}\left(\ldots f^{1}(\boldsymbol{w}^{1}x + \boldsymbol{b}^{1})\right) + \boldsymbol{b}^{L-1}\right) \tag{4.7}$$

Activation function: The output of a layer of neural network usually passes through the activation function before being fed into the next layer, and the differentiable nonlinear activation function is the key that enables neural networks to learn complex and nonlinear problems. Some commonly used nonlinear activation functions include ReLU activation $f(x) = max\ (0, x)$, Tanh activation $f(x) = tanh\ (x)$ and Sigmoid activation $f(x) = \frac{1}{1+e^{-x}}$. The convolutional neural networks usually use ReLU as the activation function, while recurrent neural networks like LSTM commonly adopt Tanh activation.

Model optimization: The optimization of the neural network is based on computing the gradients of the loss function with respect to the model parameters ($\theta = [\boldsymbol{w}^{1}, \ldots, \boldsymbol{w}^{L}, \boldsymbol{b}^{1}, \ldots, \boldsymbol{b}^{L}]$):

$$\nabla_{\theta}\mathcal{L}(\theta) \tag{4.8}$$

Using backpropagation, we can calculate each $\frac{d\mathcal{L}}{dw_{ij}^{l}}$ and $\frac{d\mathcal{L}}{db_{ij}^{l}}$, and accordingly update w_{ij}^{l} and b_{ij}^{l} using the gradient descent method:

$$w_{ij}^{l} = w_{ij}^{l} - \alpha\frac{dL}{dw_{ij}^{l}} \tag{4.9}$$

$$b_{ij}^{l} = b_{ij}^{l} - \alpha\frac{dL}{db_{ij}^{l}} \tag{4.10}$$

where α is the learning rate, one of the most important hyper-parameters for successful convergence.

Learning rates: Generally, the learning rates determine the magnitude the neural network updates the parameter. Ideally, learning rates should be tuned to allow neural networks both converge and converge in a reasonable time. A fixed learning rate often cannot satisfy this demand as larger learning rates may result in suboptimal, while lower rates usually lead to much longer convergence

time. To resolve this issue, adaptive learning rates are commonly used. On the other hand, instead of using a monotonically decreasing learning rate, we can make the learning rate oscillate by using cyclical learning rates.

Adam: Adam optimizer inherits the strengths of two gradient descent methodologies, momentum and root mean square propagation and is able to adjust the learning rate during the learning process to reduce the oscillation of learning progress and meanwhile reduce the chance falling onto local minina.

Cyclical learning rate: By defining an upper and lower bound, we allow the learning rate to oscillate between them. In doing so, the periodically increasing learning rate helps untrap the saddle point or local minima, and the oscillation also offers more flexible learning rate options.

We now construct an initial fully connected neural network starting with three hidden layers.

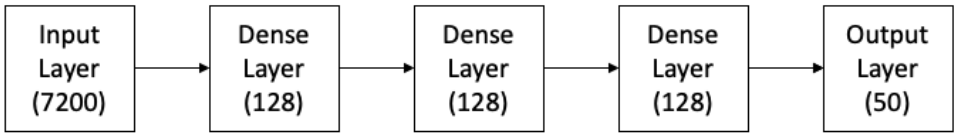


Figure 4.4 Initial fully connected network

Input layer: The input layer is crucial in passing prior knowledge to the neural network. We know that the traffic flow pattern recurs every day, so based on Equation (4.6), we use one entire day's observation as a single sample for predicting sensor biases. Since sensor readings are aggregated into 10 minutes, each sensor gives 144 observations daily. For the network of 50 sensors, a one-day sample is expressed using a 50 by 144 matrix. Due to that the fully connected neural network only takes the vector as input, we need to flatten the matrix to a 7200-dimensional vector. Therefore, the input layer takes a 7200-dimensional vector as input.

Dense layers: Each dense (fully connected) layer consists of 128 neurons.

Output: We expect the model to predict measurement biases for each sensor; hence for the modified Nguyen-Dupuis network, the model should output a 50-dimensional vector.

Loss function: This is a supervised learning model. The loss function is used to evaluate the distance between the estimated and the true (expected) sensor biases. The neural network training process is to optimize the model by minimizing the loss function. Some commonly used loss functions include mean square error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). We have the manipulated sensor measurement biases serve as the true βs to compute the loss function. We are using MSE as the loss function for our neural network models (fully connected, CNN and LSTM models), because from the statistical perspective, $MSE(\hat{y}) = Bias^2(\hat{y}) + Var(\hat{y})$ evaluates and balances both the model biases and variance. For each sample, the loss function is defined as the mean squared error of the predicted biases $\hat{\beta}$ and true biases β:

$$LOSS = \frac{1}{50}\sum_{a}\left(\hat{\beta}_a - \beta_a\right)^2 \tag{4.11}$$

where n is the number of observations.

Performance metric: To measure the model performance, we introduce a metric called Mean Absolute Traffic Flux Error (MAE).

$$MAE = \frac{1}{ns}\sum_{i=1}^{n}\sum_{a=1}^{s}\left|\frac{\widehat{f_i^a} - f_i^a}{f_i^a}\right| \times 100\% \tag{4.12}$$

where f_i^a is the true traffic flux rate during i^{th} observation for the a^{th} sensor; $\widehat{f_i^a}$ is the corresponding traffic flux reading corrected by the model. Assume that the model predicts the measurement bias to be $\hat{\beta}^a$ for the a^{th} sensor. For the sensor's i^{th} traffic flux observation v_i^a, the corrected traffic flux rate $\hat{f}_i^a$ can be computed using,

$$\hat{f}_i^a = \frac{v_i^a}{\hat{\beta}_i^a} \tag{4.13}$$

Note that, instead of measuring the model performance with sensor biases $\boldsymbol{\beta}$, we are using the traffic flux rates. This is because the numbers of sensor biases $\boldsymbol{\beta}$ are not of great attraction in real-world traffic management and the correctness of $\boldsymbol{\beta}$ estimation is hard to interpret (e.g., a 20% improvement in $\boldsymbol{\beta}$ estimation does not show a clear practical meaning). While on the other hand, the metric of traffic flux rate reading improvement is straightforward, and it originally served as our ultimate goal for the study. During the model training processes, backpropagation computes the weights of the neural networks with the gradient of the loss function but the performance metric.

Other hyper-parameters: The triangular learning rate scheduler allows the learning rate to jump between the bounds cyclically. The number of epochs and batch size are set to be 50 and 64, respectively. The 'Adam' optimizer is used as the optimization method. These parameters apply to the convolutional neural network and LSTM neural network that will be discussed in the next part as well.

Since the fully connected neural network has a relatively simple structure, the major hyperparameters we need to tune include the number of layers, the number of neurons in each layer, and the dropout probability between each layer. The number of layers and neurons in each

layer are tuned simultaneously. With the initial number of layers set to be 3, we can see a slight MAE improvement when we increase the layers to 4, while no further improvement is observed with more layers. 256 neurons per layer are found to be adequate, and the model does not benefit from more neurons. The dropout probability from 0 to 0.2 is tested and results show that no dropout case performs the best. Therefore, the final fully connected neural network model is fixed as:

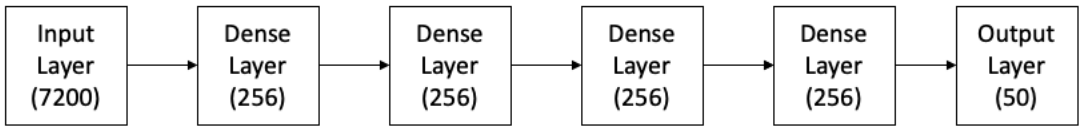


Figure 4.5 Finalized fully connected network model

Convolutional Neural Network

Convolutional Neural Networks (CNN) are special kinds of deep learning models that are usually used for computer vision tasks. Some of the advantages of using CNNs are as follows:

1) They are capable of extracting features straight from the raw data, whereas in traditional statistical models, features are usually engineered manually.
2) CNNs enable weight sharing over different parts of the input that reduces the number of parameters effectively, making it computationally feasible to train on a large set of data as compared to traditional deep fully connected networks.
3) CNNs are immune to small transformations in the data such as scaling, translating, skewing and distorting.

Given a network of s sensors and each has n observations for a day, the sensor measurement data is a two-dimensional s by n matrix for this day. Conventional CNNs are usually two-dimensional (2D-CNN) and are commonly used to extract features from three-dimensional data like images.

Since, in this study, the traffic observation data are two-dimensional, we are going to use the one-dimensional CNN (1D-CNN) which only performs 1D convolution and has a simpler configuration.

For each day's observation, we have the input date being a s by n matrix. Assume that we have a total of L convolutional multiple input and output channels layers. Each layer has s input/output channels containing observations from one sensor. The convolutional layers apply convolution filters to the data, and the data also passes through the activation function and pooling, before it is handed over to the next layer. After all the convolutional layers, the s channel data from the last convolutional layer are flattened and concatenated into a vector which then goes into a fully connected neural network layer. The output of this 1D-CNN model is set to be an s-dimensional vector, each element of which represents the measurement bias of one sensor.

We built the convolutional neural network model, as shown in Figure 4.6.

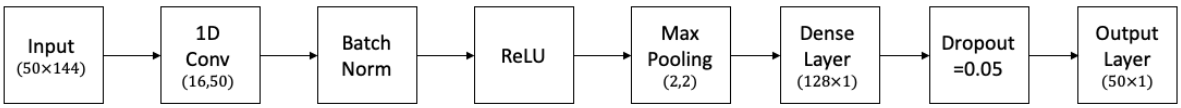


Figure 4.6 Convolutional neural network model

Input: Unlike fully connected neural networks, the convolutional neural network can take matrix as input. Therefore, we will feed the model with a 50 by 144 matrix representing a one-day observation sample. Each row contains the 144 aggregated observations from each sensor, and every element in this row is temporally correlated with the rest. Similarly, each column represents the observations of 50 sensors at a same time interval and the observations are spatially correlated.

Convolutional layers: Too few layers may result in underfitting while too many layers cause overfitting and much higher computational complexity. Different numbers of convolutional layers (1-4 layers) are tested, and layer of 2 is selected to achieve the best performance metric without overfitting.

1D Conv Layer: Each 1D convolutional layer has 50 channels. In terms of the filter size, smaller filters fail to capture broader relationships along the dimension, while larger filters lead to higher computational time. We choose a filter size of 16 after comparing the performance of filter sizes 2, 4, 8, 16, and 32.

Max pooling layer: The pooling layer helps reduce the data dimension and avoid overfitting. Max pooling only selects the maximum values from the convoluted data, thus extracting the most important feature. Here both stride and filter width to be 2.

Dense layer: A fully connected layer of 128 neurons takes the flattened output from the CNN layer and passes it to the output layer after a dropout layer. The dropout probability of 0.05 is found to obtain the best model performance.

The output layer and loss function are defined the same as for the fully connected neural network.

Long Short-Term Memory

Long short-term memory (LSTM) is a special type of recurrent neural network (RNN). RNN is a preferred neural network for sequential data, such as time series data, due to its capability of extracting the contextual information between different time stamps. However, conventional RNN may encounter vanishing or exploding gradient problems, which makes it unable to capture long-

term dependencies. The LSTM, instead, well addresses this issue. On top of the conventional RNN cell state, LSTM adds three types of gate layers (forget, input and output) to control the information flow.

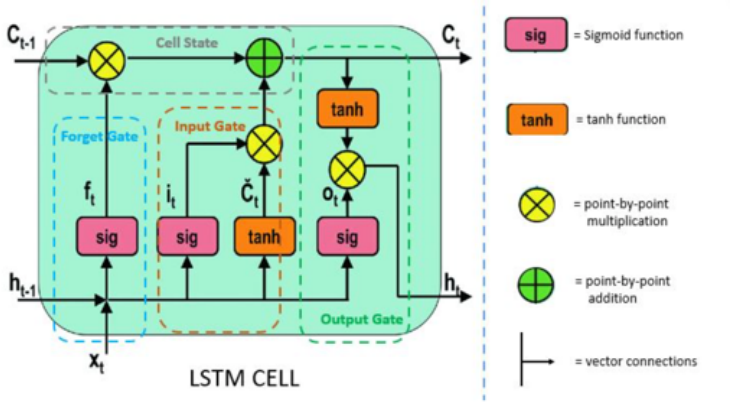


Figure 4.7 LSTM cell

The forget gate decides which information needs attention and which can be ignored. The information from the current input $X(t)$ and hidden state $h(t-1)$ are passed through the sigmoid function. The input gate performs the following operations to update the cell status. The output gate determines the value of the next hidden state. This state contains information on previous inputs.

The regular LSTM takes input in one direction, either backward or forward. By combining two independent LSTM together, we obtain the bidirectional LSTM (Bi-LSTM), which can take the input from both directions. With this property, Bi-LSTM is able to preserve both the future and the past information and increase the available information to the network.

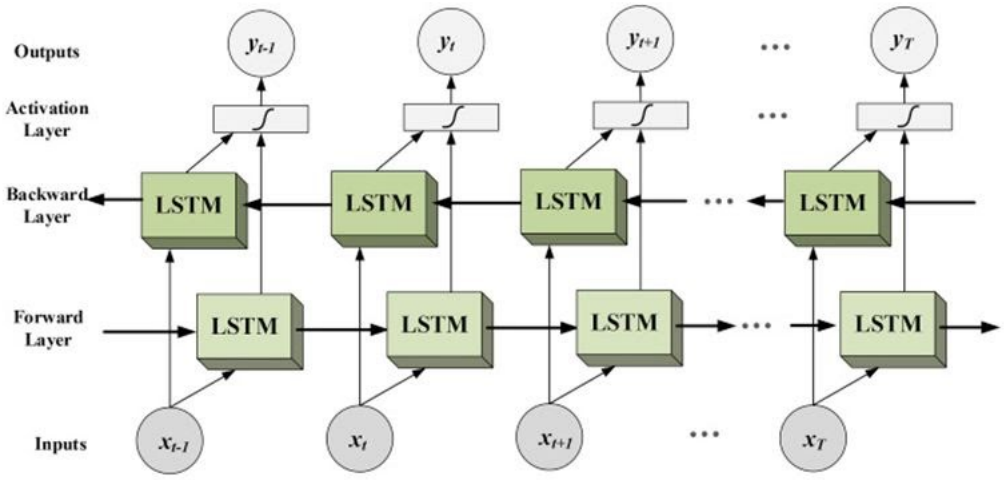


Figure 4.8 Structure of Bi-LSTM

Like CNN, the LSTM model also has several hyperparameters that need to be tuned, including several nodes and hidden layers, dropout, weight initialization, learning rate, and the choice of the activation function.

The Bi-LSTM model we finalized is shown in Figure 4.9. Its input layer also takes a 50 by 144 matrix as a sample of 24 hours. Two Bi-LSTM layers, each with 128 units each followed by a dropout layer of 0.1 to prevent overfitting. The dense layer contains 256 neurons and passes the output to an output layer of 50 units. The tanh function is selected as the activation function for this model.

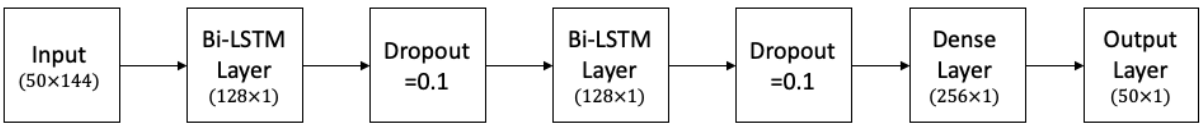


Figure 4.9 LSTM model

Sensor Bias Estimation and Flow Correction Algorithm

Algorithm 4.2:

- **Training data preparation:**

Step 1: Repeat Algorithm 4.1 until we collect sufficient training samples. Note that, our models need historical traffic measurement data serving as the ground truth. In practical, the true traffic flow volume Z_a is not observable. Instead, we assume for the ground truth data, the observation $V_a \approx Z_a$. In this study however, since we don't have the ground truth data available for the real-world network, we are using SUMO network and the simulated traffic flow for the below experiments. After Step 1, we will collect enough training data with manipulated biased sensor readings and the corresponding true biases β.

- **Model training:**

Step 2: Train and cross-validate the deep learning models using the manipulated training dataset. Use the corresponding true biases to compute the prediction loss.

Step 3: Tune the model parameters: number of nodes for each layer, number of layers, dropouts, etc. for best cross-validating results.

- **Bias estimation for unknown biased dataset:**

Step 4: Collect the traffic sensor readings with unknown biases that we want to estimate and format the data into one-day observations each serving as one sample.

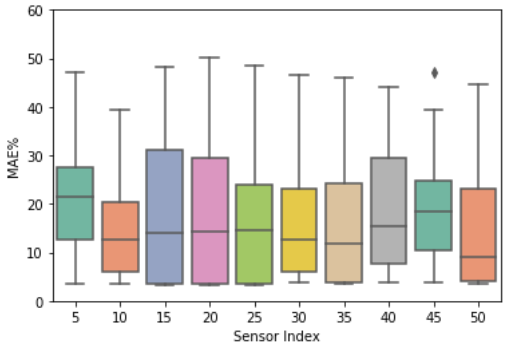
Step 5: Use the PIDL models to estimate the biases for each sample and use the mean of each estimation as the estimated measurement biases $\hat{\beta}$ for the sensors.

Step 6: Compute the corrected traffic flow volume for n^{th} observation of a^{th} sensor by $\tilde{f}^n_{crrt,a} = V^n_a / \hat{\beta}^n_a$.

