EVALUATING AND FORECASTING THE OPERATIONAL PERFORMANCE OF ROAD INTERSECTIONS

ALI NEMATICHARI

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTERS OF SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING & COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

APRIL 2022

Abstract

Road intersections represent one of the most complex configurations encountered when traversing road networks. It is therefore of vital importance to improve their operational performance, as that can significantly contribute towards the efficiency of the whole transport network. Traditional approaches to improve the efficiency of intersections are based on analysis of static data or expert opinions. However, due to the advancements on Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication technologies, it is possible to enhance safety and improve road intersection efficiency by continuously monitoring traffic conditions and enabling situational awareness of vehicle drivers. Towards this end, we design, develop and evaluate a system for evaluating and forecasting the operational performance of road intersections by mining streams of V2I data. Our system makes use of graph mining and trajectory data mining methods to continuously evaluate a set of well-defined *measures of effectiveness* (MOEs) for traffic operations at different levels of road network abstraction. In addition, the system enables interactive analysis and exploration of the various MOEs. The system architecture and methods are general and can be used in various settings requiring continuous monitoring and/or forecasting of the road network state.

Acknowledgements

First, I would like to thank my supervisor Manos Papagelis for his guidance, motivation, and excellent advice. He patiently introduced me to trajectory mining and kindly shared his knowledge with me. His constant incentive and support have made this work possible.

I would also like to thank Tilemachos Pechlivanoglou who was there when I started my master's and helped a lot with starting my research.

I would like to give my special thanks to my defense committee members, Ruth Urner, and Mehdi Nourinejad, for taking their time to read my thesis and their constructive technical reviews of the current work.

I would also like to acknowledge my colleagues and friends who helped me in this project and shared all of their experience with me and made a memorable experience for me.

Finally, I thank my loving parents and siblings, whom I miss the most, for their supports during these years.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Traffic congestion describes a situation on transport networks that occurs when demand approaches the capacity of a road (or of the intersections along the road). There is an exponential rise in vehicle numbers around the world, there are numerous issues coming up which are connected to the incompatibility of transportation infrastructure due to this rise in the traffic congestion. Traffic congestion is characterized by slower speeds, longer trip times, and increased vehicle queues and is associated to significant social, economical and environmental costs [16]. Heavy traffic causes high costs, wasted time for commuters, anxiety, and urban issues, and it leads to increased polluted air and early car wear. Short-term forecasting of traffic conditions is a critical component essential for the effective operation of intelligent transportation systems (ITS) systems. It is impossible to optimize traffic management of road systems without accurate prediction. *Road intersections* represent one of the most complex configurations encountered when traversing road networks and predicting road congestion and a high percentage of accidents occur at these locations [60]. Intersections play a significant role in the road map network, and the cars queuing behind the traffic lights or slowing down to enter another road branch have a great effect on the total travel time of cars. It is therefore of vital importance to improve their operational performance, as that can significantly contribute towards the efficiency of the whole transport network. Having the intersection evaluations can enhance the traffic efficiency and result in smooth traffic circulation and it also makes the city safer and it makes a tool to ease making traffic control measures for transport managers. The main purpose of this research is to explore a new methodology for identifying and predicting intersection congestion. To minimize traffic jams, new advanced traffic information systems (ATIS) and intelligent transportation systems are being developed. They can automatically process data about traffic conditions and monitor traffic using information from cameras, smartphones, induction loops, and other detectors [37]. An improved road intersection monitoring service is of vital importance and has the premise of: (*i*) increasing the safety and efficiency of road intersections by enabling situational awareness of vehicle drivers, and (*ii*) enabling a more data-driven decision making by informing policy makers of hard to obtain, but critical road intersection analytics data.

1.1 Limitations

Existing research on the identification of traffic congestion usually takes the threshold value of a certain traffic parameter as the discriminating criterion and establishes the identification model. When the actual value exceeds the threshold value, it is considered that there is a traffic congestion [42]. Many evaluation indicators are proposed, such as speed, traffic density, vehicle queue length, and traffic flow to identify the current traffic state. After calculating the vehicle queue length, the threshold number of queue vehicles is compared with the real-time number of queue vehicles at an intersection, and the state of the intersection is derived. Although many existing studies regarding travel times estimation use only road evaluations as the features to their work, there are some of them which have brought both road and intersection as edges and nodes and use the combination of them to do the work, they have shown that it can contribute to the state of the art and improve the other existing methods and it has been ignored for a long time in a lot of studies. One example of such papers can be seen in [53], in which the authors use dual graph for traffic forecasting. Hence, it is necessary to predict the traffic status of the intersection which is prone to frequent congestion, to determine the traffic congestion trend in advance, and to realize the early warning of traffic congestion. By providing corresponding improvement measures, it is possible to eliminate potential traffic congestion problems in advance. On the other hand, traditional approaches to improve the efficiency of intersections are based on analysis of static data or expert opinions. However, today's vehicles are no longer stand-alone transportation means. Due to the advancements on *Vehicle-to-Vehicle* (V2V) and *Vehicle-to-Infrastructure* (V2I) communication technologies it is possible to enhance safety and improve road intersection efficiency by continuously monitoring traffic information and enabling situational awareness of vehicle drivers. Towards this end, we designed and developed a system that adopts a data-driven approach on evaluating the operational performance of road intersections by monitoring and forecasting metrics of daily traffic flow that could identify critical road conditions, such as delayed road directions and traffic queue length at different times. One of the limitations we encountered was lack of long-term precise traffic data. Hence, we created a platform to generate synthetic data that is close to real-world traffic scenarios.

1.2 Our Approach

Our approach is based on first defining *road segments*, the smallest granularity of a road unit for which metrics are meaningful. In particular, every road intersection consists of a number of road segments. Then, for each road segment, we have identified a number of *measures of effectiveness* (MOEs) that can be used to determine its performance, based on industry standards. Once MOEs for each road segment are computed, we perform aggregation at multiple higher levels of abstraction to determine the performance of *a road direction* (set of continuous road segments in a direction) and *the road intersection* as a whole (see Fig. 1.1). MOEs for each road intersection are reported in a dashboard (see Fig. 6.3). In order for the current system to work, we require two inputs; One of which is the road network, and the other one is the traffic data; It is a little tricky to find the second input because it has to be a noiseless and long series. To tackle this problem, we have created a system to simulate a close to real-world data.

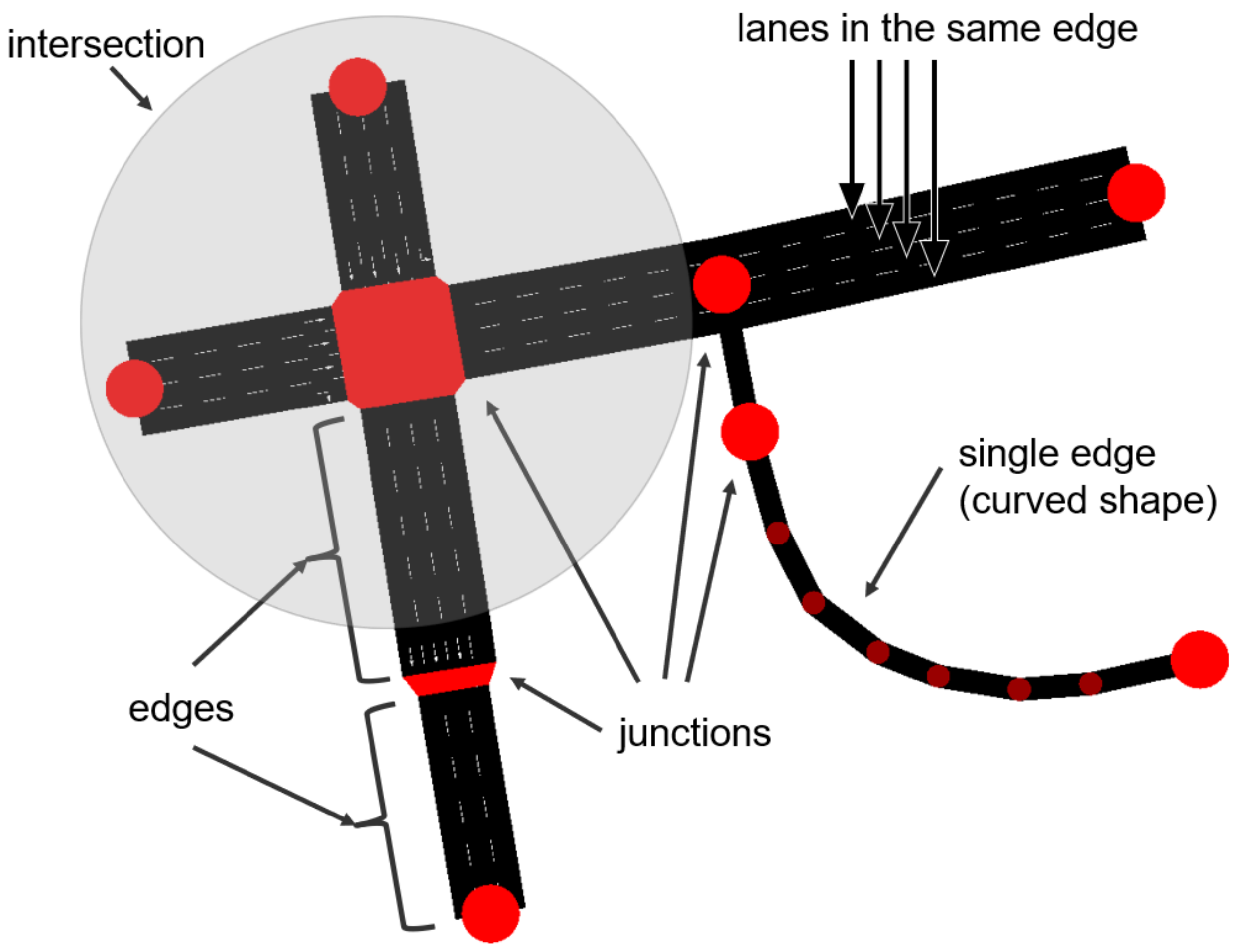


Figure 1.1: Illustration of *edges*, *junctions* and *lanes* forming a *road intersection*. Our system evaluates and forecasts the operational performance of road intersections by mining streams of trajectory data. Metrics are based on well-defined MOEs.

1.3 Contributions

. We list below the major contributions:

- We introduce the problem of *evaluating the operational performance of a road intersection* in the road network.

- We propose a novel data-driven method for addressing the problem based on graph theory and data stream mining. Complex road intersections are efficiently represented as graphs, while measures of effectiveness (MOEs) are computed in real-time using the streaming model.
- We propose a method for forecasting the operational performance of a road intersection in the future that is based on a *structural time series* deep learning model.
- We conduct a comprehensive empirical study on both synthetic and realistic road networks to evaluate the proposed method against sensible baselines.

1.4 Organization

. The remainder of the thesis is organized as follows. The related work is discussed in Section 2. Section 3 provides preliminaries and formally defines the problem of interest and engineering challenges. Section 4 describes the evaluation and prediction system architecture and its components. Section 5 discusses our experimental evaluation. Section 6 describes the platform created to generate the data and visualize them. We conclude in Section 7.

Chapter 2

Literature Review

There have been a great number of research papers in the area of trajectory mining. I bring some examples of them here. Sawas et al. [39] efficiently discover trajectories of objects that are found in close proximity to each other for a period of time by mining group patterns of moving objects. Pechlivanoglou et al. [36] devise a method that is able to simultaneously evaluate node importance metrics for all moving objects in a trajectory network. Mehmood et al. [31] use trajectory big data to automatically and accurately learn latent semantic relationships between different geographical areas as revealed by patterns of moving objects over time. There are also studies to better evaluate and improve the road traffic circulation going around in cities. Several subjects have to be developed to serve the purpose of improving traffic conditions. Three of the most important topics are traffic monitoring, route planning, and travel time estimation. In this section, we will be demonstrating the different methods that have been utilized to enhance traffic conditions. For this purpose, the first step is to be able to collect proper data, and that is where traffic monitoring studies come in handy.

The topic of traffic monitoring and optimum route detection has been a popular domain of research in the last decade. Having the number of vehicles passing by each part of a road over a certain amount of time is essential data for estimating the traffic flow of the road or the average speed of the cars passing the road. With this information one can compute certain representations for the current traffic of each road, also one can recognize the problematic roads or parts of the network which can be used for urban designing, and much more. There have been many studies conducted to enhance the existing methods of retrieving this data and develop new approaches and innovations to have a better outcome for travel time estimation and prediction. So far, there are different methods of collecting traffic data which include using videos from surveillance cameras to apply object detection on them, one can also extract license plate numbers in them to locate cars in different places, and there are magnetic or piezo-sensors, radars and recently image processing and machine vision technologies have become more often used in such areas.

After traffic monitoring is done the data is collected from the roads, it can be preprocessed and used for route planning and travel time prediction purposes. As mentioned in Section 1 many studies have been done in travel time estimation topic in most of which only the traffic flow of the road segments was taken into account. Recent studies show that both road segment traffic flows and intersection congestion conditions are responsible for travel time delays. Here we try to review the literature about data generation and travel time prediction for a better insight into studies done in the trajectory mining related to traffic enhancing.

2.1 Traffic Generation

Traffic generation is an effective method for simulating commuting scenarios and providing enough information to solve many traffic prediction issues. Therefore, all associated works fall into two categories: simulation and augmentation. The goal of the simulation is to produce enough data to simulate scenarios given historical observations, while augmentation entails generating data to describe inaccessible data for the other prediction problems.

2.1.1 Data Simulation

The majority of researchers are interested in the platform construction of transportation settings. They use the Bayes method, in particular, to compute similar data distributions given historical traffic information, then by using the distribution simulate various traffic conditions. For instance, Brinkhoff et al. [5] created a platform for producing moving objects by combining a real network with user-defined attributes of the intended dataset. A simulator is presented in [22] to assist in the preparation and execution of traffic scenario simulation, which involves a network, demand, and traffic generation. Similarly, Lon et al. [46] created a specialized framework for testing pickup-and-delivery algorithms. Adnan et al. [1] created a simulator to simulate millions of agents through a wide variety of mobility decisions. Finally, the simulator proposed in [32] is tailored to ridesharing by incorporating elements and procedures popular to ridesharing algorithms. Based on the link among cellular data and traffic condition, a simulation approach is provided that uses cellular data to detect road congestion on urban arterials in [23]. The notions of a Markov random walk, which outlines the movement of a single car, and Markov traffic, which characterizes the overall traffic on a road network are introduced in [4].

2.1.2 Data Augmentation

On the other hand, several people research how to produce individual data to solve other estimation problems. Wang et al. [48], for example, created routes to predict the travel time from an origin to a destination. To estimate the travel time, they use the kNN method to find the nearest historical path whose origin point and destination point are close to the given origin point and destination point. However, the kNN method cannot be used when historical information is insufficient. Song et al. [41] used GAN (generative adversarial networks) to create human mobility routes to solve this problem. They created two representative discriminator and generator networks, with the discriminator network containing four layers of CNN to capture important position features. On the other hand, some people concentrate on spatiotemporal data processing to create fake data to substitute real data to prevent privacy disclosure. For example, Wu et al. [55] use RNN to simulate trajectory data and thus encode a trajectory into a secret code. They treat each trajectory as a road series and then take full advantage of RNN's power to catch the variable length of the series. Meanwhile, when modeling trajectories, they take topological structure limitations on road networks into account.