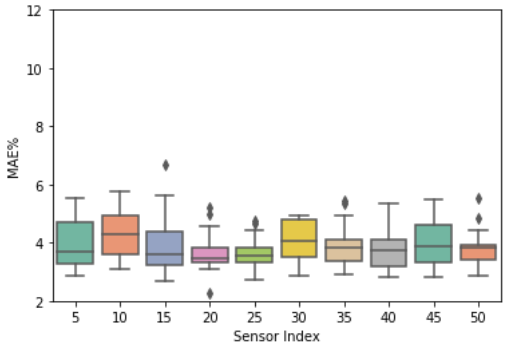
4.3 Results and Discussions

4.3.1 Models Performance

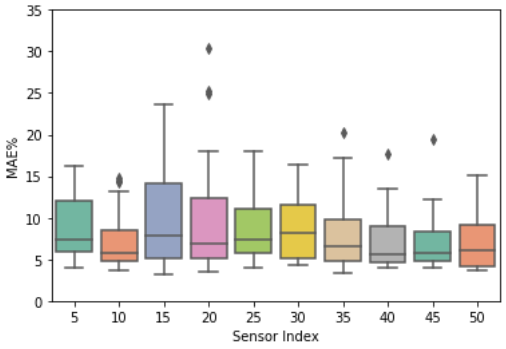
With random error ratio $\sigma_a = 0.02$, and the percentage of healthy sensors $p_h = 0.2$, we first compare the performance of the three neural network models. Being trained and validated with the training data, the MAE of models on the test data (8760 observations of the 50 sensors in the network) were obtained, and the boxplot of 10 sensor results is shown in Figure 4.10.



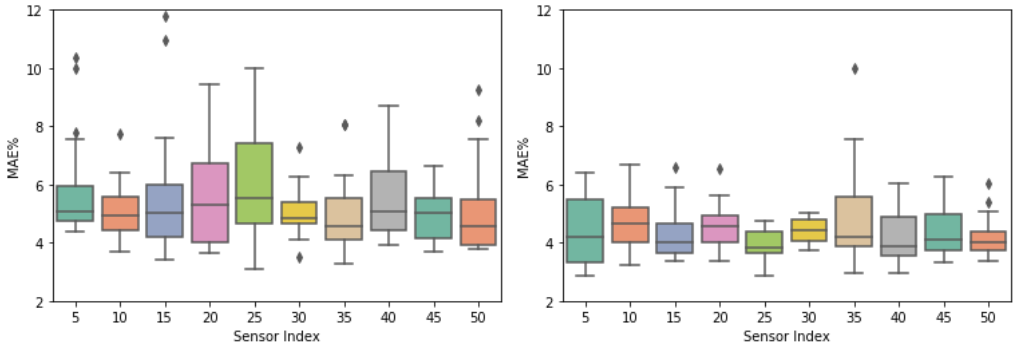
(a) Before correction



(b) Corrected by true biases



(c) Fully connected



(d) CNN (e) LSTM

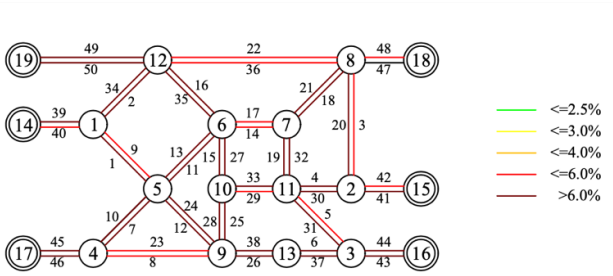
Figure 4.10 *MAE* before and after model corrections

Figure 4.10(a) shows the *MAE* before model correction, which is calculated based on formulas (4.12) and (4.13) with $\hat{\beta}^s \equiv 1$. Figure 4.10(b) shows the *MAE* corrected by the true sensor biases, namely replacing $\hat{\beta}^s$ in (4.13) with β^s. Figure 4.10 (c), (d), and (e) are the results of the three different neural network models. We can see that, after correction, the MAE drops from 15.8% to 7.6%, 5.1%, and 4.5%, respectively, for the fully-connected neural network, CNN and LSTM. Both CNN and LSTM models perform well, given that the theoretically best MAE corrected by the actual sensor biases is 3.8%. LSTM's results appear to have a relatively smaller variance than CNN.

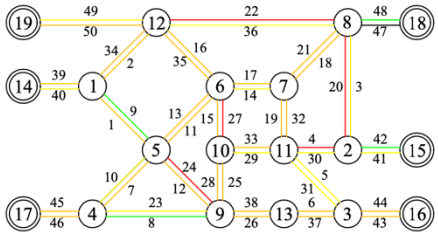
Overall, the above results show the capability of neural networks in dealing with time series traffic data. The simplest fully connected neural network is able to learn some patterns of sensor observations and reduce their *MAE*. The more complex neural network models like CNN and LSTM appear to capture more features embedded in the data and further improve the measurement quality.

Position of links

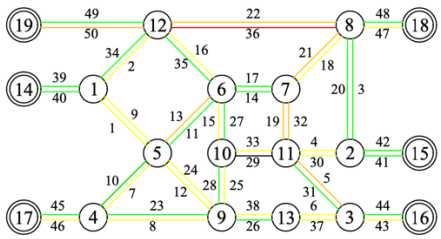
To illustrate the estimation performance for each individual sensor, we use the colored 'maps' of the network to present the *MAE*s for each sensor. Since each link has only one sensor, the color of each link represents its *MAE* level.



(a) Fully connected



(b) CNN



(c) LSTM

Figure 4.11 Performance of Different Models

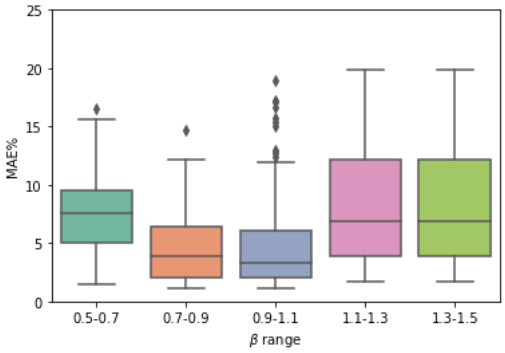
We can see that from these graphs, overall, LSTM performs best, followed by CNN, then the fully-connected neural network. The fully connected neural network gives *MAE*s over 6% for more than half of the sensors, while LSTM is able to keep the majority of *MAE*s close to 4%. But on the other hand, we don't observe a significant pattern that the position of a link in the traffic network affects

the estimation accuracy of its sensor bias, namely, the degrees of the nodes to which the links connect do not appear to impact the model performance.

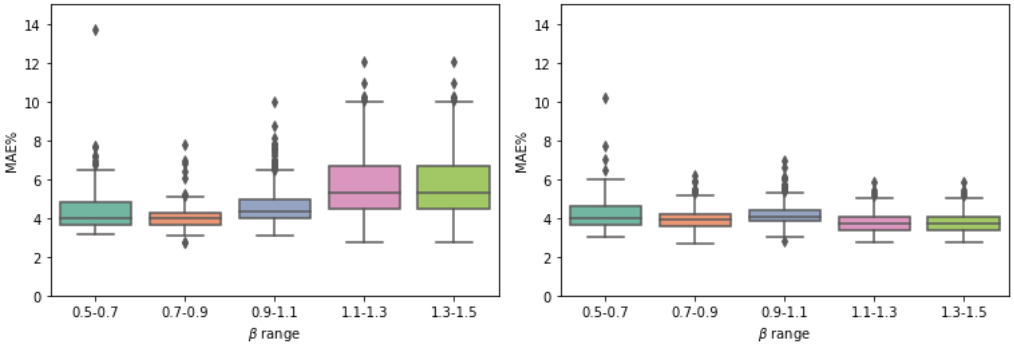
Scale of measurement biases

Intuitively, we can imagine that in a traffic network where not all sensors are fully calibrated, different sensors may have different measurement biases. It is worthwhile to see how the models react to varying scales of measurement biases. For instance, is the model performance decreasing with the increase of measurement biases? Is the variance still acceptable for the more significant biases? To perform this test, we repeat the above initial case 20 times (so that we randomly generate different biases β for each sensor) and aggregate the MAEs into five groups based on the scale of their corresponding sensor biases. Because the above results show that the MAE is insensitive to the locations of the links, the MAEs of all 50 sensors are all assigned to the groups solely based on β.

Figure 4.12 shows how MAEs of different models change concerning the scales of measurement biases. The five groups of data correspond to β of [0.5, 0.7), [0.7, 0.9), [0.9, 1.1], (1.1, 1.3] and (1.3.1.5]. Note that the farther away from 1 of β, the larger the sensor bias is. We can see from Figure 4.12(a) that the fully connected neural network model performs intuitively. Both the average MAE and its variance increase with the sensor bias level. The estimation results become less reliable when β is too far away from 1. For CNN and LSTM, the MAEs are better controlled. Larger β does have a slight impact on CNN by causing the variance to increase moderately. But on the other hand, CNN is less sensitive to smaller β. In terms of LSTM, it shows the robust performance for any β within the test range.



(a) Fully connected



(b) CNN (c) LSTM

Figure 4.12 Effect of bias scale on model performance

4.3.2 Sensitivity analysis

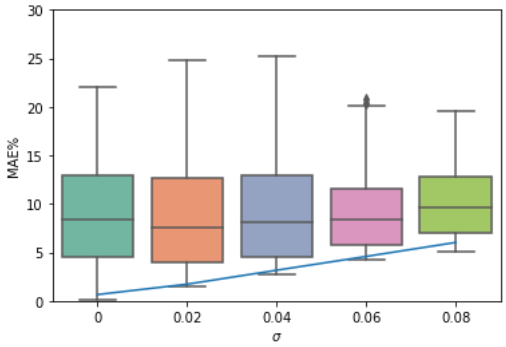
In the above part, we have introduced the randomness to both traffic flow and observations to expose the models to broader possible patterns. But the parameters of the randomness remain unchanged. For instance, we keep random error ratio $\sigma_a = 0.02$, and the probability of healthy sensors $p_h = 0.2$. The base pattern used to generate traffic flow also remains the same. In this section, however, we are going to change these parameters further to test the model's robustness under various circumstances.

Random error variance σ_a^2

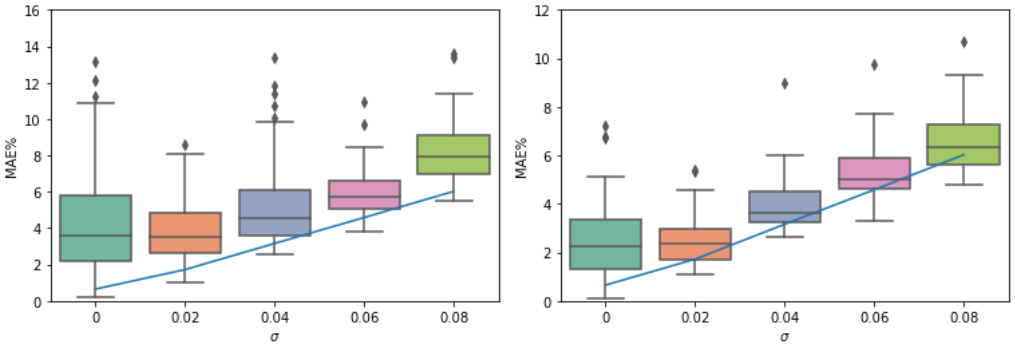
Recall that σ_a^2 is the variance of the white noise during the measurement for sensor a, which is not time-dependent and also independent of measurement bias β_a. In all above tests, we have fixed $\sigma_a^2 = 0.04$, while in this part, we are going to check the models against different σ_a^2 numbers (0, 0.02, 0.04, 0.06, and 0.08). For each σ_a^2, we run the training-testing process 10 times and plot the MAEs of all the observations for all the sensors in Figure 4.13. The blue lines in the graphs are the theoretically lower bound of MAEs given the true $\beta_a$values. Clearly, the blue line rises with the increase of σ_a^2, because according to Equation (4.14) derived from (4.12) and (4.13), we can see that the best possible MAE increases with the variance of random error ε_i^s.

$$\text{Lower bound } MAE = \frac{1}{ns}\sum_{i=1}^{n}\sum_{a=1}^{s}\left|\frac{(\beta_i^a v_i^a + \varepsilon_i^s)/\,\beta_i^a - f_i^a}{f_i^a}\right| = \frac{1}{ns}\sum_{i=1}^{n}\sum_{a=1}^{s}\left|\frac{\varepsilon_i^s/\,\beta_i^a}{f_i^a}\right| \quad (4.14)$$

As can be seen from Figure 4.13, the fully connected neural network is relatively insensitive to σ_a^2. However, this does not indicate that this model handles the larger σ_a^2 well. On the contrary, the fully connected neural network performs poorly even with small or zero σ_a^2. The MAEs of CNN and LSTM model results do increase with larger σ_a^2. But we notice that the gaps between their mean MAEs and the blue lines are narrowing, and the MAE variances are not increasing much. With $\sigma_a^2 = 0.08$, the mean MAEs of CNN and LSTM models are 7.5% and 6.2%, respectively, which are both quite acceptable.



(a) Fully connected



(b) CNN (c) LSTM

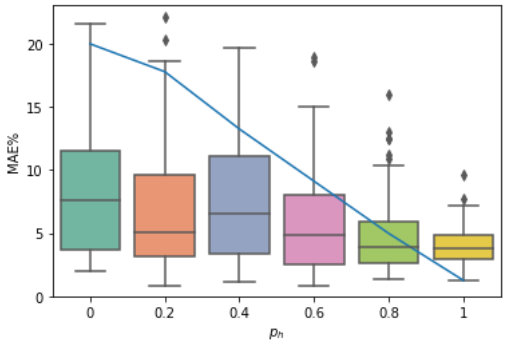
Figure 4.13 MAE corresponding to various levels of σ_a

The percentage of healthy sensors p_h

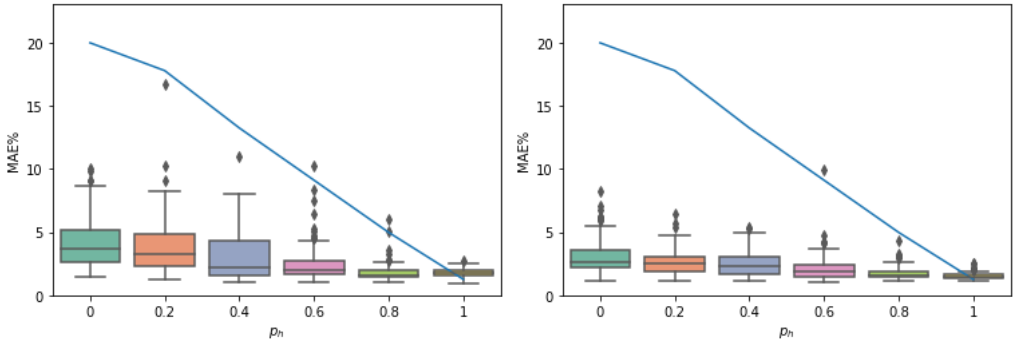
Initially, we assume that 20% of the sensors in this modified Nguyen-Dupuis network are healthy ($\beta = 1$), namely, $p_h = 0.2$. Now we are going to see how models perform with different numbers of p_h. We repeat the experiment 10 times for each of $p_h = 0, 0.2, 0.4, 0.6, 0.8$ and 1.0 and show the results in Figure 4.14.

The blue lines in Figure 4.14 represent the average MAE of observed traffic flow before model correction, so it decreases when more healthy sensors are present in the network. Similar to the

previous section, the fully connected neural network is not sensitive to p_h but gives worse flow corrections than CNN and LSTM, given any number of healthy sensors. Although when there are fewer healthy sensors, the model corrected MAE is better than the uncorrected ones, when more healthy sensors are present, the corrected MAE does not improve. When most of the sensors in the network have no measurement biases, the corrected MAE is even worse than the uncorrected ones. On the contrary, CNN and LSTM have good performance. Though their MAEs increase slightly when fewer healthy sensors are present, given no healthy sensor in the network, their MAEs are still well controlled, with LSTM performing slightly better.



(a) Fully connected



(b) CNN (c) LSTM

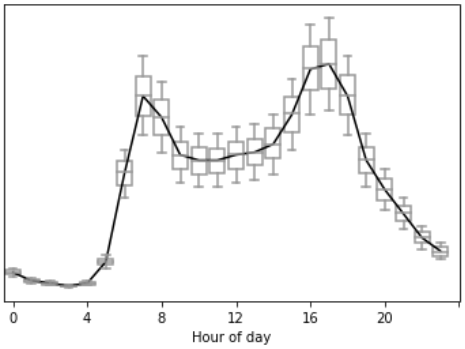
Figure 4.14 Different levels of p_h

Effect of flow imbalance

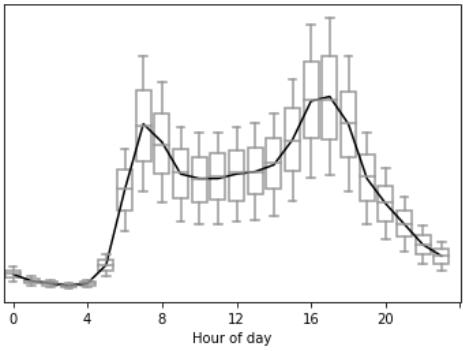
In the author's previous work, as discussed in Chapter 3, the sensor error estimation GMM model is built upon the flow conservation law, which however, can be violated when traffic conditions like shockwaves or queues are present. Since SUMO is a dynamic traffic simulation model, we can easily obtain the traffic volume at each sensor at any time within the simulation time range and compute the level of flow imbalance. Because the traffic observations in this experiment are aggregated at the 10-min level, the flow imbalance rate is also calculated for each 10-min observation. For any node a in the network, the flow imbalance rate for any t^{th} 10-min observation data is calculated following:

$$Imbalance\ ratio_a = (|\mathcal{A}^+(i)| + |\mathcal{A}^-(i)|) \times \frac{\sum_{a\in\mathcal{A}^+(i)} Z_a^t - \sum_{a\in\mathcal{A}^-(i)} Z_a^t}{\sum_{a\in\mathcal{A}^+(i)} Z_a^t + \sum_{a\in\mathcal{A}^-(i)} Z_a^t} \times 100\% \quad (4.15)$$

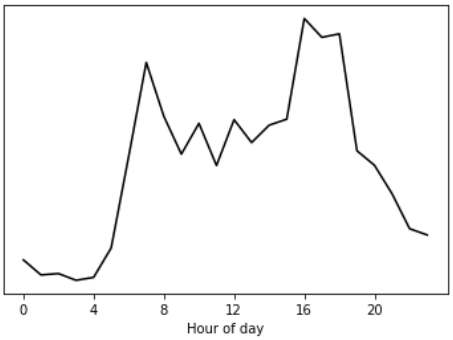
Since the traffic shockwave can be caused by the unstable traffic flow, we can manually increase the chance of flow imbalance by increasing the traffic flow fluctuation throughout the day. Initially, for all previous tests, we use SUMO simulation data with the hourly flow pattern, as shown in Figure 4.15(a). After we increase the variance of the flow pattern randomness as shown in Figure 4.15(b), we are more likely to obtain more significant flow fluctuations, e.g., the cases in Figure 4.15(c) and Figure 4.15(d).