2.2 Travel Time Prediction

The Travel-Time-Prediction issues tend to forecast specific future traffic states. we divide the travel time prediction problem into four subcategories: Origin-Destination (OD) Travel Time, Path Travel Time, Network Flow, and Traffic Speed. Existing related work can be classified into two parts: non-learning and learning approaches. Learning methods are further subdivided into traditional and deep learning approaches. In particular, these approaches use a variety of techniques. Non-learning approaches include, for example, kNN and historical average, while conventional learning methods contain regression, decision tree, and hidden Markov model. Furthermore, when evaluating these methods, five characteristics (road map, environmental conditions, spatial information, temporal information, and nonlinearity) are taken into account. To begin, when dealing with traffic prediction on roads or intersections, the configuration of the road map is an important constraint. Second, environmental data, such as weather, are crucial in traffic prediction. Third, spatial properties (such as POIs, paths, and maps) affect traffic. For example, transportation in the downtown area is very different from transportation in the suburbs. Fourth, temporal properties (such as holidays and events) can be useful for improving the accuracy of traffic prediction. For example, traffic patterns on weekends vary from those on weekdays. Fifth, when predicting potential traffic, there is a dynamic nonlinearity relation among inputs and outputs, so managing nonlinearity is one way to test the effectiveness of various forecasting approaches. In summary, we conduct the following review of literature based on the five features mentioned above.

2.2.1 Origin-Destination Travel Time

The goal of the OD-Travel-Time problem is to predict the travel time for a given origin-destination input, which includes an origin point, a destination point, and a departure time. The authors of [48] first used the kNN method to select historical trajectories with origin point and destination point that are close to the given origin-destination input, and then calculate the mean value of the chosen trajectories as the predicted result. Later on, some people used deep neural networks to solve the issue. In [21], the multilayer perceptron (aka. a multilayer fully connected neural network) is used to predict the origin-destination travel time. The researchers, in particular, use multilayer perceptron to predict travel distance given origin point and destination point and use multilayer perceptron to predict travel time provided the predicted distance and given departure time. However, since these approaches disregard certain features (for example, the structure of the road network), other deep neural networks are used to solve this issue. Li et al. [25], for example, utilized the residual neural network (ResNet) to encode given origin-destination input, and also road network characteristics, spatial and temporal properties, and so on. Given the utility of historical trajectories, Yuan et al. [61] used LSTM and CNN methods to create an auxiliary model to encode historical trajectories, allowing the approximate travel time to be accurately associated with a trajectory. Likewise, they regard environmental data characteristics, temporal and spatial attributes, like road networks.

2.2.2 Path Travel Time

The Path-Travel-Time problem is characterized as estimating travel time on road networks for a given route. As a result, all current works take into account road networks. Initially, Rahmani et al. [38] used kNN techniques to pick nearest neighbors of historical sub-trajectories to calculate the travel time, similar to origin-destination travel time prediction. Given the inefficiency of using historical data due to its scarcity, Wang et al. [51] used a three-dimensional tensor to model various drivers' travel times on various road segments at different times and then used a context-aware tensor decomposition (TD) method to fill in the tensor's missing values. Others consider the problem to be a linear regression problem[19, 63], which relates to learning-based methods and takes the specified route as input. The writers of [63], on the other hand, found the temporal dynamic, that would learn different weights for various time slots. However, since they both learn the linear weight of matching regression models, they are unable to manage nonlinearity. Other machine learning models, such as decision tree [13] and hidden Markov model [57], are used to address this problem. They divide the entire path into a series of links and then predict the travel time for each connection. The authors of [13] estimated each relation independently using boosting methods, whereas the authors of [57] modeled the entire sequence using the HMM technique. However, certain useful attributes, such as spatial and temporal attributes, are overlooked by conventional approaches. As a result, some works concentrate on using deep learning methods (e.g., CNN and LSTM) to address the problem. For example, in [47, 62], the authors first considered the given path as a series of segments. Then, they used the CNN model to encode every segment to capture local spatio-temporal correlations, after which they used the RNN model to encode the entire path. They also encode external data (for example, environmental data, spatial and temporal characteristics) to improve prediction. additionally, there is a related work [52] that solves the problem using the Wide-Deep model. They split inputs into various components, which are encoded by various wide (e.g., affine transformation) and deep (e.g., MLP and LSTM) models. Finally, some researchers would suggest the distribution of travel time over the distribution of value. Hunter et al. [18], in particular, modeled the path using a generative distribution model and used the expectation-maximization (EM) to learn the model's parameters. Likewise, Asghari et al. [2] learned the commute time probability distributions for every link on the road map from historical information and jointly calculated the distribution for the entire route. The writers of [11], on the other hand, stopped splitting trajectories into small pieces and instead allocated distributions to paths rather than just links. In addition, the authors of [24] used deep learning methods to produce probability parameters for associated generative models.

2.2.3 Network Flow

The Network-Flow issue is concerned with the flow of traffic across each intersection on a transport network. Other conventional learning approaches exist in addition to general time series forecasting approaches (e.g., HA and ARIMA). Jin et al. [20], for example, use principal component analysis and support vector regression techniques to forecast network traffic data. PCA is used to minimize the dimension of traffic information by producing eigen-flow data. Following that, they use SVR to estimate the eigen-flow data, which they then use to recreate the flow data. Tang et al. [43] still use the SVR approach to address the problem, but they improve it with denoising strategies. To enhance estimation accuracy, they merge a type of denoising algorithm (ensemble empirical mode decomposition) and the fuzzy C-means neural network (FCMNN) [44]. Yan et al. [56] use a weighted Frobenius norm to predict resemblance between multivariate time series, with the weights calculated by the PCA method, to forecast multivariate traffic flows. Most people have recently reconsidered traffic flow forecasting using deep architecture models. Lv et al. [28] begin by learning generic traffic flow attributes with a stacked autoencoder model that is trained greedily layer-wise. Later, the authors of [12, 8, 15] use GCN models to forecast network flows by taking into account the nature of road networks. Fang [12] created the spatio-temporal module to encode historical traffic information, which includes a multi-resolution temporal module and a global correlated spatial module. Wang et al.[8] suggest a two-stream network in which the first stream refers to a novel matrix-based spatio-temporal convolutional layer aimed at extracting features from a graph representation of road network traffic flow, and the second stream predicts complex graph structures, which are then fed into the first stream. Guo et al. [15] offer two modules for encoding historical data: The first section makes use of the attention mechanism to record changing spatio-temporal correlations in network traffic, while the second section makes use of the GCN methodology to record spatial patterns and ordinary regular convolutions to represent temporal aspects. Furthermore, some authors [26, 50] address the sequential aspects of the previous network flows, therefore they add RNN techniques to encode past data. Li et al. [26] specifically characterize network traffic as a diffusion process on a directed graph and propose DCRNN, a deep learning architecture for traffic prediction that includes both spatial and temporal variables in network traffic. Wang et al. [8], on the other hand, use a spatial GNN to encode historical information before employing a GRU model to encode the entire sequence. Lastly, they employ the transformer to encode the GRU's output. Finally, the meta-learning technique is used to capture the dynamic relationship between traffic flow data [33]. They have the advantage of taking into account the spatial features of road networks.

2.2.4 Traffic Speed

The goal of the Traffic-Speed issue is to predict the speed of vehicles on roads. Like other traffic prediction challenges, instead of employing basic time series prediction techniques (e.g., HA and ARIMA), numerous deep learning techniques have lately been applied. To begin, some people merely explore using various deep models to encode historical network traffic. For instance, Ma et al. [30] use a two-dimensional time-space matrix encoded by the CNN model to translate spatiotemporal traffic data into visuals describing the time and spatial relations of traffic flow. Cui et al. [10] present a deep-stacked LSTM model for predicting network-wide flow speed that takes into account forward and backward correlations in time series data. A bidirectional LSTM layer is also used to extract geographical features and bidirectional temporal correlations from historical data. The authors of [29, 49] make use of RNN and CNN algorithms by rationally combining them. They specifically use the CNN approach to extract topology-aware characteristics, and then the frequency and context variables are addressed to further enhance precision by using the LSTM model. Tang et al. [45] offer an evolving fuzzy neural network with 2 suggested learning processes, the first being to cluster inputs and the second being to optimize parameters in Takagi–Sugeno-type fuzzy rules, to anticipate traffic speed for several steps ahead. They also have the effects of recurrent elements in raw speed data considered, as in [59]. However, many contextualized elements, such as geographical and temporal aspects, are ignored by the methods described above. As a result, Liao [27] considers several implicit but critical criteria for forecasting traffic speed, and they combine these data as follows. Initially, they include offline geographical and social factors, such as road geography or public social activities. They encode the data using the GCN model. Second, they include online crowd questions, that are encoded by LSTM as a series.

Chapter 3

Preliminaries and Problem Definition

In this section, we introduce notation and preliminaries of the problem of interest, as well as formal problem definitions.

3.1 Preliminaries

Definition 1 (Road network). The road network can be modeled as a directed multigraph (*multidigraph*), where edges have their own identity. This means that edges are primitive entities (just like nodes) and that when multiple edges connect two nodes, these are different edges. Formally, we define a road network as a multidigraph $G := (\mathcal{V}, \mathcal{E}, s, t)$, where:

- $\mathcal{V}$ is a set of nodes that represent junctions of the road network. A junction typically has two or more edges adjacent to it or a single adjacent edge (i.e., representing an edge entering or exiting a junction); they maintain right-of-way information (e.g., intersections, lane splits, sharp turns, etc.).

- $\mathcal{E}$ is a set of edges representing a connection between two junctions; they maintain shape information (e.g. one-way street, one direction of highway segment).
- Two functions: a function $s : \mathcal{E} \rightarrow \mathcal{V}$ assigning the *source node* of the edge, and another function $t : \mathcal{E} \rightarrow \mathcal{V}$ assigning the *target node* of the edge.

As G is a multigraph, there can be multiple edges connecting two nodes. Functions s and t are used to define these edges. In the road network these represent *road lanes* that maintain speed and length information (see an example in Fig. 1.1).

Definition 2 (Road intersection). A road intersection is defined as a node $v \in \mathcal{V}$ of the road network $G := (\mathcal{V}, \mathcal{E}, s, t)$, with a node degree d_v greater than two ($d_v > 2$) (see Fig. 1.1).

Definition 3 (Measures of effectiveness). We briefly introduce the measures of effectiveness (MOEs) that our system evaluates and monitors in real-time and can be used to assess the operational performance of the road network:

- `Capacity (#vehicles)`: The number of vehicles that can exist in-system.
- `Throughput (#vehicles/sec)`: The number of vehicles passed through in a time unit.
- `Delay (sec)`: The additional time that vehicles going through experienced compared to ideal free-flow time.
- `Mean system speed (meters/sec)`: The average speed of vehicles in-system.
- `Travel Time Index (TTI)`: An index defined as the `average travel time` divided by the `free-flow travel time`. The higher the `TTI` the larger the congestion at this

part of the road network.

These MOEs are designed to help the domain experts and decision makers rapidly assess the state of the road network system and identify key intersections that need improvements.

3.2 Problem Definition

Let a road network $G := (\mathcal{V}, \mathcal{E}, s, t)$, an observation time period $[0, T]$, and a set of trajectories $\mathcal{T} = \{C_i\}$, representing all instances of vehicles that have been in G during $[0, T]$, where $C_i = \{(t_i, e)\}$ is a registry of vehicles found at edge $e \in \mathcal{E}$ at time $t_i \in [0, T]$. Now, given a road intersection $v \in \mathcal{V}$ of the road network G and $\mathcal{T}$, we want to:

(i) compute the `TTI` of the intersection during $[0, T]$;

(ii) forecast the `TTI` of the intersection for the period $[T, T + \Delta]$, where $\Delta > 0$.

Note that (i) describes a real-time data analytics problem (stream analytics), while (ii) describes a prediction problem.

Chapter 4

Methodology

In this section, we provide an overview of the system that allows us to evaluate and forecast the operational performance of road intersections. The complete process is depicted in the diagram of Fig. 4.1 and includes two main components.

(A) **Road network MOEs evaluation**. Given the (road network and traffic) data as input, this component is responsible for computing the measures of effectiveness (MOEs) of any road intersection at different levels of abstraction, including *edge-level* and *higher-level* MOEs.

(B) **Prediction model**. Given the computed MOEs as input, this component is responsible for forecasting the operational performance of any road intersection in the near future, by employing a prediction model.

We elaborate on the inputs and these components in the next paragraphs.

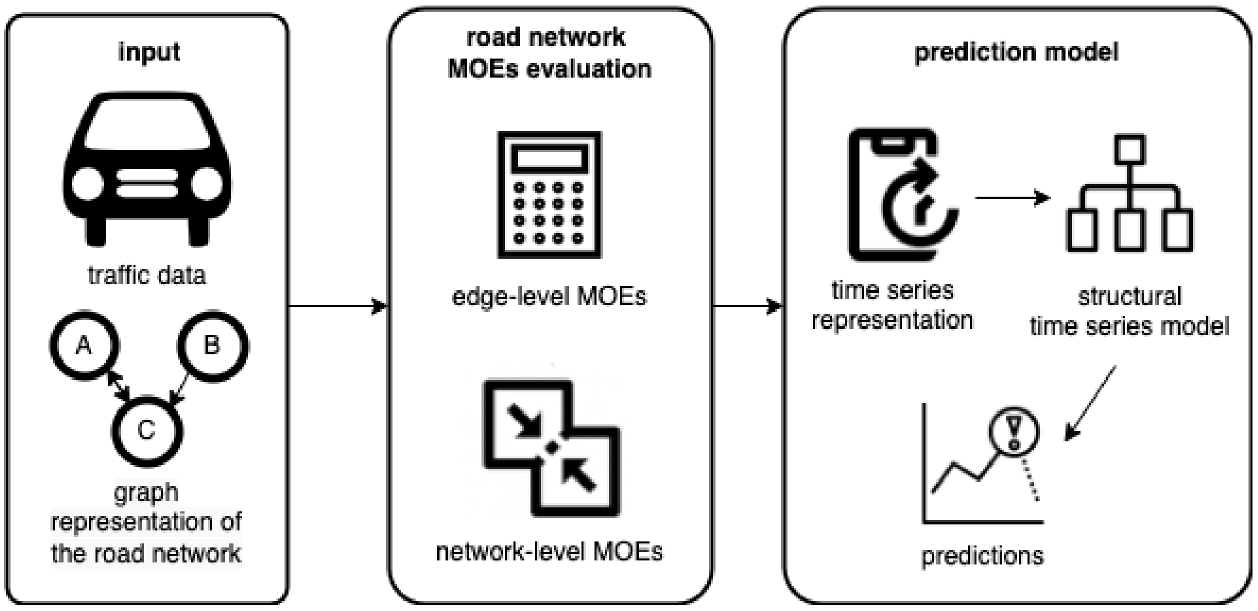

input
traffic data
A
B
C
graph representation of the road network
road network MOEs evaluation
edge-level MOEs
network-level MOEs
prediction model
time series representation
structural time series model
predictions

Figure 4.1: Illustration of the high-level evaluation and prediction system architecture.

4.1 Data Representation

The input to our method consists of two types of data representation.

(A) **Traffic data**. The data showing the traffic flow of the cars moving on the roads.

(B) **Graph representation of the road network**. The graph representation of the road network extracted from a map.

We explain more details about these input data in the next paragraphs, but they are more elaborated on in Section 6.

4.1.1 Traffic Data

Traffic data is the information about all the cars moving inside the given network. For the given time period, we have to access the information about the direction of the cars, their location, and their speed in any given second.