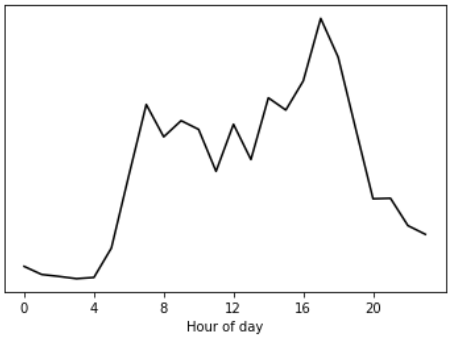
(a) Initial flow pattern base



(b) flow pattern base with larger variance



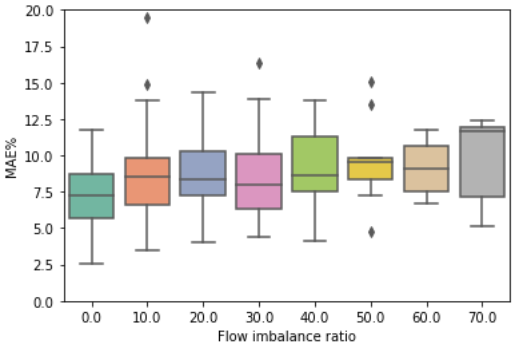
(c) A sample of randomized flow pattern



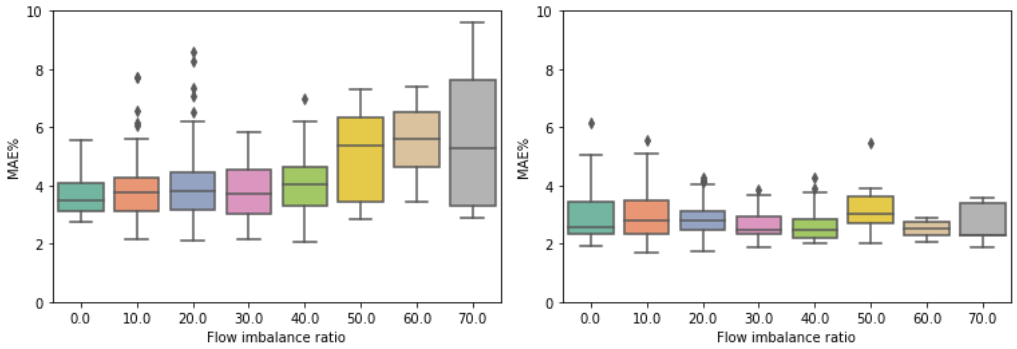
(d) A sample of randomized flow pattern

Figure 4.15 Hourly flow patterns of a day

With this change, after randomly repeating the simulation 50 times, we are able to observe a broader range of flow imbalance rates (from 0% to 70%+). Training and testing the data using our models, we obtain one MAE and one flow imbalance rate for each 10-min observation data of one node. Grouping MAEs based on different levels of flow imbalance rate (0%-10%, 10%-20%, 20%-30%, 30%-40%, 40%-50%, 50%-60%, 60%-70%, and 70%-80%), we obtain the boxplots below.



(a) Fully connected



(b) CNN (c) LSTM

Figure 4.16 Effect of flow imbalance on model performance

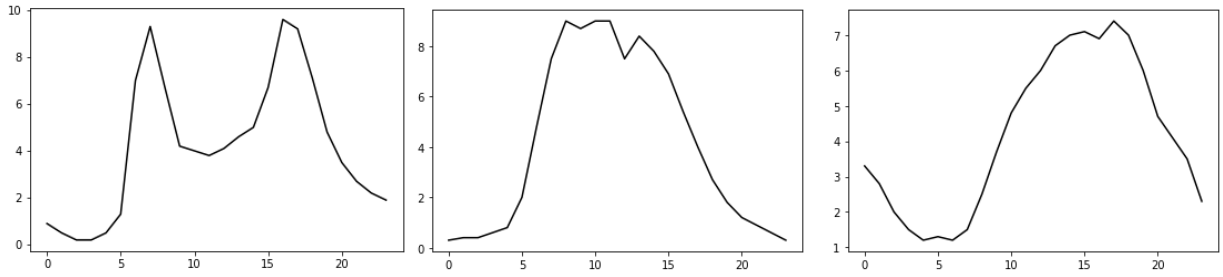
Figure 4.16 shows that all three models show relatively consistent performance when there is no significant flow imbalance (<=50%). CNN's performance worsens when a greater imbalance ratio presents while LSTM appears to well handle most imbalanced flow cases in this experiment. From this test, we may infer that the fully connected neural network model and CNN model still generalize certain levels of flow conservation law during training and use this law to predict new data. Therefore, when the training data frequently breaks the conservation rule, the model may struggle to fully capture the complete pattern. On the other hand, LSTM is almost not affected by the flow imbalance. The reason could be that, since LSTM is excellent at extracting features from

the temporal dimension, some additional information in this dimension well compensates for the missing flow conservation law.

Change in traffic flow pattern

In the previous section, we increase the variance of randomness when using the flow pattern base to generate simulation data. However, the 'shape' of the pattern base does not change. In reality, we know that in some short-term cases, for instance, when there is severe weather or road work, the pattern base may vary significantly. Thus, it is also necessary to examine the robustness of the models with respect to flow pattern changes.

In this test, the models are still trained using the initial flow pattern base. But the test data are generated separately from different flow patterns to imitate the road condition change. Three different flow patterns (Figure 4.17) are used for this experiment. The model performance is shown in Figure 4.18.

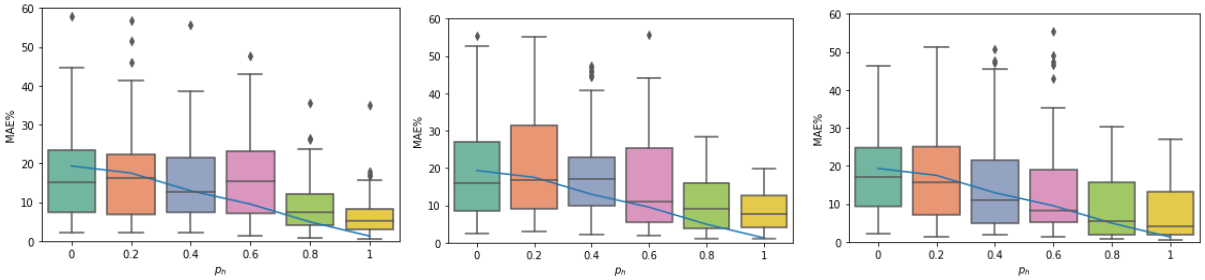


(a) Pattern 1 (b) Pattern 2 (c) Pattern 3

Figure 4.17 Different test flow patterns

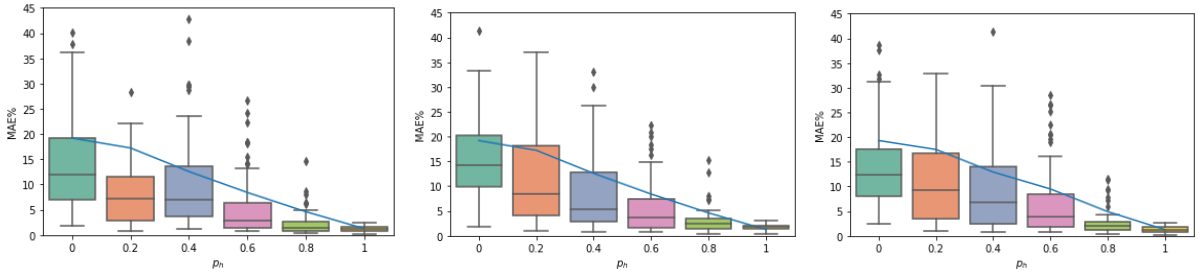
From Figure 4.18, we can find that the fully connected neural network is not able to provide reasonable prediction to the testing data, when there are significant traffic flow pattern changes. In terms of CNN and LSTM, their performance depends on the percentage of healthy sensors.

With few or no healthy sensors available, they are also struggling to give a good prediction for the test data generated from new flow patterns. Nevertheless, if there are sufficient (e.g., $p_h > 40\%$) healthy sensors in the network, the additional information provided by these sensor observations significantly improves the models' estimation accuracy. In most all the tests above, the LSTM model slightly outperforms CNN, while in this experiment, CNN corrects the traffic flow more accurately than LSTM, when the portion of healthy sensors is higher. One hypothesis of this difference could be that, as previously discussed, the CNN model relies more on flow conservation law than LSTM. So when there are more correct measurements, CNN can gain help from the flow conservation law to better predict the unfamiliar data. In reality, for instance, in PeMS traffic network, more than half of the sensors are considered to be in good working condition, namely $p_h \geq 50\%$. Under this situation, the traffic flow correction ability of both CNN and LSTM models for short-term traffic pattern change is approved.



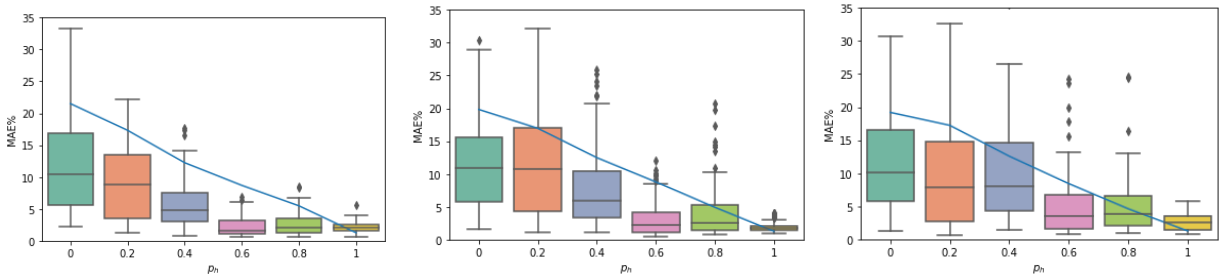
(i) Pattern 1 (ii) Pattern 2 (iii) Pattern 3

(a) Fully connected



(i) Pattern 1 (ii) Pattern 2 (iii) Pattern 3

(b) CNN



(i) Pattern 1 (ii) Pattern 2 (iii) Pattern 3

(c) LSTM

Figure 4.18 Effect of flow pattern changes on model performance

4.3.3 Discussions

In this study, we developed three Physics-Informed Deep Learning models, based on the fully connected neural network, CNN, and LSTM, to estimate the measurement biases for traffic network sensors and recover the erroneous traffic volume readings. The domain knowledge about the sensor observation data is passed to the models via the design of training data and input layer structure. During SUMO simulation, the Origin-Destination demand pairs are prepared so that the simulated traffic flow data can comprehensively represent the non-linear traffic assignments. The dimension and the structure of the input data also reflect the spatial-temporal relationship embedded in the sensor observations.

After training with the same training data, the three models show different MAE correction abilities. In general, the fully connected neural network performs the worst. With the simpler model structure, the fully connected neural network model is not capable of fully utilizing the spatiotemporal information embedded in the training data, even with the help of the PIDL design. Under the initial setting, the model can improve the MAE from 17% to under 10%, but the performance is not robust and consistent, and the generalization ability is also weak. With the

improvement in the overall sensor conditions in the network (e.g., more minor white noises, more healthy sensors), the MAEs still do not decrease. On the contrary, the PIDL models based on CNN and LSTM estimate the sensor biases much better, with LSTM-based slightly outperforming CNN-based. The nature of the structures of CNN and LSTM models allows the models to better exploit the correlations in both spatial and temporal dimensions. With the same initial setting, these two models are able to reduce the MAEs to under 5%. Their estimation performance is also very consistent, and relatively insensitive to factors including the sensor (link) positions in the network, the scale of sensor measurement biases, and the variance of white noise. They are also benefitting from the healthy sensors present in the network.

With the help of more correct observations from the healthy sensors, the models can significantly reduce both the estimation errors and the error variance. On the other hand, we observe that the models' performance does not heavily rely on the flow conservation law. Especially for the LSTM-based PIDL model, the presence of heave flow imbalance does not obviously impact its flow correction accuracy. This indicates that a more complicated spatial-temporal correlation other than the flow-conservation law is observed and learned by the PIDL model. Lastly, the CNN-based and LSTM-based PIDL models demonstrated the ability to estimate the measurement biases under short-term traffic condition changes. When the traffic flow pattern changes, the models can gain help from healthy sensor readings. Albeit the estimation correctness decreases, the significant improvements in MAEs compared to the untreated data still help traffic management better understand the truth.

Chapter 5 Single Loop Detector Measurements Bias Correction Using Deep Learning

5.1 Introduction

In Chapter 4, we studied sensor biases in traffic flow measurement. In transportation engineering, macroscopic traffic models often involve not only traffic flow but also density. The traffic flow rate can be derived from the traffic count given by the single loop detector system. Though a single loop detector does not measure traffic density directly, its traffic occupancy reading is often used to infer the traffic density. Research has been focusing on the study of traffic volume measurement, but the health of traffic occupancy data is rarely discussed. This part of the dissertation will try to fill this gap by examining traffic volume and traffic occupancy measurements of the loop detector sensor. To align our study object with the macroscopic traffic models, we narrow our scope from the entire network down to a traffic segment.

Introducing the traffic flow ordinary differentiate equation (ODE) model which draws a relation between the traffic flux (volume) and traffic density (occupancy), we are utilizing the time attributes of the traffic observations so that taking advantage of the additional information that may be contained in the traffic flow dynamics. Incorporating the time attribute, we will be looking at multi-dimensional time-series data containing complex non-linear feature information. Conventional analytical models, if not impossible, would be very difficult to deal with this situation. Therefore in this study, we will take advantage of the neural network models informed by physics to extract high-level features from traffic data and learn a non-parametric mapping to predict measurement biases in the sensor readings.

5.2 Methodology

5.2.1 Traffic Measurement for a Road Segment

A typical inductive loop traffic detector is usually installed on the road pavement. It generates an electronic signal when a vehicle passes over it. Figure 5.1 shows some examples of the electronic signal of certain types of vehicles.

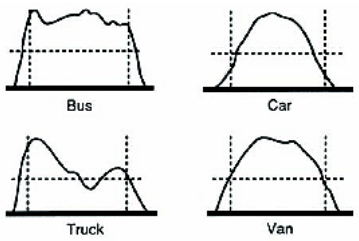


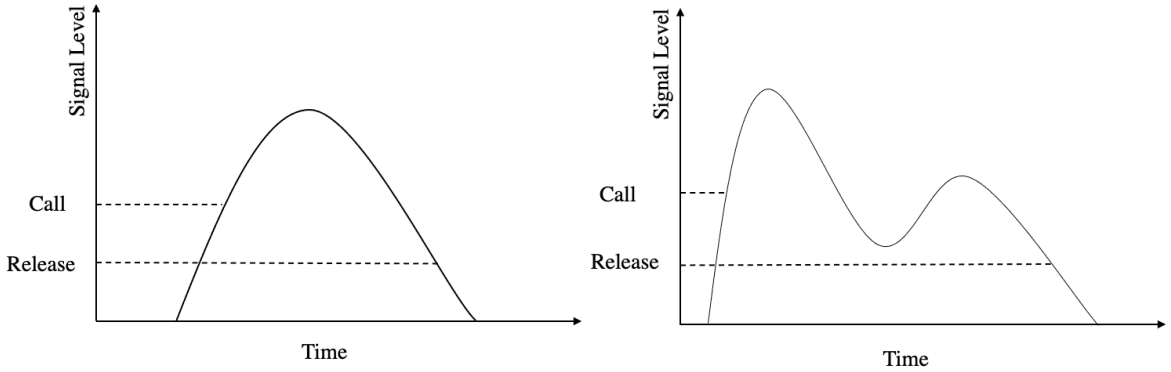
Figure 5.1 Representative vehicle signatures obtained from loops (US DOT, 2006)

After processing the raw electronic signals for inductive loop sensors, traffic managements are able to estimate the macroscopic variables of the measured traffic flow, such as traffic flow, density, and speed.

Traffic volume (flux) measurement:

Suppose that when a small vehicle passes over a certain sensor, the sensor outputs a signal as shown in Figure 5.2a. For each sensor, we need first to define two thresholds: call and release (US DOT, 2006). The call threshold means that only after the signal passes this threshold, do we start to recognize this signal as a vehicle count. The vehicle presence signal is terminated after the signal drops below the release threshold. The two thresholds can be and are often time different, with the release lower than the call. Doing so can effectively reduce measurement errors for larger vehicles

like trucks. Suppose that Figure 5.2b represents a typical truck signal which usually consists of two peaks. In this case, if the release equals the call, the signal would trigger the call-release twice, and the sensor ends up double counting this truck. But separating the two thresholds can alleviate this issue.