4.1.2 Graph Representation of the Road Network

In order to work with the traffic and the network data, there has to be something to model the network in a structured way. We have chosen a directed weighted graph as the model to represent the road network. In this model, the edges are the road, also known as links. The nodes are the points where the roads intersect or where there is a change in the road like the speed limit or the number of lanes. Each edge has a number of lanes that define the direction of the edge. If the lanes inside the edge are from point a to point b, then the direction of the edge would be from node a to node b. The edges also store the attributes of the link inside them like the number of lanes, the speed limit, the shape of the road, and such. As an example, you can see the graph representation of an intersection in Fig. 6.2 (c).

4.2 Road Network MOEs Evaluation

In Section 3 we formally defined the MOEs that are monitored in our system. These MOEs are primarily defined at the level of a single road segment (edge). However, it is possible to define higher levels of system abstractions that include more complex road network structures (see

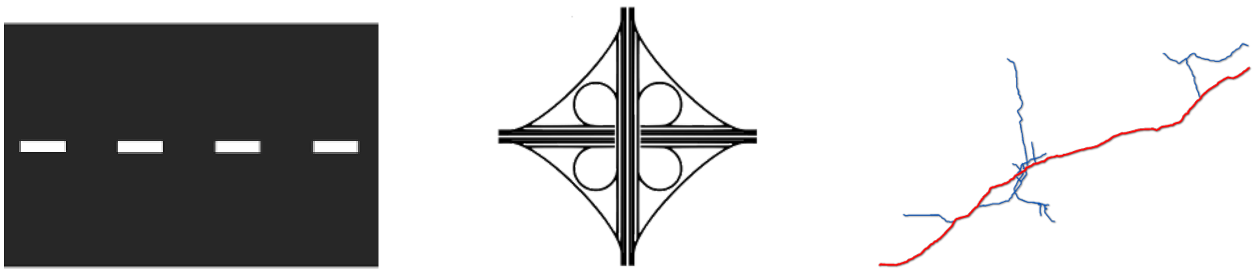
Fig. 4.2). For example, Fig. 4.2(a) depicts a *micro-level* system (a single road segment), Fig. 4.2(b) a *meso-level* system (a single intersection), and Fig. 4.2(c) a *macro-level* system (an entire highway including multiple intersections). Our system can accommodate monitoring of MOEs at different levels of system abstraction. We elaborate below how MOEs are computed for these cases.

4.2.1 Edge-level MOE Evaluation

Traffic flow data is processed in a temporal order (for the observation time period $[0, T]$) and MOEs are computed in a pseudo-streaming fashion for each single road segment (edge) as a function of time. In particular, for each edge we evaluate the capacity of the edge (`capacity`), the number of vehicles passed (`throughput`), the speed of the vehicle (`mean system speed`), and the delay of the vehicles (`delay`). We also compute the travel time index `TTI` for each edge. At the end, for each edge, each MOE defined in Section 3 is represented as a time series, a sequence of data points occurring in successive order in $[0, T]$.

4.2.2 Network-level MOE Evaluation

Now that all edge-level MOEs are computed, we can evaluate MOEs for higher-level system abstractions or otherwise *network-level MOEs*. A higher-level system abstraction is a more complex system that includes multiple junctions and edges. Formally, given the multidigraph $G := (\mathcal{V}, \mathcal{E}, s, t)$ we define a system abstraction as a subgraph $G' := (\mathcal{V}', \mathcal{E}', s, t)$ formed from a subset of the vertices and edges of G, so $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$, respectively, such that G' is



(a) micro-level (b) meso-level (c) macro-level

Figure 4.2: Different levels of system abstraction.

connected. G' is connected if every pair of vertices in G' is connected, or otherwise there is a path between every pair of vertices.

Now given a subgraph $G' := (\mathcal{V}', \mathcal{E}', s, t)$ representing a higher-level system abstraction, we can compute the network-level MOEs by aggregating edge-level MOEs of all edges $\mathcal{E}'$ belonging to the subgraph G'. Specifically, the `capacity` would be the sum of the `capacity` values of all edges. For computing the `throughput`, `delay`, `mean system speed` and `TTI`, we consider the union of all the incoming and outgoing vehicles and all the vehicles inside the custom system and assume they belong on a single edge. Then, MOEs are defined for that edge. We have created a class called multi edge system which has several edge systems inside and it overwrites the method to update the existing cars so that whenever a car is inside any edge of this system we consider the distance it takes and we only remove it when it leaves all of the existing edges in the multi edge system. This system includes all the paths going from every entrance of intersections to every exit of them, ans also the whole intersection.

For example, if we want to compute the MOEs of a specific intersection, we first need to define a subsystem for each possible *road intersection direction*; a road intersection direction is a subgraph (representing a path) connecting an incoming edge to an outgoing edge of the intersection. Then, MOEs of each subsystem will need to be combined to provide the MOEs for the whole intersection.

4.3 Structural Time Series

An important feature of our system is the ability to forecast the operational performance of a road intersection in the near future. Out of all the MOEs evaluated, the `TTI` is the most comprehensive one, we therefore focus on designing and evaluating predictive models for that measure. As the continuous `TTI` computation represents a time series, the prediction problem reduces to that of time series forecasting. Below, we first elaborate on *time series smoothing* based on a *moving average technique*. Then, we discuss the proposed prediction model based on *structural time series*.

The time series of the `TTI` measure is characterized by short-term fluctuations, represented by altering rising and falling values. It is easy to see why this is the case in an intersection with a traffic light. When the light turns red, the vehicles stop moving, while when it turns green they start moving and/or traversing the intersection without delays. A common technique for smoothing the time series fluctuations is the "moving average" model. A moving average smooths a series by consolidating several single data points into longer units of time by taking their average. The formula of a moving average is as follows:

$$\bar{y}_t = \frac{y_t + y_{t-1} + \dots + y_{t-w-1}}{w} \tag{4.1}$$

where y is the variable, t is the current time step, and w is the size of the the *span* (or window) of the moving average, controlling the number of time steps contributing to the average. The larger the w, the smoother the series become. In our setting, where each time step represents five (5) minutes, we set $w = 6$; that reduces the fluctuations in the time series as each data point is now representing a 30 minute time interval. An illustration of time series smoothing using a moving average technique is shown in Fig. 5.1.

4.3.1 Overview of the Method

Although forecasts of future happenings are essentially uncertain, forecasting is an important aspect of long-term planning.

In our system, we have designed and developed *structural time series* models [17] for predicting the `TTI` of a road intersection in the near future. A structural time series (STS) model represents a time series as the sum of several simple components as in Eq. (4.2).

$$f(t) = f_1(t) + f_2(t) + ... + f_n(t) + \varepsilon; \varepsilon \sim N(0, \sigma^2) \tag{4.2}$$

Each component is a time series regulated by a specific structural assumption. Structured time series can typically provide good projections by allowing modellers to include assumptions about the processes generating the data. We can interpret the model's assumptions about prediction by visualising the structural decomposition of historical data and future forecasts. Furthermore, structural time series models employ a probabilistic formulation that handles missing data naturally and provides a formal measurement of uncertainty. Our model, considers the following decomposed components, also shown in Fig. 4.3 for a more clear