(a) electric signal for a car (b) electric signal for a truck

Figure 5.2 Typical electric signal by inductive loop sensor

Both thresholds for each individual sensor need to be calibrated to ensure the sensor gives the most accurate measurements. However, several factors may still lead to the measurement deviation from the actual traffic volume. For example:

Tailgating: the tailgating may result in a very similar output signal to that of the truck. As a consequence, the sensor may undercount the traffic flux.

Sensor aging: A sensor may become less sensitive and output a weaker signal when it ages. When the signal valley of the trucks starts dropping below the release threshold, the sensor may over-count the traffic volume.

Other possible causes of sensor measurement errors include change of vehicle patterns, weather, etc. Overall, the sensor working condition is subject to several uncertainties, and as discussed in the previous chapters, the accuracy of the sensor readings is not always guaranteed.

Traffic density measurement:

Inductive loop sensors directly measure the traffic occupancy, from which the traffic density is estimated. Traffic occupancy is defined as the percentage of time the sensor is occupied by a vehicle. Similar to the measurement for traffic flux, we can set a call and a release threshold for the sensor signal. Any signal that is within a call-release time window is considered to be "occupied", so the traffic occupancy is the "occupied" time divided by the entire measurement time (US DOT, 2006). Both traffic flux and occupancy are directly deduced from the electronic sensor signal. But due to different measuring processes (presence mode for traffic occupancy, pulse mode for traffic flux) and different threshold setups, the two measurements are not entirely dependent; each can supplement additional information.

Once the traffic occupancy (denoted as r) is determined, the traffic density (k) can be derived based on:

$$k = \frac{r}{\overline{l_v}} \tag{5.1}$$

Where $\overline{l_v}$ is the estimated average length of the measured vehicles.

Fundamental diagram

As a macroscopic traffic model, the fundamental diagrams studied the relation between the three aggregated traffic flow characters, density (k), flux (f), and average speed (v). More specifically, the fundamental diagrams depict the three pair-wise relationships speed-density, flow-density, and speed-flux. Here, the flux is the number of vehicles passing a specific point within a certain time frame. The density is the number of vehicles showing at a certain length of a road segment at a certain moment. The speed is the space mean speed (v_s), the average speed of vehicles showing at the road segment at the certain moment.

Even if there appear to be three pair-wise relationships, because of the fundamental relation between flux, density, and mean speed,

$$f = k \times v_s \tag{5.2}$$

The three pairs are not independent and any two of them can be deduced if the other one is defined.

(Greenshields et al., 1935) initially defined the speed-density relation as,

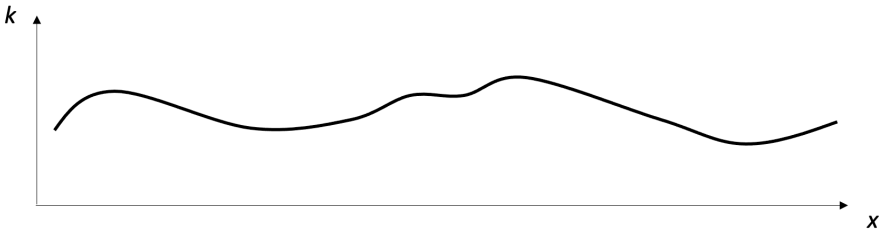
$$v_s = v_f \left(1 - \frac{k}{k_j}\right) \tag{5.3}$$

Where v_f is the free flow speed and k_j is the congested traffic density. Since then, there have been several studies looking into its improvement, and several variant models have been proposed, for instance, the earlier (Greenberg, 1959) model and (Underwood, 1961) model, and more recent (MacNicholas, 2008) model.

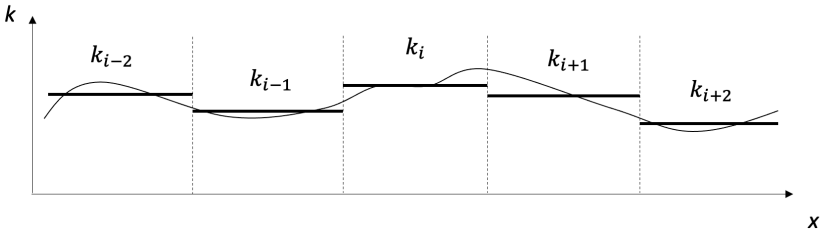
Since first proposed in 1935, the fundamental diagram greatly influenced the following traffic flow theory research work and still plays a vital role in current transportation engineering.

5.2.2 ODE Approximation

Suppose Figure 5.3a represents the traffic density-location graph for a piece of traffic road at certain time. According to the Godunov type numerical scheme (Lebacque, 1996), the road can be spatially discretized into smaller, finite segments (Figure 5.3b).



(a) Traffic density-location graph



(b) Discretization of density-location graph

Figure 5.3 Traffic density-location graph and discretization

For each local segment, $k(x,t)$ is simplified using the average density $k(t)$, which does not involve the space domain. Therefore, the LWR can be approximated using the Ordinary

Differential Equation (ODE). For a local segment, as shown in Figure 5.3, based on the local conservation law (Agarwal et al., 2016), we can derive the ODE model as,

$$\dot{k}(t) \equiv \frac{d\,k(t)}{dt} = \frac{f^{in}(t) - f^{out}(t)}{l} \tag{5.4}$$

Where $k(t)$ is the average traffic density for this l length segment at time t. $f^{in}(t)$ and $f^{out}(t)$ are the simultaneous incoming and outgoing traffic flux. From this equation, we can clearly see that, if $\dot{k}(t)$ is nonzero, $f^{in}(t) \neq f^{out}(t)$.

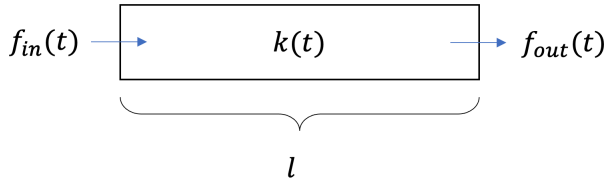


Figure 5.4 ODE model for a road segment

Figure 5.5 is a simplification of a segment of continuous traffic road with neither entrance or exit. The traffic road can have multiple lanes, but the multiple sensors across a road section are aggregated into a single one. Therefore, eventually the l length road segment only has two aggregated sensors: one upstream and one downstream.

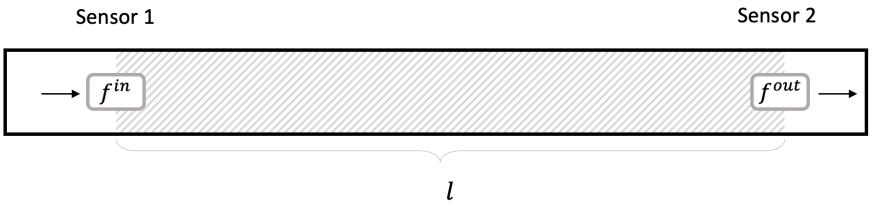


Figure 5.5 Simplified Road Segment

Let's assume that at time t, the traffic flux at sensor 1 is $f^{in}(t)$ and the traffic flux at sensor 2 is $f^{out}(t)$, and the average traffic density over the segment between sensor 1 and sensor 2 is $k(t)$. Then after combining the ODE traffic model (5.4) with the density-occupancy relation (5.1) we obtain:

$$\dot{r}(t) \equiv \frac{d\,\bar{l}_v k(t)}{dt} = \frac{\bar{l}_v(t)}{l}\left(f^{in}(t) - f^{out}(t)\right) \tag{5.5}$$

It is found that the average vehicle lengths vary a lot for different times and different locations (Jia et al., 2001). However, in this study, we will assume $\overline{l_v^{in}}(t) = \overline{l_v^{out}}(t)$. Because: 1. The average vehicle length is not our study object and when we choose the input data for our model, we can only use the measurement samples with relatively constant average vehicle length; 2. In our study scope, since we are looking at two adjacent sensors that are often installed closely, we can assume that the difference between the average vehicle lengths at both sensor locations during the same time is negligible. Therefore, Equation (5.5) discloses the relationship between the traffic flux and occupancy and offers us the opportunities to cross-check their validity and improve the overall characterization of the traffic flow.

5.2.3 Traffic Measurements Error Model

We assume the measurement of traffic flux has systematic error following the equation in Chapter 2, namely,

$$\widetilde{f^{in}} = \mu_f^{in} f^{in} + \varepsilon_f^{in} \tag{5.6}$$

$$\widetilde{f^{out}} = \mu_f^{out} f^{out} + \varepsilon_f^{out} \tag{5.7}$$

Similarly, for traffic occupancy measurements, we have,

$$\widetilde{r^{in}} = \mu_r^{in} r^{in} + \varepsilon_r^{in} \tag{5.8}$$

$$\widetilde{r^{out}} = \mu_r^{out} r^{out} + \varepsilon_r^{out} \tag{5.9}$$

But notice that $r(t)$ in (5.11) is the link-level occupancy measurement, which is different from the point measurement r^{in} and r^{out}. Combining (5.5), (5.6) and (5.7), and let $\beta_f = \frac{1}{\mu_f}$, we obtain,

$$\tilde{\bar{r}} = \frac{\bar{l_v}}{l}\left(\beta_f^{in}\widetilde{f^{in}} - \beta_f^{out}\widetilde{f^{out}}\right) + \frac{\bar{l_v}}{l}\left(\beta_f^{in}\varepsilon^{in} - \beta_f^{out}\varepsilon^{out}\right) \tag{5.10}$$

Here, because we don't know yet the relationship between the link-level occupancy measurement and $\widetilde{r^{in}}$, $\widetilde{r^{out}}$, we still use $\tilde{\bar{r}}$ on the LHS of the equation temporarily.

The expectation of the right half of the RHS is zero, namely,

$$E\left(\frac{\bar{l_v}}{l}\left(\beta_f^{in}\varepsilon^{in} - \beta_f^{out}\varepsilon^{out}\right)\right) = 0 \tag{5.11}$$

This model (5.10) resembles an ordinary least square (OLS) model which potentially offers the chance to estimate β_f^{in} and β_f^{out} (μ_f^{in} and μ_f^{out}) when we have enough observations of $\tilde{\bar{r}}$, $\widetilde{f^{in}}$ and $\widetilde{f^{out}}$. But there are two obvious obstacles that make the estimation non-trivial: 1. We are still missing the link-level traffic occupancy measurement and the traffic occupancy measurement itself is not unbiased. 2. The traffic flow is not continuous flow so instantaneous measurements (e.g., measurements of each 5s or 10s) may not comply with this macro-level equation. We'll start looking at the latter one.

Discontinuity of traffic flow: Different from the fluid flow, the traffic flow is discontinuous. The LHS of equation (5.5) looks at how traffic density varies over time. Overall, a reasonable estimation $\dot{r}(t) \approx \frac{r(t+\Delta t)-r(t)}{\Delta t}$ needs the Δt to be small. However, when Δt is small, the traffic flux measurement for that small interval is too discrete to providing any useful information to this model. For instance, let us assume that $\Delta t = 1s$ and if during a measuring interval a vehicle passes over the sensor, the "accurate" flux for this second would be 3600 vehicle per hour. On the other hand, if there is no vehicle appearing in that second, the flow rate would just be 0 vehicle per hour. Neither these two numbers along provides any valid information. This is due to the fact that, the term of traffic flux and the differential equation models are applicable to macroscopic level traffic flow and may not hold under the microscopic condition.

Missing observation of link-level traffic occupancy: As pointed out previously, the traffic occupancy appears at equation (5.5) is the link-level occupancy for the road segment, while in reality, we only have point occupancy measurements at its two edges. In general, $r(t) \neq r^{in}(t) \neq r^{out}(t)$. Even if we can find some situations with relatively constant traffic flux when $r(t) \approx r^{in}(t) \approx r^{out}(t)$, due to the existence of measurement biases (μ_r^{in} and μ_r^{out}), we still cannot assume $\tilde{r}(t) = \widetilde{r^{in}}(t) = \widetilde{r^{out}}(t)$.

The traffic flux is an aggregated attribute of the macroscopic level traffic flow while the traffic volume does apply to both macro- and microscopic levels. Assume that both $\dot{r}(t)$ and $f(t)$ are integrable in a time window $[t_1,\ t_2]$. We can take the integral on both sides of equation (5.5), from time t_1 to t_2, namely,

$$\int_{t_1}^{t_2} \dot{r}(t)dt = \int_{t_1}^{t_2} \frac{\overline{l_v}(t)}{l}\left(f^{in}(t) - f^{out}(t)\right)dt \tag{5.12}$$

Here, the average vehicle length $\overline{l_v}(t)$ is also time dependent. However, since we are choosing t_2 close to t_1 and we can always choose a proper $[t_1,\ t_2]$ interval with relatively constant $\overline{l_v}(t)$, we assume that we can drop the time dependence of $\overline{l_v}(t)$. Now, we can convert (5.12) into,

$$\int_{t_1}^{t_2} \dot{r}(t)dt = \frac{\overline{l_v}}{l}\int_{t_1}^{t_2} \left(f^{in}(t) - f^{out}(t)\right)dt \tag{5.13}$$

Which further implies:

$$\frac{l}{\overline{l_v}}\left(r(t_2) - r(t_1)\right) = \left(\widetilde{Q^{in}}(t_2) - Q^{in}(t_1)\right) - \left(Q^{out}(t_2) - Q^{out}(t_1)\right) \tag{5.14}$$

Here $Q(t)$ means the accumulative traffic volume from time 0 to t. This equation has a clear physical meaning. We can notice that its RHS means the overall difference between the incoming and outgoing traffic volume for this road segment during $[t_1,\ t_2]$. The LHS means the difference in numbers of vehicles within the l length road at time t_2 and t_1. Intuitively, they should be equal.

Similarly, we can convert (5.10) to,

$$\frac{l}{\overline{l_v}}\left(r(t_2) - r(t_1)\right) = \beta_f^{in}\left(\widetilde{Q^{in}}(t_2) - \widetilde{Q^{in}}(t_1)\right) - \beta_f^{out}\left(\widetilde{Q^{out}}(t_2) - \widetilde{Q^{out}}(t_1)\right) + \zeta \tag{5.15}$$

Where $\zeta = \frac{\overline{l_v}}{l}\left(\beta_f^{in}\varepsilon^{in} - \beta_f^{out}\varepsilon^{out}\right)(t_2 - t_1)$, because neither β nor ε is time dependent. Hence, ζ can still be treated as a random measurement error with $E(\zeta) = 0$. With this conversion, we no

longer need to estimate the changing rate of the traffic occupancy $\dot{r}(t)$ and no longer need to use the microscopic level traffic flux measurement.

Now the remaining part is to get an estimation of the link-level traffic occupancy using the two end-point measurements. We will explore their relationship with the help of the Lighthill-Whitham-Richards (LWR) model and the finite difference method idea.

Lighthill-Whitham-Richards (LWR) model

The macroscopic LWR model (Lighthill & Whitham, 1955) is a Partial Differential Equation (PDE) model commonly used to describe traffic flow. This model considers the traffic flow as a type of fluid flow so that it can model the traffic flow using the fluid dynamics method with the partial differential equation. With the assumption that vehicle conservation follows the dynamic continuity equation, the LWR model describes the one-dimensional traffic flow at location x and time t as:

$$\frac{\partial k(x,t)}{\partial t} + \frac{\partial f(x,t)}{\partial x} = 0 \tag{5.16}$$

Where $k(t,x)$ is the traffic density and $f(t,x)$ is the traffic flux at location x and time t. Combining the LWR model with the static hydrodynamic relation $f = kv = f(k)$, we obtain

$$\frac{\partial k(x,t)}{\partial t} + \frac{\partial f(k(x,t))}{\partial x} = 0 \tag{5.17}$$

The LWR model describes the traffic flow in the space-time domain and the solution to this model is time consuming even with appropriate schemes (Herty et al., 2007). Hence, based on different objectives, the LWR model can be approximated using different approaches.

Numerical approximation of LWR model

There are several schemes we can use to approximate the LWR model, for example, the finite difference method. Its basic idea is to use the finite difference to approximate the derivative, for instance, using the right-hand side of the below equation to estimate the left-hand side.

$$\dot{k}(t) \equiv f'(t) \approx \frac{f(t+\Delta t) - f(t)}{\Delta t} \tag{5.18}$$

For the LWR model, we will first discretize it in both space and time domain. For instance, for a L length road from time 0 to T, we can discretize it into grids each with Δx in length and Δt in time (Figure 5.6).

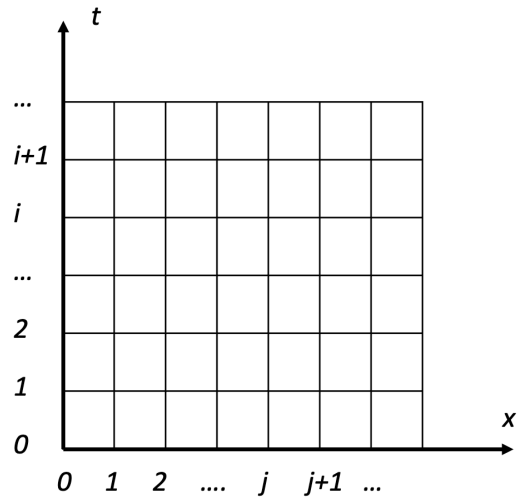


Figure 5.6 Spatial and temporal discretization

Then we can estimate $\frac{\partial k}{\partial t}$ and $\frac{\partial k}{\partial x}$ by,

$$\frac{\partial k(x,t)}{\partial t} \approx \frac{k(x,t+\Delta t) - k(x,t)}{\Delta t} \tag{5.19}$$

$$\frac{\partial k(x,t)}{\partial x} \approx \frac{k(x,t) - k(x-\Delta x,t)}{\Delta x} \tag{5.20}$$

To estimate the link-level occupancy for the l length segment at time t_1 and t_2, we are going to first define our study scope: $[0,\ l]$ in space dimension, $[t_1',\ t_2]$ in temporal dimension. Here, we want to select a $t_1' \leq t_1$and more commonly $t_1' < t_1$. The reason will be discussed later. Now we discretize the space dimension into L equal segments and the temporal dimension into T grids, therefore $\Delta t = \frac{t_2 - t_1'}{T}$ and $\Delta l = \frac{l}{L}$. We use $k(i,j)$, $r(i,j)$, and $v(i,j)$ to denote the traffic density, traffic occupancy and average traffic speed for the i^{th} traffic segment at j^{th} time grid. Here $0 \leq i \leq L$ and $0 \leq j \leq T$.