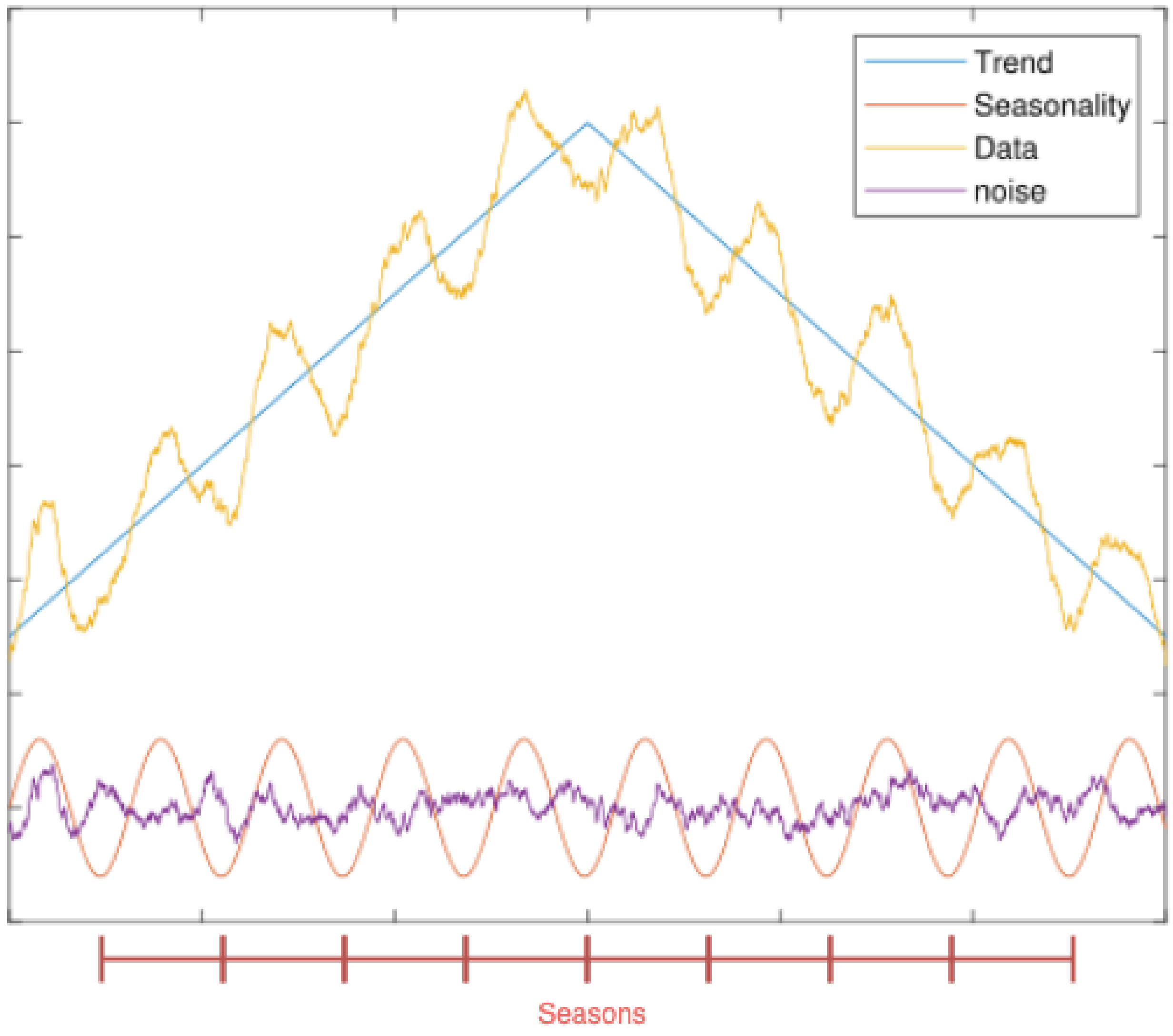


Figure 4.3: Structural time series decomposition.

intuition:

- `trend`: A trend is a data pattern that depicts the progression of a series to substantially increase or decrease over time. A trend is seen when the slope of a time series increases or decreases. A trend usually lasts for a short period of time before disappearing; it does not reoccur.

- `seasonality`: A flexible periodic function is required to encode arbitrary periodic patterns. The Fourier series can be used to approximate any periodic function. A Fourier series is a weighted combination of sine and cosine terms that increase in frequency. The required amount of Fourier terms for a component can be determined by performing a brute hyper-parameter search to get the optimal performance. We are using two seasonal components in this work: *the hour-of-day effect* and *the day-of-week effect.*

- `external regressors`: The prediction is not solely dependent on time, but also on the values of additional regressors. the `TTI`s of the close-by edges that have correlations with the `TTI` of the whole intersection can be used with a one week lag because of the seasonal attribute of the time series. These additional covariates can be included via a linear regression model.

- `noise`: An autoregressive component is used to model any unexplained residual effects. This component maintains bounded variance over time, and can be used for modeling time series with a level or slope that evolves according to a random walk or other processes.

We modeled time series as a sum of local linear trend, seasonal, linear regression, and autoregressive models. Then, we used Kalman filter algorithm [54] and variational inference technique [34] based on the training data and the structural model to learn the model parameters and make predictions. For simplicity we only take the local linear trend in consideration and explain the next steps. The other parts of the model can be added to the system accordingly. Here are the steps we took to perform this operation:

1. **Local linear trend**. We defined the local linear trend model as a parameterized linear dynamical system.

2. **Likelihood calculation**. The likelihood p(y|z) was then calculated using Gaussian linear transformation and the Kalman filter technique.

3. **Posterior approximation**. Using a distribution q(z), we employed variational inference to estimate the posterior p(z|y). The variational distribution with variational parameters is represented by q(z).

4. **Probability density transformation**. We used probability density transformation to ensure that p(z|y) and q(z) had the same variable domains.

5. **PDF distance minimization**. To minimise the KL-divergence between q(z) and p(z|y), we employed ELBO(q(z)) as the optimisation target. The optimisation algorithm is gradient descent.

6. **Sample average**. To estimate the integration inside formulation of ELBO(q(z), we used sample average. The samples are drawn from the q(z) distribution.

7. **Reparameterization approach**. We used the reparameterization approach to create q(z) samples from simpler distribution samples. This simplified distribution is not dependent on the parameters that we are optimising. Gradient descent can proceed in this manner.

8. **Predicting the future**. With having the parameters of Kalman filter in hand, we run Kalman filter and get values for the next timesteps.

Following, we elaborate on each of these steps in more detail.

4.3.2 Local Linear Trend

The local linear trend has the following definition with t as the timestep:

$$\begin{aligned} slope_t &= slope_{t-1} + \epsilon_1, & \epsilon_1 \sim \mathcal{N}\left(0, \sigma^2_{slope}\right) \\ level_t &= level_{t-1} + slope_{t-1} + \epsilon_2, & \epsilon_2 \sim \mathcal{N}\left(0, \sigma^2_{level}\right) \\ y_t &= level_t + \epsilon_3, & \epsilon_3 \sim \mathcal{N}\left(0, \sigma^2_{obs}\right) \end{aligned}$$

σ_{level}, σ_{slope}, and σ_{obs} are the model parameters which we need to learn from the observations. The matrix form of the local linear trend is written in the following equation:

$$\begin{aligned} x_t &= Ax_{t-1} + w_t \quad \text{the state variable} \\ y_t &= Hx_t + v_t \quad \text{the observation variable} \end{aligned}$$

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad x_{t-1} = \begin{bmatrix} level_{t-1} \\ slope_{t-1} \end{bmatrix}, \quad x_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

$$\begin{aligned} w_t \sim& \mathcal{N}\left(0, \Sigma_w\right), \quad \text{where } \Sigma_w = \begin{bmatrix} \sigma^2_{level} & 0 \\ 0 & \sigma^2_{slope} \end{bmatrix} \\ v_t \sim& \mathcal{N}\left(0, \Sigma_v\right), \quad \text{where } \Sigma_v = \begin{bmatrix} \sigma^2_{obs} \end{bmatrix} \end{aligned}$$

In order to learn the parameters we use the Bayesian model, so we need to calculate the posterior distribution of the model parameters given observed data.