Combining the occupancy-density relation (5.1) and the LWR model (5.16), we can get a variant LWR model:

$$\frac{\partial r}{\partial t} + v\frac{\partial r}{\partial x} + r\frac{\partial v}{\partial x} = 0 \tag{5.21}$$

Here $v = v_{max}(1 - \frac{r}{r_{max}})$. v_{max} means the free flow speed and r_{max} represents the maximum occupancy. We don't need to know their values in our final approach so we assume they are known at this moment. Now we can approximate $\frac{\partial r}{\partial t}$, $\frac{\partial r}{\partial x}$, and $\frac{\partial v}{\partial x}$ by

$$\begin{aligned} \frac{\partial r(i,j)}{\partial t} &\approx \frac{r(i,j+1) - r(i,j)}{\Delta t} \\ \frac{\partial r(i,j)}{\partial l} &\approx \frac{r(i,j) - o(i-1,j)}{\Delta l} \end{aligned} \tag{5.22}$$

$$\frac{\partial v(i,j)}{\partial l} \approx \frac{v(i,j) - v(i-1,j)}{\Delta l}$$

Plugging equations (5.22) back to (5.21), we obtain (5.23) which shows how we can approximate the occupancy for each grid in the spatial discretization.

$$r(i,j+1) = r(i,j) - \frac{dt}{dl}\Big(v(i,j)\big(r(i,j) - r(i-1,j)\big) + r(i,j)\big(v(i,j) - v(i-1,j)\big)\Big) \quad (5.23)$$

The finite difference method is applicable when there is light traffic with no congestion or light congestion. When the heavy traffic congestion presents, we can use other approaches such as Godunov scheme to approximate the LWR model.

With discretization of the traffic segment, the numerical approximation methods allow us to estimate the traffic occupancy at any location $j\Delta l$ at any time $t_1 + i\Delta t$. Accordingly, the link-level occupancy at time t_1 and t_2 can be further estimated by

$$r(t = t_1) \approx \frac{\sum_j r(0,j)}{L} \quad (5.24)$$

$$r(t = t_2) \approx \frac{\sum_j r(T,j)}{L} \quad (5.25)$$

These equations disclose that, if we have the boundary conditions (two endpoints traffic occupancy) of the traffic segment, we should be able to approximate the link-level traffic occupancy, namely, there should exist a mapping function that can map the two point-measurements to the link measurement:

$$r(t) = \mathbb{F}(\boldsymbol{r}^{in}, \boldsymbol{r}^{out}) \text{ and } \dot{r} = \mathbb{F}'(\boldsymbol{r}^{in}, \boldsymbol{r}^{out}) \quad (5.26)$$

Note that, in this equation, $\widetilde{\boldsymbol{r^{in}}}$ and $\widetilde{\boldsymbol{r^{out}}}$ are the vector form of $\widetilde{r^{in}}$ and $\widetilde{r^{out}}$. Obviously, we cannot estimate the link level occupancy $r(t)$ at time t solely using the two-point occupancy measurements $\widetilde{r^{in}(t)}$ and $\widetilde{r^{out}}(t)$. According to the LWR model, to estimate $r(t)$, we will need multiple measurements of $\widetilde{r^{in}}$ and $\widetilde{r^{out}}$ along the temporal dimension (e.g., $\widetilde{\boldsymbol{r^{in}}} = \left[\widetilde{r^{in}}(t_{r1}), \dots, \widetilde{r^{in}}(t_{r2})\right]^T$ and $\widetilde{\boldsymbol{r^{out}}} = \left[\widetilde{r^{out}}(t_{r1}), \dots, \widetilde{r^{out}}(t_{r2})\right]^T$, where $t_{r1} < t < t_{r2}$). In this study, we will not directly use the finite element method, but it provides us not only the idea that we can use the point occupancy observation to approximate the link-level observation, but also the choice of the range of t_{r1} and t_{r2}. Using Figure 5.7 below as an example, if we want to estimate the link-level occupancy at time t with all the black vehicles presenting on this link, assume that there is no shockwave, we want to choose t_{r1} and t_{r2} that at least cover the time window from the entry of the first black vehicle and the leave of the last one, namely, $t_{r1} < t - l/v$ and $t_{r2} < t + l/v$. Since we may know the average speed v, we can use the free flow speed v_{max} instead.

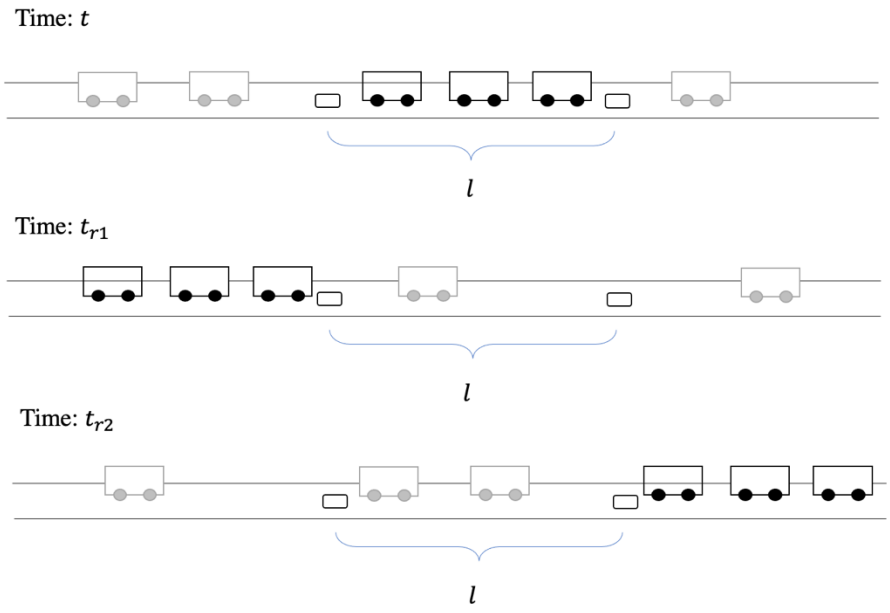


Figure 5.7 An example for determining t_{r1} and t_{r2}

Combining (5.26) with the measurement error model, we get,

$$\tilde{\dot{r}} = \mathbb{F}'(\beta_r^{in}\widetilde{\boldsymbol{r^{in}}} + \beta_r^{in}\boldsymbol{\varepsilon_r^{in}}, \beta_r^{out}\widetilde{\boldsymbol{r^{out}}} + \beta_r^{out}\boldsymbol{\varepsilon_r^{out}}) \tag{5.27}$$

Here, we use $\dot{r} = \mathbb{F}'(\widetilde{\boldsymbol{r^{in}}}, \widetilde{\boldsymbol{r^{out}}})$ instead of directly mapping from $\widetilde{\boldsymbol{r^{in}}}$ and $\widetilde{\boldsymbol{r^{out}}}$ to $\dot{r}$ (e.g. use $\dot{r} = \mathbb{F}(\widetilde{\boldsymbol{r^{in}}}, \widetilde{\boldsymbol{r^{out}}})$) is due to the consideration that the scales of ε_r^{in} or ε_r^{out} should be much smaller than $\widetilde{r^{in}}$ or $\widetilde{r^{out}}$, but not necessarily $\widetilde{r^{in}}$ and $\widetilde{r^{out}}$.

Combining (5.5) with (5.27), and letting $\beta_r = \frac{1}{\mu_r}$, we obtain,

$$\begin{aligned} &\frac{l}{\bar{l}_v}\mathbb{F}(\beta_r^{in}\widetilde{\boldsymbol{r^{in}}} + \beta_r^{in}\boldsymbol{\varepsilon_r^{in}}, \beta_r^{out}\widetilde{\boldsymbol{r^{out}}} + \beta_r^{out}\boldsymbol{\varepsilon_r^{out}}) \\ &\qquad = \beta_f^{in}\left(\widetilde{Q^{in}}(t_2) - \widetilde{Q^{in}}(t_1)\right) - \beta_f^{out}\left(\widetilde{Q^{out}}(t_2) - \widetilde{Q^{out}}(t_1)\right) + \zeta \end{aligned} \tag{5.28}$$

This equation describes the relation between the measurements of incoming and outgoing traffic flux at the two endpoints of the link, the measurements of corresponding point occupancy and their measurement errors and biases. Clearly, we don't have an explicit expression for the mapping function $\mathbb{F}$. We may use similar finite element approximating methods as discussed in previous section to approximate $\mathbb{F}$ or $\mathbb{F}'$, however, since both βs and εs are yet to be determined variables and the approximating expression of $\mathbb{F}'$ using βs and εs will surely contain their very high order forms, this makes it difficult or even impossible to solve for their estimates.

Fortunately, the deep learning methods offer us a chance to estimate the measurement biases βs, without knowing the expression of $\mathbb{F}'$. Because we assume these random measurement errors have relatively small scales ($\varepsilon_r^{in} \ll \widetilde{r^{in}}$ and $\varepsilon_r^{out} \ll \widetilde{r^{out}}$) and $E\left[\varepsilon_r^{in}\right] = E[\varepsilon_r^{out}] = 0$, we simplify (5.28) to,

$$\frac{l}{\overline{l_v}}\mathbb{F}'(\beta_r^{in}\widetilde{\boldsymbol{r^{in}}}, \beta_r^{out}\widetilde{\boldsymbol{r^{out}}}) = \beta_f^{in}\left(\widetilde{Q^{in}}(t_2) - \widetilde{Q^{in}}(t_1)\right) - \beta_f^{out}\left(\widetilde{Q^{out}}(t_2) - \widetilde{Q^{out}}(t_1)\right) \quad (5.29)$$

Letting $\mathbb{G}(\beta_f^{in}, \beta_f^{out}, \widetilde{Q^{in}}(t_2) - \widetilde{Q^{in}}(t_1), \widetilde{Q^{out}}(t_2) - \widetilde{Q^{out}}(t_1)) = \beta_f^{in}\left(\widetilde{Q^{in}}(t_2) - \widetilde{Q^{in}}(t_1)\right) - \beta_f^{out}\left(\widetilde{Q^{out}}(t_2) - \widetilde{Q^{out}}(t_1)\right)$, we obtain,

$$\frac{l}{\overline{l_v}}\mathbb{F}'\left(\beta_r^{in}\widetilde{\boldsymbol{r^{in}}}, \beta_r^{out}\widetilde{\boldsymbol{r^{out}}}\right) = \mathbb{G}(\beta_f^{in}, \beta_f^{out}, \widetilde{Q^{in}}(t_2) - \widetilde{Q^{in}}(t_1), \widetilde{Q^{out}}(t_2) - \widetilde{Q^{out}}(t_1)) \quad (5.30)$$

Since $\widetilde{r^{in}}$, $\widetilde{r^{out}}$, $\widetilde{Q^{in}}$ and $\widetilde{Q^{out}}$ are observable, $\beta_f^{in}, \beta_f^{out}, \beta_r^{in}$ and β_r^{out} are yet to be determined, and we do not need to know the explicit expression of $\mathbb{F}$ and $\mathbb{G}$, by introducing another mapping function $\mathbb{H}$, we can further convert (5.30) to (5.31) below.

$$\left[\widetilde{\beta_f^{in}}, \widetilde{\beta_f^{out}}, \widetilde{\beta_r^{in}}, \widetilde{\beta_r^{out}}\right]^T = \mathbb{H}\left(\widetilde{\boldsymbol{r^{in}}}, \widetilde{\boldsymbol{r^{out}}}, \widetilde{Q^{in}}(t_2) - \widetilde{Q^{in}}(t_1), \widetilde{Q^{out}}(t_2) - \widetilde{Q^{out}}(t_1)\right) \quad (5.31)$$

Here, the mapping function $\mathbb{H}$ represents a series of transformations, which may be hard if not impossible to express explicitly. The transformations themselves also depend on several important attributes of the physical conditions, such as the shape of the road segment's fundamental diagram and the average vehicle length, which are all unknown. Conventional methods may be difficult dealing with this situation, but the machine learning techniques offer us a chance to solve this problem. Therefore, the remaining task is to construct a neural network model to estimate $\mathbb{H}$ so that if we have the observation $\widetilde{\boldsymbol{r^{in}}}$, $\widetilde{\boldsymbol{r^{out}}}$, $\widetilde{Q^{in}}$ and $\widetilde{Q^{out}}$ we can plug them into (5.31) and obtain $\widetilde{\beta_r^{in}}$ and $\widetilde{\beta_r^{out}}$. But before we move on, we will need to make an assumption: The attributes of mapping function $\mathbb{H}$ is similar for the time we want to estimate the occupancy measurement biases and the time the historical unbiased observations were collected. For instance, if we want to

estimate the measurement biases for some weekday sensor readings, we need to have some unbiased measurements of past weekdays.

5.3.4 LSTM-based PIDL Model

Now we are going to construct a physics-informed neural network model to learn the mapping behavior of function $\mathbb{H}$. The long short-term memory (LSTM) model is used due to its ability in capturing temporal correlations in the input data. Below graph shows the scheme of this neural network model.

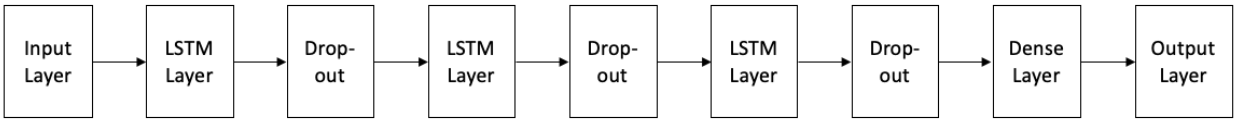


Figure 5.8 LSTM model setup

Input layer: In the training stage, the input layer takes $\left[\beta_{r*}^{in}\widetilde{\boldsymbol{r}^{in}}, \beta_{r*}^{out}\widetilde{\boldsymbol{r}^{out}}, \beta_{f*}^{in}\left(\widetilde{\boldsymbol{Q}^{in}}(t_2) - \widetilde{\boldsymbol{Q}^{in}}(t_1)\right), \beta_{f*}^{out}\left(\widetilde{\boldsymbol{Q}^{in}}(t_2) - \widetilde{\boldsymbol{Q}^{in}}(t_1)\right)\right]$ as input. Here $\widetilde{\boldsymbol{r}^{in}}$, $\widetilde{\boldsymbol{r}^{out}}$, $\widetilde{\boldsymbol{Q}^{in}}$ and $\widetilde{\boldsymbol{Q}^{out}}$ are the historical unbiased sensor readings. β_{r*}^{in}, β_{r*}^{out}, β_{f*}^{in} and β_{f*}^{out} are the randomly generated measurement biases (e.g., $\beta_{r*}^{in} = 1.1$ and $\beta_{f*}^{out} = 0.9$). We randomly pick multiple batches of $\widetilde{\boldsymbol{r}^{in}}$, $\widetilde{\boldsymbol{r}^{out}}$, $\widetilde{\boldsymbol{Q}^{in}}$ and $\widetilde{\boldsymbol{Q}^{out}}$. For each batch, we generate multiple β_{r*}^{in}, β_{r*}^{out}, β_{f*}^{in} and β_{f*}^{out}. As a results, we will obtain multiple batches of $\left[\beta_{r*}^{in}\widetilde{\boldsymbol{r}^{in}}, \beta_{r*}^{out}\widetilde{\boldsymbol{r}^{out}}, \beta_{f*}^{in}\left(\widetilde{\boldsymbol{Q}^{in}}(t_2) - \widetilde{\boldsymbol{Q}^{in}}(t_1)\right), \beta_{f*}^{out}\left(\widetilde{\boldsymbol{Q}^{in}}(t_2) - \widetilde{\boldsymbol{Q}^{in}}(t_1)\right)\right]$. The length for any of $\widetilde{\boldsymbol{r}^{in}}$, $\widetilde{\boldsymbol{r}^{out}}$, $\widetilde{\boldsymbol{Q}^{in}}$ and $\widetilde{\boldsymbol{Q}^{out}}$ (the choice of measurement time period $[t_1, t_2]$) should ensure that the input data contains sufficient information to infer $\mathbb{H}$ at time t. If we don't know the initial and ultimate condition of the entire link, we want to have at least $t_1 \leq t - l/\bar{v}$ and $t_2 \geq$

$t + l/\bar{v}$. Since we have plenty of observations to choose from, we can increase the number of observations for each time t.

LSTM layers: Each layer contains multiple nodes. Each layer is followed by a dropout to prevent overfitting.

Dense layer and output layer: The dense layer takes the output from LSTM layer and output a two-element vector $\left[\widehat{\beta_{f*}^{in}}, \widehat{\beta_{f*}^{out}}, \widehat{\beta_{r*}^{in}}, \widehat{\beta_{r*}^{out}}\right]^T$.