$$p(z \mid y)=\frac{p(y \mid z) p(z)}{\int p(y \mid z) p(z) \mathrm{d} z} \quad \text{the posterior density}$$

$$z=[\sigma_{level}, \sigma_{slope}, \sigma_{obs}] \quad \text{the vector of model parameters}$$

$$Y=Y_{1:t} \quad \text{the vector of the observations}$$

We assume that the priors are drawn from a LogNormal distribution and are independent of one another. As a result, the prior p(z) is equal to the product of the three distinct LogNormal probability densities:

$$\begin{aligned} p(z) & =p\left(\sigma_{\text{level }}, \sigma_{\text{slope }}, \sigma_{obs}\right) \\ & =p\left(\sigma_{\text{level }}\right) \cdot p\left(\sigma_{\text{slope }}\right) \cdot p\left(\sigma_{obs}\right) && \text{independence assumption} \\ & =LN\left(\sigma_{\text{levei }} ; \cdot\right) LN\left(\sigma_{\text{slope }} ; \cdot\right) LN\left(\sigma_{obs} ; \cdot\right) \end{aligned}$$

4.3.3 Likelihood Calculation

We denote the likelihood and apply the chain rule to it as follows:

$$\begin{aligned} p(y \mid z) & =p\left(y_1, y_2, \ldots, y_T \mid z\right) \\ & =p\left(y_1 \mid z\right) p\left(y_2 \mid y_1, z\right) p\left(y_3 \mid y_{1:2}, z\right) \cdots p\left(y_T \mid y_{1:T-1}, z\right) \end{aligned}$$

Because all the production terms have the same structure, if we can describe the analytical formula for one arbitrary term, we can do the same for all of them. If we make the following assumption:

$$p\left(x_{t-1} \mid y_{1:t-1}, z\right) = \mathcal{N}\left(\mu_{t-1}, \Sigma_{t-1}\right) \quad \text{assumption}$$

Then we can conclude the following because of the rule of Gaussian linear transformation:

$$p\left(x_t \mid y_{1:t-1}, z\right) = \mathcal{N}\left(A\mu_{t-1}, A\Sigma_{t-1}A^\top + \Sigma_w\right) \quad \text{conclusion}$$

$$p\left(y_t \mid y_{1:t-1}, z\right) = \mathcal{N}\left(H\left(A\mu_{t-1}\right), \quad H\left(A\Sigma_{t-1}A^\top + \Sigma_w\right)H^\top + \Sigma_v\right)$$

This is the analytical form for an arbitrary term in our likelihood p(y|z). We have the following rules using Kalman filtering technique:

$p\left(x_{t-1} \mid y_{1:t-1}, z\right)$ assumption

$\downarrow$

$p\left(x_t \mid y_{1:t-1}, z\right)$ Gaussian linear transformation via system equation

$\downarrow$

$p\left(y_t \mid y_{1:t-1}, z\right)$ Gaussian linear transformation via system equation

$\downarrow$

$p\left(x_t, y_t \mid y_{1:t-1}, z\right)$ multivariate Gaussian: joint rule

$\downarrow$

$p\left(x_t \mid y_{1:t}, z\right)$ multivariate Gaussian: conditional rule

The first three steps provide the model's prediction for the next system state and observation. The latter two steps update this belief using the actual observation.

4.3.4 Posterior Approximation

Because calculating the marginal likelihood analytically is difficult, we must approximate the posterior numerically. The variational distribution is used to estimate the posterior distribution p(z|y).

$$\begin{aligned}
& q(z) \\
& = q\left(\sigma_{\text{level}}, \sigma_{\text{slope}}, \sigma_{obs}\right) && \text{expand z} \\
& = p\left(\sigma_{\text{level}}\right) \cdot p\left(\sigma_{\text{slope}}\right) \cdot p\left(\sigma_{obs}\right) && \text{mean-field definition} \\
& = \mathcal{N}\left(\sigma_{\text{level}}; \mu_l, \sigma_l^2\right) \cdot \mathcal{N}\left(\sigma_{\text{slope}}; \mu_s, \sigma_s^2\right) \cdot \mathcal{N}\left(\sigma_{obs}; \mu_o, \sigma_o^2\right) && \text{individual density}
\end{aligned}$$

4.3.5 Probability Density Transformation

q(z) has its own set of parameters: vp=[μ_l, μ_s, μ_o, σ_l, σ_s, σ_o] called variational parameters. q(z) has a straightforward structure and analytical form, and shares the same variables as p(z|y). However, the domains of the variables in q(z) differ from those in p(z|y). To tackle this problem, we transform the posterior p(z|y) of a probability density function of variables over the R+ domain into a probability density function of variable vector u over the R domain, p(u|y). By using exponential as the transformation function from variables in u to variables in z we have:

$$\sigma_{level} = e^{u_{level}}$$

$$\sigma_{slope} = e^{u_{slope}}$$

$$\sigma_{obs} = e^{u_{obs}}$$

And it comes down to p(u|y) = m p(y|u) p(u) for the set of the variable u over the full real domain R. From now on, we should use p(y|u)p(u) instead of p(y|z)p(z). However, to make things simpler, let's redefine our z to u. This means from now on σ_{level}, σ_{slope}, σ_{obs}, and p(y|z)p(z) are in the transformed domain.

4.3.6 PDF Distance Minimization

To quantify the level of overlapping between q(z) and p(z|y), we use the KL-divergence [14]:

$$\begin{aligned} KL(q(z)\|p(z \mid y)) &= \mathbb{E}_{z\sim q(z)}\left[\log \frac{q(z)}{p(z \mid y)}\right] \\ &= \int q(z) \log \frac{q(z)}{p(z \mid y)} \mathrm{d}z \end{aligned}$$

$$q(z; vp)^* = \arg_{vp} \min KL(q(z)\|p(z \mid y))$$

The theory of variational inference tells us that minimising the KL-divergence is equivalent to maximizing Evidence Lower Bound, denoted by ELBO(q(z)) [58]:

$$\begin{aligned} &\mathrm{ELBO}(q(z)) \\ &= \mathbb{E}_{z\sim q(z)}[\log p(y \mid z)] - \mathbb{E}_{z\sim q(z)}\left[\log \frac{q(z)}{p(z)}\right] && \text{definition of ELBO} \\ &= \mathbb{E}_{z\sim q(z)}\left[\log p(y \mid z) - \log \frac{q(z)}{p(z)}\right] && \text{linearity of expectation} \\ &= \mathbb{E}_{z\sim q(z)}\left[\log \frac{p(y \mid z)p(z)}{q(z)}\right] && \text{property of log} \\ &= \mathbb{E}_{z\sim q(z)}\left[\log \frac{p(y, z)}{q(z)}\right] && \text{product rule} \\ &= \int q(z) \log \frac{p(y, z)}{q(z)} \mathrm{d}z && \text{definition of expectation} \end{aligned}$$

4.3.7 Sample Average

We can use gradient descent to maximize ELBO(q(z)) with respect to its parameters vp. Then, we can use sample average to approximate the expectation:

$$\begin{aligned}
&\nabla_{vp} ELBO(q(z)) \\
&= \nabla_{vp} \int q(z) \log \frac{p(y,z)}{q(z)} \mathrm{d}z && \text{ELBO definition} \\
&\approx \nabla_{vp} \left[\frac{1}{n} \sum_{i=1}^{n} \log \frac{p(y, Z_i)}{q(Z_i)} \right], \quad Z_i \sim q(z) && \text{expectation as sample average} \\
&= \nabla_{vp} \left[\frac{1}{n} \sum_{i=1}^{n} (\log p(y, Z_i)) - \log q(Z_i) \Big) \right] && \text{property of log} \\
&= \nabla_{vp} \left[\frac{1}{n} \sum_{i=1}^{n} \log p(y, Z_i) \right] - \nabla_{vp} \left[\frac{1}{n} \sum_{i=1}^{n} \log q(Z_i) \right] && \text{linearity of differentiation}
\end{aligned}$$

4.3.8 Reparameterization Approach

In order not to lose the parameters during sampling we should sample from a simpler distribution that does not depend on the variational parameters. We reparameterize each of the model parameters. You can see the reparameterization of σ_{level} as an example:

$$\begin{aligned}
\sigma_{\text{level}} &\sim \mathcal{N}\left(\mu_l, \sigma_l^2\right) && \text{original model parameter} \\
\theta_{\text{level}} &\sim \mathcal{N}(0,1) && \text{reparameterization variable} \\
\sigma_{level} &= \sigma_l \theta_{\text{level}} + \mu_l && \text{reparameterized } \sigma_{level} \\
r\left(\theta_{level}\right) &= \sigma_l \theta_{level} + \mu_l && \text{reparameterization function}
\end{aligned}$$