Loss function: For the input batch $\left[\beta_{r*}^{in}\widetilde{\boldsymbol{r}^{in}}, \beta_{r*}^{out}\widetilde{\boldsymbol{r}^{out}}, \beta_{f*}^{in}\left(\widetilde{\boldsymbol{Q}^{in}}(t_2) - \widetilde{\boldsymbol{Q}^{in}}(t_1)\right), \beta_{f*}^{out}\left(\widetilde{\boldsymbol{Q}^{in}}(t_2) - \widetilde{\boldsymbol{Q}^{in}}(t_1)\right)\right]$, its corresponding loss function is:

$$LOSS = \left(\widehat{\beta_{f*}^{in}} - \beta_{f*}^{in}\right)^2 + \left(\widehat{\beta_{f*}^{out}} - \beta_{f*}^{out}\right)^2 + \left(\widehat{\beta_{r*}^{in}} - \beta_{r*}^{in}\right)^2 + \left(\widehat{\beta_{r*}^{out}} - \beta_{r*}^{out}\right)^2 \tag{5.32}$$

For training purpose, for the two sensors we are studying, we need to have some (historical) unbiased observations of $\widetilde{\boldsymbol{r^{in}}}, \widetilde{\boldsymbol{r^{out}}}, \widetilde{Q^{in}}$ and $\widetilde{Q^{out}}$ for both of them, namely, we need to have some observations where $\beta_f^{in} = \beta_f^{out} = \beta_r^{in} = \beta_r^{out} = 1$. These data may come from the sensor readings obtained right after the sensors are freshly installed or calibrated. We then follow the Algorithm 5.1 below to prepare for the training data. During the training process, we hope that the neural network model could learn the behavior of the unknow function $\mathbb{H}$, so that once trained, it could take the occupancy measurements with unknown biases as input and gives the estimated biases as output, given $\mathbb{H}$ unchanged.

Simulation Data Generation

We are going to use the simulation data generated from SUMO to test and verify our models. A road segment with length l (e.g. $l = 400m$) is created in SUMO (Figure 5.9). Two inductive loop sensors are placed at the two endpoints of this road segment.

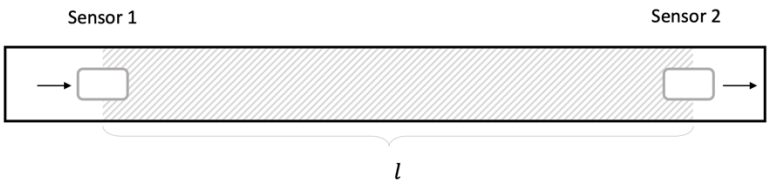


Figure 5.9 Road Segment Sample

For each test, we carry out multiple independent simulations, each simulating a 24-hour time period. The length and the speed limit of this link is set to be 400m and 15m per second respectively. The average vehicle length is set to be 5m. The total traffic volume for each of the 24-hour time period follows a uniform distribution between [10000, 30000], with the hourly traffic flow following the below pattern, added with some randomness as well.

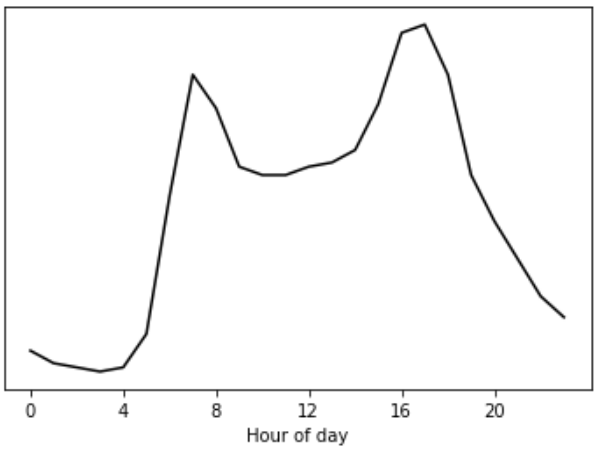


Figure 5.10 Hourly traffic flow pattern

Traffic flow and occupancy are collected from the two inductive loop sensors at a 5-second aggregation level. Therefore, after running the simulation 40 for times, we obtain 40*24*720=4024720 observations for each sensor reading.

We first divide the 40-day raw sensor readings into two groups: one with 35-day readings for model training and cross-validating; one with 5-day readings for model test.

For the 35-day readings, we prepared the training data following the Step 1 to 4 of Algorithm 1, then used the data to train the neural network model with 5-fold cross-validation. In step 3 of algorithm 1, if we choose $n = 40$, given one day observations, there are $17280 - 40 = 17240$ overlapped training samples or $17280 / 40 = 431$ independent training samples. We follow the bootstrap resample method to randomly choose 500 samples from each day to construct the training data. Note that, since in this study we are using simulated sensor readings, the raw data do not contain the random error term (ε_f or ε_r). We need to manually "add" this white noise to the readings on top of the sensor biases. We are generating the unbiased historical sensor readings following $\tilde{r} \sim N(r, \sigma r)$ and $\tilde{f} \sim N(f, \sigma f)$ for occupancy and flux, respectively. σr and σf are the standard deviations for the corresponding normal distribution and the initial value of σ it set to be 0.02. Similarly, the biased sensor readings are generated following $\tilde{r} \sim \beta_{r*} N(r, \sigma r)$ and $\tilde{f} \sim \beta_{f*} N(f, \sigma f)$.

Performance Metrics

Because our ultimate goal is to calibrate the sensor using the estimated biases and correct the readings, we are going to use the mean absolute error (MAE) below as the performance metric of the estimations.

For traffic flux correction:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\widehat{f_i} - f_i}{f_i}\right| \times 100\% \tag{5.33}$$

where f_i is the i^{th} true traffic flux; $\widehat{f_i}$ is the i^{th} traffic flux reading before or after correction.

For traffic occupancy estimation:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\widehat{r_i} - r_i}{r_i}\right| \times 100\% \tag{5.34}$$

where r_i is the i^{th} true traffic occupancy; $\widehat{r_i}$ is the i^{th} traffic flux occupancy before or after correction.

Algorithm 5.1:

- **Training data preparation:**

Step 1: Repeat step 2 to 4 until obtaining sufficient amount of training data:

Step 2: Generate manipulated sensor biases β_{r*}^{in}, β_{r*}^{out}, β_{f*}^{in} and $\beta_{f*}^{out} \in [0.5,1.5]$. Repeat step 3 to 4 until obtaining sufficient amount of data for each $[\beta_{r*}^{in}, \beta_{r*}^{out}, \beta_{f*}^{in}, \beta_{f*}^{out}]$.

Step 3: Randomly choose n consecutive observations $\left[\widetilde{\boldsymbol{r}^{in}}, \widetilde{\boldsymbol{r}^{out}}, \widetilde{\boldsymbol{Q}^{in}}, \widetilde{\boldsymbol{Q}^{out}}\right]$ from the unbiased observation pool. The choice of n should follow the instructions discussed above.

Step 4: Append the manipulated observation $\left[\beta_{r*}^{in}\widetilde{\boldsymbol{r}^{in}}, \beta_{r*}^{out}\widetilde{\boldsymbol{r}^{out}}, \beta_{f*}^{in}\left(\widetilde{\boldsymbol{Q}^{in}}(t_2) - \widetilde{\boldsymbol{Q}^{in}}(t_1)\right), \beta_{f*}^{out}\left(\widetilde{\boldsymbol{Q}^{in}}(t_2) - \widetilde{\boldsymbol{Q}^{in}}(t_1)\right)\right]$ to the input dataset. Add the corresponding $[\beta_{r*}^{in}, \beta_{r*}^{out}, \beta_{f*}^{in}, \beta_{f*}^{out}]$ to the output true output dataset.

- **Model training:**

Step 5: Train and cross-validate the LSTM model using the input dataset. Use the true output dataset to compute the loss function.

Step 6: Tune the LSTM model parameters: number of nodes for each layer, number of layers, dropouts, etc. for best cross-validating results.

- **Error estimation for unknown bias dataset:**

Step 7: Repeat step 8 until obtaining sufficient amount of input data

Step 8: Randomly choose n consecutive observations $\left[\widetilde{\boldsymbol{r}^{in}}, \widetilde{\boldsymbol{r}^{out}}, \widetilde{\boldsymbol{Q}^{in}}(t_2) - \widetilde{\boldsymbol{Q}^{in}}(t_1), \widetilde{\boldsymbol{Q}^{in}}(t_2) - \widetilde{\boldsymbol{Q}^{in}}(t_1)\right]$ from the unknown bias observation pool.

Step 9: Feed the trained LSTM model with the new inputs and obtain the model outputs $\left[\widehat{\beta_f^{in}}, \widehat{\beta_f^{out}}, \widehat{\beta_r^{in}}, \widehat{\beta_r^{out}}\right]^T$

Step 10: Use the mean of all $\widehat{\beta_f^{in}}, \widehat{\beta_f^{out}}, \widehat{\beta_r^{in}}$ and $\widehat{\beta_r^{out}}$ as the estimated occupancy measurement biases for the sensors.

Step 11: Correct traffic occupancy reading by $\widetilde{f^{in}}_{crrt} = \widetilde{f^{in}}/\widehat{\beta_f^{in}}$, $\widetilde{f^{out}}_{crrt} = \widetilde{f^{out}}/\widehat{\beta_f^{out}}$, $\widetilde{r^{in}}_{crrt} = \widetilde{r^{in}}/\widehat{\beta_r^{in}}$, and $\widetilde{r^{out}}_{crrt} = \widetilde{r^{out}}/\widehat{\beta_r^{out}}$

5.3 Results and Discussion

5.3.1 Model Performance

We constructed several neural network models following Figure 5.8. After tuning the model parameters, we finalized the last neural network with three LSTM layers and one dense layer and set the number of nodes in each LSTM layer to be 32 and dropout to be 0.2.

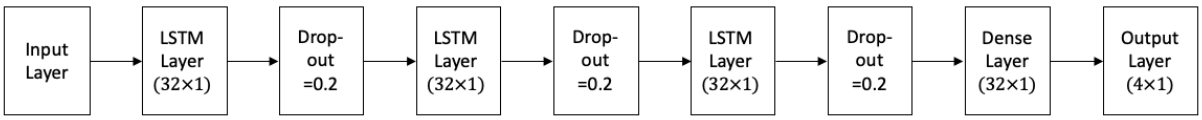


Figure 5.11 Finalized LSTM model

Using the last model, we repeated the training process 15 times. The training process is relatively stable, with only one training out of the 15 being significantly worse than the rest. We plot the loss function value for the first six trainings in Figure 5.12. We can see from this graph that, overall, the losses decrease rapidly during the first several epochs and then slowly reach equilibrium.

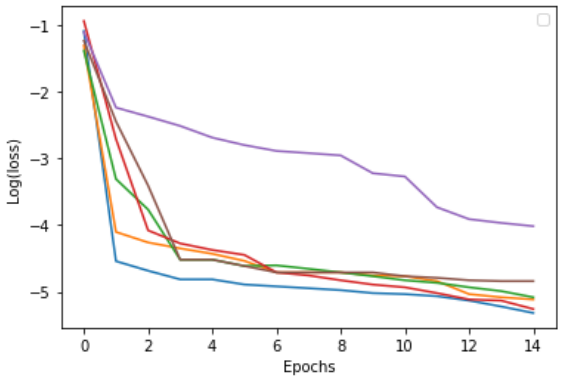
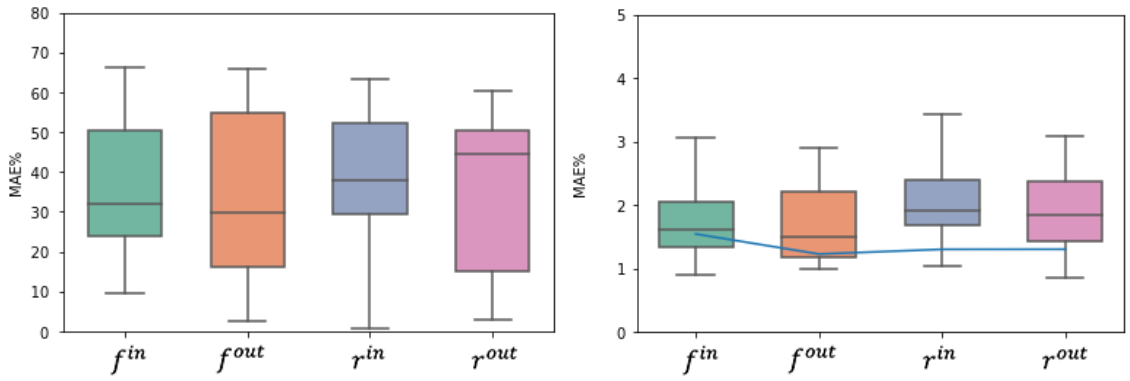


Figure 5.12 Loss value during training

Since we are using simulated observation data to verify our model, the testing data were generated using the rest 5-day sensor reading, following the same steps 2 to 4. In this case, we have the ground truth, and the true measurement biases for each observation, which we can use to test the model estimation performance.

Firstly, we tested how the model performs when all the four sensor readings have unknown biased, namely, unknown $\beta_f^{in}, \beta_f^{out}, \beta_r^{in}$ and β_r^{out}. We ran the training-testing processes 20 times and plot the results in Figure 5.12. Figure 5.12a shows the $MAE\%$ for sensor readings before correction. Since all the sensor biases were generated from 0.5 to 1.5, the average cap M cap A. cap E % for all the uncorrected measurements is around 40%. Figure 5.13a shows the $MAE\%$ of the readings calibrated by our model. The blue line is the $MAE\%$ calibrated using the actual biases and represents the theoretical optimum. We can see from the graphs that our model significantly improves the MAE for all four measurements. After correction, the $MAEs$ drop to around 2%, slightly higher than the theoretical optimum.



(a) before correction (b) after correction

Figure 5.13 MAE for sensor readings before and after correction

In reality, it may be more common that not all but some of the four sensor readings (traffic flux and occupancy readings for both upstream and downstream sensors) are biased. Therefore, we will now discuss the model performance for different scenarios separately. Figure 5.14 shows the model estimation results when both sensors are in healthy condition ($\beta_f^{in} = \beta_f^{out} = \beta_r^{in} = \beta_r^{out} = 1$, but unknown to the model). For f^{in} and r^{out}, the model almost reaches the theoretical optimum. For f^{out} and r^{in}, the model gives MAE around 2%, which is still acceptable and not much higher than the blue line.

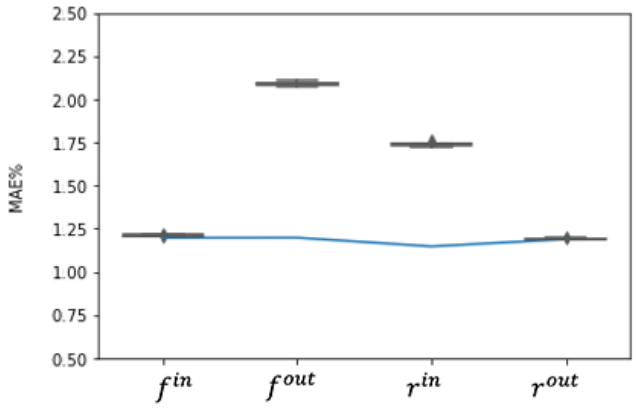
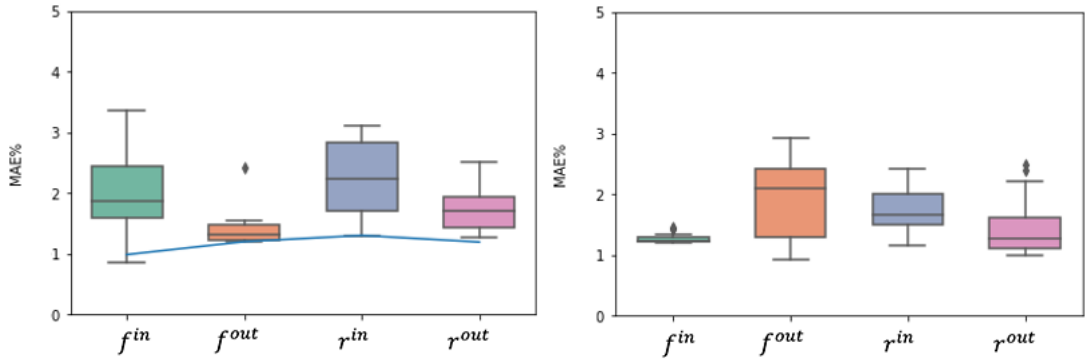


Figure 5.14 MAE with all sensor readings unbiased

Then we test the scenarios when we have one of the two sensors in good working condition. Below are two graphs comparing when we have healthy downstream sensor $\beta_f^{out} = \beta_r^{out} = 1$ (Figure 5.15a) and healthy upstream sensor $\beta_f^{in} = \beta_r^{in} = 1$ (Figure 5.15b).



(a) $\beta_f^{out} = \beta_r^{out} = 1$ (b) $\beta_f^{in} = \beta_r^{in} = 1$

Figure 5.15 MAE with one healthy sensor

5.3.2 Sensitivity Analysis

Scale of measurement biases

In this part, we are going to test how the scale of measurement biases (β) affects the model performance. Since from the previous section, we've seen no significant performance difference in estimating upstream and downstream sensors, so we pick upstream sensor measurements as the study objects. In this experiment, the studied measurement biases were generated following uniform distributions of ranges [0.5, 0.7], [0.7, 0.9], [0.9, 1.1], [1.1, 1.3] and [1.3,1.5]. MAE were computed for each group.

Figure 5.16 shows the results of the MAE for f^{in} when $\beta_f^{out} = \beta_r^{in} = \beta_r^{out} = 1$, and only β_f^{in} is randomly generated. Similarly, Figure 5.17 shows the results of the MAE for r^{in} when $\beta_f^{in} = \beta_f^{out} = \beta_r^{out} = 1$, and only β_r^{in} is randomly generated.

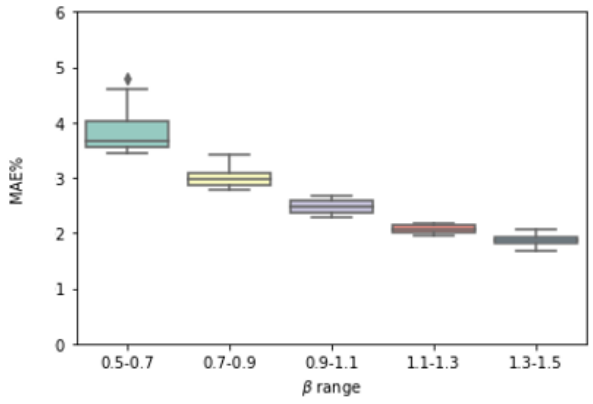


Figure 5.16 MAE of f^{in} for different β_f^{in}

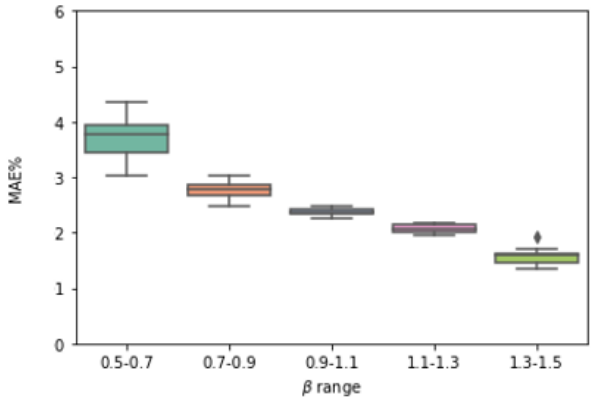


Figure 5.17 MAE of r^{in} for different β_r^{in}

In reality, it may be more common if a sensor malfunctions, both its traffic flux and occupancy readings are biased. Figure 5.18 shows the MAE for f^{in} and r^{in} when we assume both traffic flux and occupancy readings of upstream sensors are biased (unknown β_f^{in} and β_r^{in}). In contrast, the downstream sensor is in healthy condition ($\beta_f^{out} = \beta_r^{out} = 1$). We can see the pattern that MAE decreases with the increase of the measurement biases. Due to the fixed scale of white noises, smaller systematic measurement biases result in a larger fraction of uncorrectable white noises, which consequently, leads to higher *MAE* after flow volume correction.

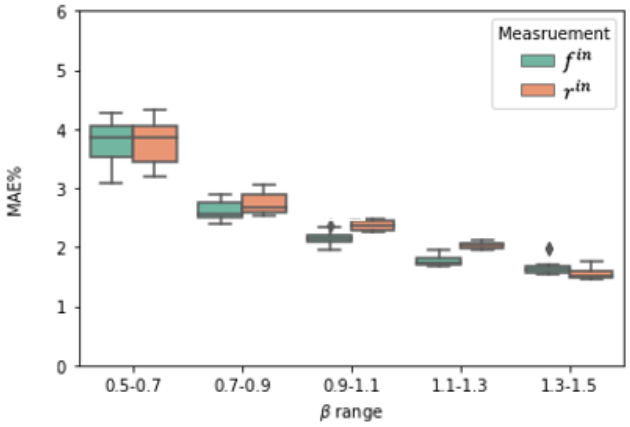
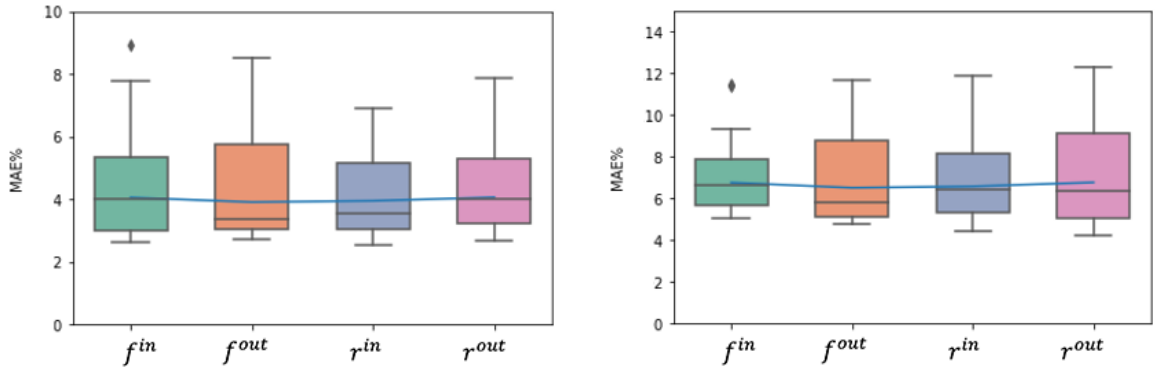


Figure 5.18 MAE of f^{in} and r^{in} for different β_f^{in} and β_r^{in}

Scale of white noise (random error) variance

As discussed previously, the simulating traffic flux and sensor readings were generated following $\tilde{r} \sim N(\beta_r r, \sigma r)$ and $\tilde{f} \sim N(\beta_f f, \sigma f)$, respectively. Initially, we set $\sigma = 0.02$ for both training data and testing data. Now we will test for two more cases when $\sigma = 0.04$ and 0.06 (Figure 5.19) and keep σ still same for both training and testing data. Here, the blue line in each graph still represents the theoretical optimum, calibrated with true sensor biases. Comparing with the original case $\sigma = 0.02$, we can see that the model corrected $MAEs$ increase with the increase of the random error variance, as well as the theoretical optimums, so the overall corrected $MAEs$ still lie around the blue line. On the other hand, we do see the upper tails of the boxplots grow a lot with larger σ, which means that with larger random error variance, our model may occasionally give relatively 'bad' estimations; however, since the majority of the estimations are in a reasonable range(<0.06 for $\sigma = 0.04$, (<0.08 for $\sigma = 0.06$), we can avoid the 'bad' estimations by repeating the training-estimating processes multiple times and get the overall mean values as the ultimate results.



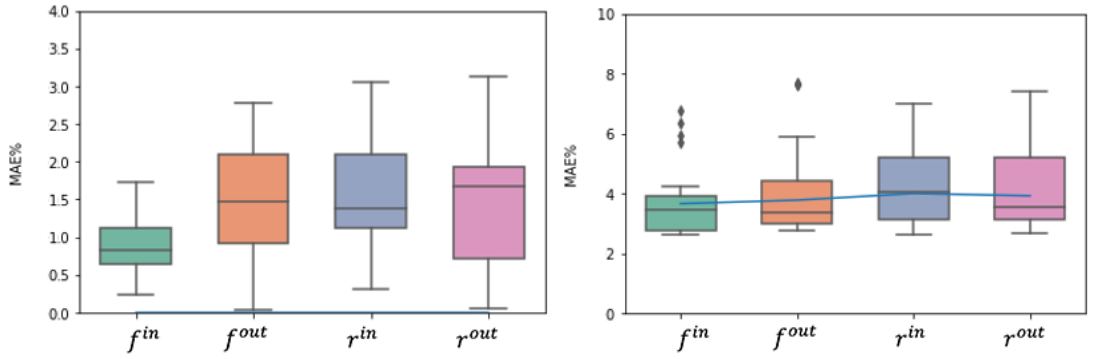
(a) $\sigma = 0.04$ (b) $\sigma = 0.06$

Figure 5.19 MAE for different scale of σ

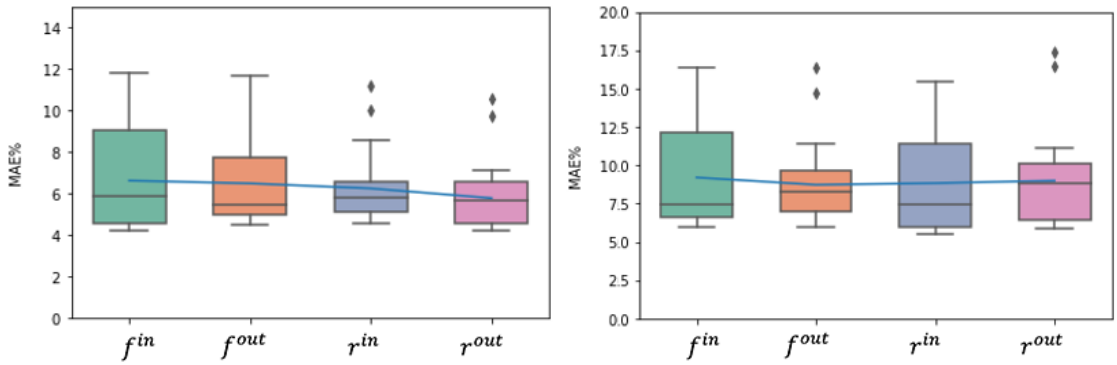
Change in variance of white noise (random error)

The former experiments generate training and testing data with the same σ. However, when we want to apply the model in the real world, since the training data (the unbiased historical observations) and testing data (the observations with biases unknown and yet to be determined) may come from two timestamps far away from each, the variance of random measurement error may also change. Now in this part, we are going to test how this change in the random error variance impacts the model estimation results. The training data were still prepared with $\sigma_{train} = 0.02$. We generated extra testing cases for $\sigma_{test} = 0.00, 0.04, 0.06,$ and 0.08. The results are shown in Figure 5.20. Similarly, the blue line in each graph represents the theoretical optimum.

The results resemble the last part. The corrected $MAEs$ do increase with larger σ, and they still do not deviate much from the theoretical optimum in general. To avoid extremely bad estimations, we can use the same method mentioned in the last part.



(a) $\sigma_{test} = 0.00$ (b) $\sigma_{test} = 0.04$



(c) $\sigma_{test} = 0.06$ (d) $\sigma_{test} = 0.08$

Figure 5.20 MAE for $\sigma_{train} = 0.02$ and different σ_{test}

Traffic segment length

In this part we are going to compare the model performance for different traffic segment lengths, as in reality, the length of our studying road may vary. We choose the segment lengths from 200m to 1000m and repeat for each length the simulating-training-testing process. The corresponding MAEs are shown below Figure 5.21, from which we cannot find a significant relationship between the segment length and the corrected MAE. In most of the tests, the model is able to keep the

estimation errors below 0.03 and the model shows its robustness against different road segment lengths.

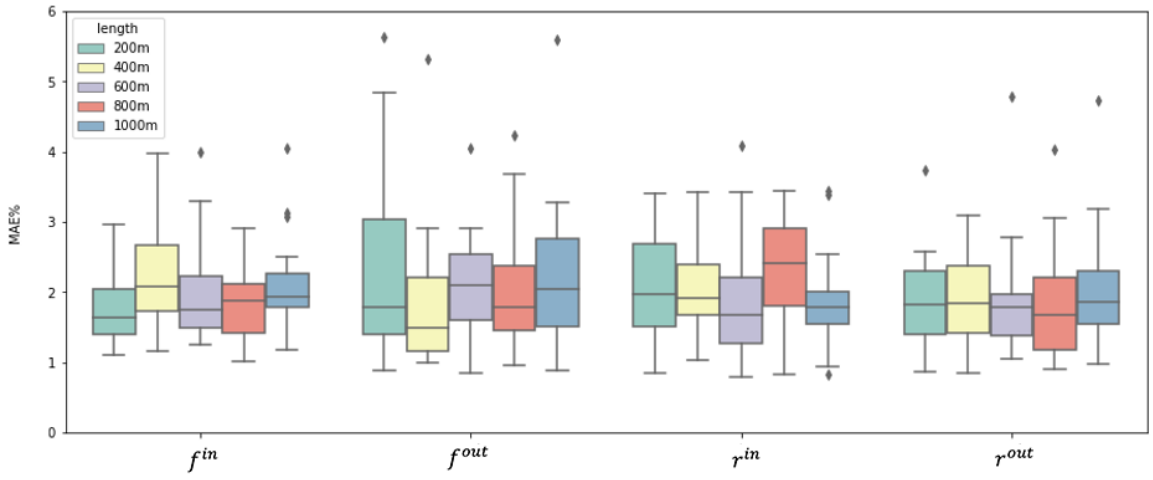


Figure 5.21 MAE for different traffic segment length

Sample size

In general, machine learning techniques are data-driven, requiring large sample sizes to train the models, so in this section, we test how our model handles different training data sizes. Since we initially obtained 40-day sensor reading data from SUMO simulations and randomly picked 35-day data as the training set, we want to see how the model reacts if we reduce the training data size. We did three more experiments, each with training data of 25-day, 15-day, and 5-day, and plot the results in Figure 5.22. The figure illustrates that, the model trained with 35-day and 25-day data have quite similar performance. As the training sample size further decreases to 15-day and 5-day, the model does perform slightly worse. However, even with only 5-days of training data, the MAE s are well controlled (under 0.04). This property may greatly improve the applicability of the mode.

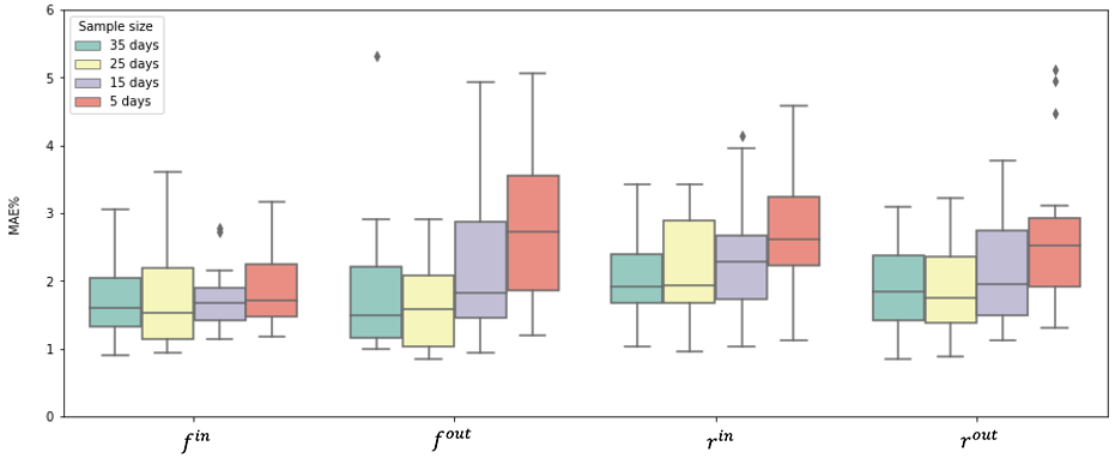


Figure 5.22 MAE for different training data sizes

5.3.3 Discussions

In this study, a sensor error correction model was developed for the traffic flux measurement and the traffic occupancy measurement. Since the traffic flux and occupancy are the two direct and non-identical measurements from the single-loop detector system, we may take advantage of the additional information one measurement contains to calibrate the potential errors in the other. Based on the traffic flow ordinary differential equation (ODE) model, which describes the dynamics of the traffic flow, we were able to discover for a road segment the relationship between its two end-point traffic flux and the link level traffic occupancy. We further gained insights from the traffic LWR model and its numerical approximation method on the relation between the link level and its end-points traffic occupancy. Combining all these ideas, we constructed a mapping function that connects the four sensor observations (upstream and downstream traffic flux and occupancy) to their unknown biases and a Physics-Informed LSTM neural network model, which we hoped could learn this connection. With the historical data serving as ground truth, we trained the neural network model with the training set manipulated from the historical data. After proper

training, the neural network model showed the ability to capture the attributes of the mapping function and could provide a good estimation of sensor biases and accordingly calibrate the sensor observations. The sensitivity analysis also proves the reliability and robustness of the sensor error correction model.

The main contribution of this study is the development of a sensor reading error correction model using the dynamical property of the traffic flow. Most other research that studies traffic flux measurement errors merely uses the traffic flux data. But taking use of the traffic dynamics, the traffic ODE model in this study could make a connection of the traffic flux to the traffic occupancy, another raw measurement from a same sensor. Conventional studies identifying the flux measurement errors usually require the assists of data from the multiple sensors within the traffic network and may require the presence of healthy sensors. Using the insight that the traffic flux and occupancy are dynamically related; our model can narrow down the study scope into a traffic segment with only two sensors and have no assumption of a healthy sensor presenting at the time of estimating.

On the other hand, this works well combined the traffic flow domain knowledge with the advantage of the machine learning techniques. It would be very difficult if not impossible, to analytically disclose the connection between the traffic flux measurements, two end-point traffic occupancy measurements, and the measurement biases, since this connection may consist of several layers of non-linear transformations which may contain several unknown traffic or network parameters (e.g., fundamental diagram, average vehicle length). But constructed on top of the traffic ODE model, the Physics-Informed LSTM neural network in this study well matches our demand. After proper training, the model shows its ability to capture the dynamics of the traffic flow and the multiple

transformations, therefore, it can successfully discover the mapping from the sensor measurements to their biases.

Chapter 6 Conclusions

This dissertation focused on estimating and correcting the measurement biases for the inductive loop sensor system. The critical role the inductive loop sensors play in public traffic management and the need for their reliability justify the importance and urgency of studying their health condition. Instead of solely identifying the malfunctioning sensors, this study attempted to quantify the measurement bias for each sensor by taking advantage of the spatial-temporal relationship of the sensor data and the power of Physics-Informed Deep Learning (PIDL) techniques.

To exploit the spatial-temporal relationship between traffic flows in a network level, we developed three PIDL models based on the fully connected neural network, CNN and LSTM to estimate the measurement biases for traffic network sensors and recover the erroneous traffic volume readings. The domain knowledge about the sensor observation data is reflected in the machine learning models via input layer structure design besides the training data itself. We found that with the simpler model structure, the fully connected neural network model cannot fully utilize the spatial-temporal information embedded in the training data, even with the help of the PIDL design. On the contrary, the PIDL models based on CNN and LSTM estimate the sensor biases much better, with LSTM-based model slightly outperforming CNN-based one. Their estimation performance is also very consistent and relatively insensitive to factors including the sensor (link) positions in the network, the scale of sensor measurement biases, and the variance of white noise. With the help of more correct observations from the healthy (recently calibrated) sensors, the models can significantly reduce both the estimation errors and the error variance and produce promising results even under changes of traffic flow pattern beyond the range used for training.

To exploit the physical relationship between different traffic state parameters, we constructed a sensor error correction model for the traffic flux measurement and the traffic occupancy measurement. Based on the traffic flow ordinary differential equation (ODE) model, which describes the traffic flow dynamics, we were able to discover for a road segment the relationship between its two end-point traffic flux and the link level traffic occupancy. We further gained insights from the partial differential equation based LWR model and its numerical approximation method for the relation between the link level and its end-points traffic occupancy. Combining all these ideas, we constructed a mapping function that connects the four sensor observations (upstream and downstream traffic flux and occupancy) to their unknown biases and trained a Physics-Informed LSTM neural network model. The prior knowledge of traffic flow models is passed onto the LSTM model via the choice of the input variables and their structure and the preparation process of training data. With this setup, the neural network model showed the ability to capture the attributes of the mapping function, give a reasonable estimation of sensor biases, and accordingly calibrate the sensor observations. The sensitivity analysis also proves the reliability and robustness of the sensor error correction model.

Statistical models use statistics to extract information from data and deduce the relationship between variables. Conventional statistical models usually have a high level of interpretability, giving additional prediction information like covariance, significance, P-values, confidence interval, etc. For instance, in the study described in Chapter 3, with the sensor error estimation model constructed using the generalized method of moments, the authors were able to statistically prove the significance of a sensor bias via a hypothesis test. Having experience in both statistical modeling and machine learning modeling, I do see situations where machine learning is more applicable. When high dimensionality and high complexity are present in the data, and the data

comes in vast volume, the conventional statistical models may be insufficient to capture its full features. On the contrary, machine learning can rely on data itself to train and test the models and benefit from the complexity of the data. When we have little domain knowledge about the data, machine learning models can solely use the data to tell stories. However, since data science is a multi-disciplinary field with knowledge from computer science, statistics, and domain-specific expertise, machine learning techniques should also take advantage of the domain knowledge. Beginning with the pure analysis of raw data, machine learning technologies had recognized the significance of domain knowledge in the accurate prediction of complex systems. In fact, the domain knowledge can be passed onto machine learning modeling in almost every step, including data collection, processing, training data preparation, and model designing. The physics-informed deep learning models constructed in this dissertation adopted this idea, by providing prior knowledge in the transportation engineering domain to the deep learning models. On the other hand, incorporating domain knowledge in machine learning testing and validating is also essential, since testing is a knowledge-building process for machine learning models. Domain expertise can guide the preparation of testing data and ensure the coverage of the testing cases, and accordingly, improve the models and confirm the models' applicability.

Looking at the readily usable infrastructure-based traffic sensor data and improving its accuracy, this study has the potential to contribute to more effective traffic management promptly. In the short term, the penetration of mobile data is still low, so this part of the data may not be statistically representative to provide aggregated data needed for system-level analyses or draw definitive conclusions about the current transportation system. On the other hand, the increase in the mobile data penetration will not diminish the value of this study, instead, mobile data could complement infrastructure-based on sensor to provide more information and physical relation that could benefit

the machine learning model. For example, the mobile data can provide traffic model parameters such as average vehicle length, average speed, and free flow speed at locations where there is no infrastructure-based sensor or the sensors are malfunctioning. Mobile sensors also generate vehicle trajectory data which could provide more spatial-temporal relations than conventional fixed-location sensors. Therefore, there are great opportunities to study the incorporation of mobile data into infrastructure-based traffic data, which will be left for future work.

References

Agarwal, S., Kachroo, P., & Contreras, S. (2016). A Dynamic Network Modeling-Based Approach for Traffic Observability Problem. *IEEE Transactions on Intelligent Transportation Systems*, *17*(4), 1168–1178. https://doi.org/10.1109/TITS.2015.2499538

Behrisch, M., Bieker, L., Erdmann, J., & Krajzewicz, D. (2011). SUMO – Simulation of Urban MObility: An Overview. In S. & U. of O. Aida Omerovic, R. I.-R. T. P. Diglio A. Simoni, & R. I.-R. T. P. Georgiy Bobashev (Eds.), *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind. http://www.thinkmind.org/index.php?view=instance&instance=SIMUL+2011

Bickel, P. J., Chen, C., Kwon, J., Rice, J., Zwet, E. van, & Varaiya, P. (2007). Measuring Traffic. *Statistical Science*, *22*(4), 581–597. https://doi.org/10.1214/07-STS238

Brumbaugh, S., Firestine, T., Notis, K., Randrianarivelo, S., & United States. Department of Transportation. Bureau of Transportation Statistics. (2018). *Transportation Economic Trends 2018* (Transportation Economic Trends (TET)). https://doi.org/10.21949/1502599

Cao, M., Li, V. O. K., & Chan, V. W. S. (2020). A CNN-LSTM Model for Traffic Speed Prediction. *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 1–5. https://doi.org/10.1109/VTC2020-Spring48590.2020.9129440

Chandramohan, A., Poel, M., Meijerink, B., & Heijenk, G. (2019). Machine Learning for Cooperative Driving in a Multi-Lane Highway Environment. *2019 Wireless Days (WD)*, 1–4. https://doi.org/10.1109/WD.2019.8734192

Chen, C. (2003). Freeway Performance Measurement System (PeMS). *PATH Research Report*, Article UCB-ITS-PRR-2003-22. https://trid.trb.org/view/662789

Chen, C., Kwon, J., Rice, J., Skabardonis, A., & Varaiya, P. (2003). Detecting Errors and Imputing Missing Data for Single-Loop Surveillance Systems. *Transportation Research Record*, *1855*(1), 160–167. https://doi.org/10.3141/1855-20

Choi, Y.-Y., Shon, H., Byon, Y.-J., Kim, D.-K., & Kang, S. (2019). Enhanced Application of Principal Component Analysis in Machine Learning for Imputation of Missing Traffic Data. *Applied Sciences*, *9*(10), 2149. https://doi.org/10.3390/app9102149

Cui, Z., Ke, R., Pu, Z., & Wang, Y. (2020). Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values. *Transportation Research Part C: Emerging Technologies*, *118*, 102674. https://doi.org/10.1016/j.trc.2020.102674

Dailey, D. J. (1993). *IMPROVED ERROR DETECTION FOR INDUCTIVE LOOP SENSORS* (WA-RD 300.1). Article WA-RD 300.1. https://trid.trb.org/view/489576

Farhan, J., & Fwa, T. F. (2013). Airport Pavement Missing Data Management and Imputation with Stochastic Multiple Imputation Model. *Transportation Research Record*, *2336*(1), 43–54. https://doi.org/10.3141/2336-06

Greenberg, H. (1959). An Analysis of Traffic Flow. *Operations Research*, *7*(1), 79–85. https://doi.org/10.1287/opre.7.1.79

Greenshields, B., Bibbins, J., Channing, W., & Miller, H. (1935). *A study of traffic capacity*. *1935*.

Han, Y., Wang, M., Li, L., Roncoli, C., Gao, J., & Liu, P. (2022). A physics-informed reinforcement learning-based strategy for local and coordinated ramp metering. *Transportation Research Part C: Emerging Technologies*, *137*, 103584. https://doi.org/10.1016/j.trc.2022.103584

Hawkins, D. M. (1980). *Identification of Outliers*. Springer Netherlands. https://doi.org/10.1007/978-94-015-3994-4

Herty, M., Klar, A., & Singh, A. K. (2007). An ODE traffic network model. *Journal of Computational and Applied Mathematics*, *203*(2), 419–436. https://doi.org/10.1016/j.cam.2006.04.007

Huang, A. J., & Agarwal, S. (2020). Physics Informed Deep Learning for Traffic State Estimation. *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. https://doi.org/10.1109/ITSC45102.2020.9294236

Jacobson, L. N., Nihan, N. L., & Bender, J. D. (1990). DETECTING ERRONEOUS LOOP DETECTOR DATA IN A FREEWAY TRAFFIC MANAGEMENT SYSTEM. *Transportation Research Record*, *1287*. https://trid.trb.org/view/352876

Jagadeesh, G. R., Dhinesh, G. R., & Srikanthan, T. (2014). Method for accuracy assessment of aggregated freeway traffic data. *IET Intelligent Transport Systems*, *8*(4), 407–414. https://doi.org/10.1049/iet-its.2013.0094

Jia, Z., Chen, C., Coifman, B., & Varaiya, P. (2001). *The PeMS algorithms for accurate, real-time estimates of g-factors and speeds from single-loop detectors*. 536–541. https://doi.org/10.1109/ITSC.2001.948715

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, *3*(6), 422–440. https://doi.org/10.1038/s42254-021-00314-5

Kheterpal, N., Parvate, K., Wu, C., Kreidieh, A., Vinitsky, E., & Bayen, A. (2018). *Flow: Deep Reinforcement Learning for Control in SUMO*. 134–115. https://doi.org/10.29007/dkzb

Klein, L. A., Mills, M. K., Gibson, D., & Lawrence A. Klein. (2006). *Traffic detector handbook: Third edition. Volume II* (FHWA-HRT-06-139). https://rosap.ntl.bts.gov/view/dot/936

Koh, S. S., Zhou, B., Yang, P., Yang, Z., Fang, H., & Feng, J. (2018). Reinforcement Learning for Vehicle Route Optimization in SUMO. *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 1468–1473. https://doi.org/10.1109/HPCC/SmartCity/DSS.2018.00242

Kwon, J., Chen, C., & Varaiya, P. (2004). Statistical Methods for Detecting Spatial Configuration Errors in Traffic Surveillance Sensors. *Transportation Research Record*, *1870*(1), 124–132. https://doi.org/10.3141/1870-16

Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, *9*(5), 987–1000. https://doi.org/10.1109/72.712178

Lebacque, J. P. (1996). *THE GODUNOV SCHEME AND WHAT IT MEANS FOR FIRST ORDER TRAFFIC FLOW MODELS*. https://trid.trb.org/view/481282

Li, Y., Li, Z., & Li, L. (2014). Missing traffic data: Comparison of imputation methods. *IET Intelligent Transport Systems*, *8*(1), 51–57. https://doi.org/10.1049/iet-its.2013.0052

Lighthill, M. J., & Whitham, G. B. (1955). 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*, *229*(1178), 317–345.

Liu, Z., Sharma, S., & Datla, S. (2008). Imputation of Missing Traffic Data during Holiday Periods. *Transportation Planning and Technology*, *31*(5), 525–544. https://doi.org/10.1080/03081060802364505

Lu, X.-Y., Varaiya, P., Horowitz, R., Guo, Z., & Palen, J. (2012). Estimating Traffic Speed with Single Inductive Loop Event Data. *Transportation Research Record*, *2308*(1), 157–166. https://doi.org/10.3141/2308-17

Ma, M., Liang, S., Guo, H., & Yang, J. (2017). Short-term traffic flow prediction using a self-adaptive two-dimensional forecasting method. *Advances in Mechanical Engineering*, *9*(8), 1687814017719002. https://doi.org/10.1177/1687814017719002

MacNicholas, M. J. (2008). *A Simple and Pragmatic Representation of Traffic Flow*. https://trid.trb.org/view/868786

Margiotta, R. (2002). *STATE OF THE PRACTICE FOR TRAFFIC DATA QUALITY* (EDL #13768). Article EDL #13768. https://trid.trb.org/view/731770

Mo, Z., Shi, R., & Di, X. (2021). A physics-informed deep learning paradigm for car-following models. *Transportation Research Part C: Emerging Technologies*, *130*, 103240. https://doi.org/10.1016/j.trc.2021.103240

Nguyen, H., Kieu, L.-M., Wen, T., & Cai, C. (2018). Deep learning methods in transportation domain: A review. *IET Intelligent Transport Systems*, *12*(9), 998–1004. https://doi.org/10.1049/iet-its.2018.0064

Nihan, N. L. (1997). Aid to Determining Freeway Metering Rates and Detecting Loop Errors. *Journal of Transportation Engineering*, *123*(6), 454–458. https://doi.org/10.1061/(ASCE)0733-947X(1997)123:6(454)

Nihan, N. L., & Holmesland, K. O. (1980). Use of the box and Jenkins time series technique in traffic forecasting. *Transportation*, *9*(2), 125–143. https://doi.org/10.1007/BF00167127

Payne, H. J., Helfenbein, E. D., Knobel, H. C., & Technology Service Corporation. (1976). *Development and testing of incident detection algorithms. Vol. 2, research methodology and detailed results.* (FHWA-RD-76-20). https://rosap.ntl.bts.gov/view/dot/744

Polson, N. G., & Sokolov, V. O. (2017). Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, *79*, 1–17. https://doi.org/10.1016/j.trc.2017.02.024

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). *Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations* (arXiv:1711.10561). arXiv. https://doi.org/10.48550/arXiv.1711.10561

Rajagopal, R., Nguyen, X., Ergen, S. C., & Varaiya, P. (2008). Distributed Online Simultaneous Fault Detection for Multiple Sensors. *2008 International Conference on Information Processing in Sensor Networks (Ipsn 2008)*, 133–144. https://doi.org/10.1109/IPSN.2008.41

Rajagopal, R., & Varaiya, P. (2009). *Evaluating the Health of California's Loop Sensor Network*. 18.

Ranjan, N., Bhandari, S., Zhao, H. P., Kim, H., & Khan, P. (2020). City-Wide Traffic Congestion Prediction Based on CNN, LSTM and Transpose CNN. *IEEE Access*, *8*, 81606–81620. https://doi.org/10.1109/ACCESS.2020.2991462

Shladover, S. E. (1993). Potential contributions of intelligent vehicle/ highway systems (IVHS) to reducing transportation's greenhouse gas production. *Transportation Research Part A: Policy and Practice*, *27*(3), 207–216. https://doi.org/10.1016/0965-8564(93)90060-X

Soares, G., Macedo, J., Kokkinogenis, Z., & Rossetti, R. J. F. (2013). *An Integrated Framework for Multi-Agent Traffic Simulation using SUMO and JADE*. 8.

Turner, S. (2007). *Quality Control Procedures for Archived Operations Traffic Data*. 44.

Underwood, R. T. (1961). *SPEED, VOLUME, AND DENSITY RELATIONSHIPS*. https://trid.trb.org/view/115231

US DOT. (2006, October). *Traffic Detector Handbook: Third Edition—Volume I - FHWA-HRT-06-108*. https://www.fhwa.dot.gov/publications/research/operations/its/06108/03.cfm

Vanajakshi, L., & Rilett, L. R. (2004). Loop Detector Data Diagnostics Based on Conservation-of-Vehicles Principle. *Transportation Research Record*, *1870*(1), 162–169. https://doi.org/10.3141/1870-21

Varaiya, P. (2001). Freeway performance measurement system. *PeMS v3, Phase 1: Final Report, California PATH Research Report. UCBITS-PWP-200117*.

Waller, S. T., Kockelman, K. M., Sun, D., Boyles, S., Lin, D.-Y., Ng, M., Seraj, S., Tassabehji, M., Valsaraj, V., & Wang, X. (2008). *Archiving, Sharing, and Quantifying Reliability of Traffic Data* (FHWA/TX-09/0-5686-1). Article FHWA/TX-09/0-5686-1. https://trid.trb.org/view/891045

Wegener, A., Piórkowski, M., Raya, M., Hellbrück, H., Fischer, S., & Hubaux, J.-P. (2008). TraCI: An interface for coupling road traffic and network simulators. *Proceedings of the 11th Communications and Networking Simulation Symposium*, 155–163. https://doi.org/10.1145/1400713.1400740

Weijermars, W. A. M., & Van Berkum, E. C. (2006). Detection of Invalid Loop Detector Data in Urban Areas. *Transportation Research Record*, *1945*(1), 82–88. https://doi.org/10.1177/0361198106194500111

Williams, B. M., Durvasula, P. K., & Brown, D. E. (1998). Urban Freeway Traffic Flow Prediction: Application of Seasonal Autoregressive Integrated Moving Average and Exponential Smoothing Models. *Transportation Research Record*, *1644*(1), 132–141. https://doi.org/10.3141/1644-14

Yang, Y., Yang, H., & Fan, Y. (2019). Networked sensor data error estimation. *Transportation Research Part B: Methodological*, *122*, 20–39. https://doi.org/10.1016/j.trb.2019.01.013

Zhang, J., Wang, F.-Y., Wang, K., Lin, W.-H., Xu, X., & Chen, C. (2011). Data-Driven Intelligent Transportation Systems: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, *12*(4), 1624–1639. https://doi.org/10.1109/TITS.2011.2158001

Zhao, Z., Chen, W., Wu, X., Chen, P. C. Y., & Liu, J. (2017). LSTM network: A deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, *11*(2), 68–75. https://doi.org/10.1049/iet-its.2016.0208

Zheng, J., & Huang, M. (2020). Traffic Flow Forecast Through Time Series Analysis Based on Deep Learning. *IEEE Access*, *8*, 82562–82570. https://doi.org/10.1109/ACCESS.2020.2990738

Zhuang, Y., Ke, R., & Wang, Y. (2019). Innovative method for traffic data imputation based on convolutional neural network. *IET Intelligent Transport Systems*, *13*(4), 605–613. https://doi.org/10.1049/iet-its.2018.5114