This trick turns q(z) and ELBO(q(z)) into a function of θ and vp:

$$\theta = [\theta_{level}, \theta_{slope}, \theta_{obs}] \qquad \text{vector of reparameterization variables}$$

$$q(z) = q(r(\theta)) \qquad \text{replace variable with function}$$

$$\begin{aligned}
&\text{ELBO}(q(z)) \\
&= \int q(z) \log \frac{p(y,z)}{q(z)} \mathrm{d}z && \text{definition of ELBO} \\
&= \int q(r(\theta)) \log \frac{p(y, r(\theta))}{q(r(\theta))} \frac{\mathrm{d}r(\theta)}{\mathrm{d}\theta} \mathrm{d}\theta && \text{integration by substitution} \\
&= \int \mathcal{N}(\theta; \mathbf{0}, \mathbf{1}) \log \frac{p(y, r(\theta))}{q(r(\theta))} \mathrm{d}\theta \\
&\approx \frac{1}{n} \sum_{i=1}^{n} \log \frac{p\left(y, r\left(\Theta_i\right)\right)}{q\left(r\left(\Theta_i\right)\right)}, \quad \Theta_i \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) && \text{expectation as sample average}
\end{aligned}$$

Since σ_l, σ_s, and σ_o represent standard deviations, they must be non-negative, so we use three variables with full domains to rewrite them:

$$\sigma_l = e^{\phi_l}$$

$$\sigma_s = e^{\phi_s}$$

$$\sigma_o = e^{\phi_o}$$

$$\begin{aligned} q(z) &= q\left(\sigma_{\text{level}}, \sigma_{\text{slope}}, \sigma_{\text{obs}}\right) \\ &= p\left(\sigma_{\text{level}}\right) \cdot p\left(\sigma_{\text{slope}}\right) \cdot p\left(\sigma_{obs}\right) \\ &= \mathcal{N}\left(\sigma_{\text{level}}; \mu_l, \left(e^{\phi_l}\right)^2\right) \cdot \mathcal{N}\left(\sigma_{\text{slope}}; \mu_s, \left(e^{\phi_s}\right)^2\right) \cdot \mathcal{N}\left(\sigma_{obs}; \mu_o, \left(e^{\phi_o}\right)^2\right) \end{aligned}$$

Now the ELBO becomes a function of μ_l, μ_s, μ_o, ϕ_l, ϕ_s, and ϕ_o all of which can take values from the full real domain R.

We use gradient descent to find optimal values for vp. We first create an Adam optimizer with the ELBO as the objective function and then perform 200 gradient descent steps.

4.3.9 Predicting the Future

We can run the Kalman filter now because now we can know values of its parameters. After the Kalman filter algorithm finishes, the result is the distribution of the system state variable x_t. We can plug $x_T|y_{1:T}, z$ into the linear dynamical system equations to get predictive distributions for the state variable $x_{T+1}|y_{1:T}, z$ and observation variable $y_{T+1}|y_{1:T}, z$. And we can do it as long as the prediction horizon goes.

$$\begin{aligned} x_{T+1} &= Ax_T + w_{T+1} \quad \text{gives } p\left(x_{T+1} \mid y_{1:T}, z\right) \\ y_{T+1} &= Hx_{T+1} + v_{T+1} \quad \text{gives } p\left(y_{T+1} \mid y_{1:T}, z\right) \end{aligned}$$

$$\begin{tikzcd} p\left(x_T \mid y_{1:T}, z\right) \arrow[r] & p\left(x_{T+1} \mid y_{1:T}, z\right) \arrow[r] \arrow[d] & p\left(x_{T+2} \mid y_{1:T}, z\right) \arrow[r] \arrow[d] & \cdots \arrow[r] & p\left(x_{T+i} \mid y_{1:T}, z\right) \arrow[d] \\ & p\left(y_{T+1} \mid y_{1:T}, z\right) \arrow[r] & p\left(y_{T+2} \mid y_{1:T}, z\right) \arrow[r] & \cdots \arrow[r] & p\left(y_{T+i} \mid y_{1:T}, z\right) \end{tikzcd}$$

The process of prediction is to use the first 3 out of 5 steps in the Kalman filter algorithm to derive our belief of the system state and observation possibility distributions in the future.

Chapter 5

Experimental Evaluation

In this section, we discuss the evaluation of the forecasting component of the system. We first provide information about the data used, the evaluation metric and the sensible baseline methods evaluated. Finally, we report the results and discuss the implications for our proposed method.

5.1 Data

To evaluate the forecasting models, we employ a two-month (9 weeks) synthetic traffic flow dataset using the data generator described in Section 6 and the map of Fig. 6.2(a); we focus on the road intersection indicated with a black brush. The total population inside the map network is around $10,000$. The data is split into a training dataset (eight (8) weeks) and a testing dataset (one (1) week). Fig. 5.1 demonstrates the time series of the `TTI` measure and the effect of the smoothing based on the moving average technique. Fig. 5.2 shows the

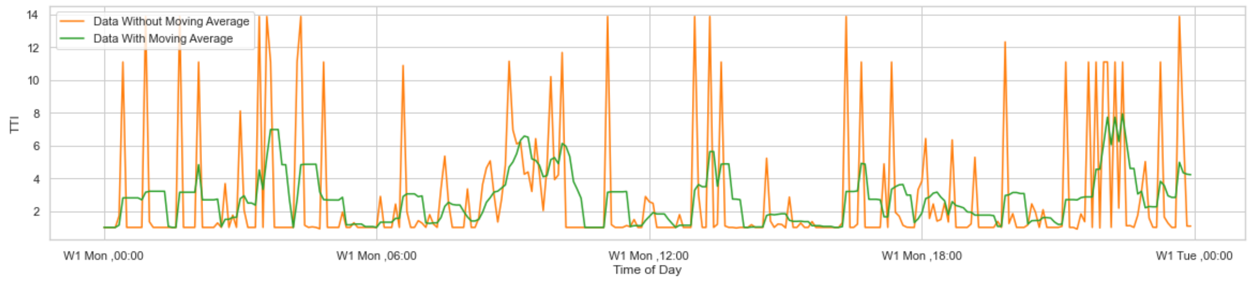


Figure 5.1: The effect of smoothing on time series data.

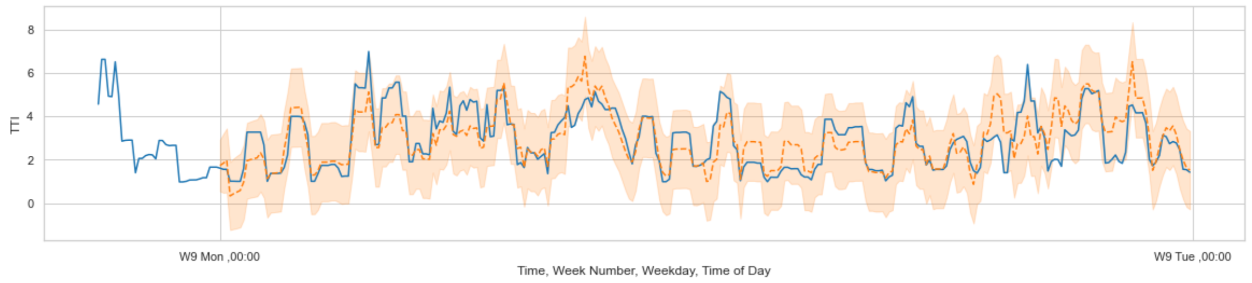


Figure 5.2: The prediction for the first day of the ninth week.

prediction against the ground truth; the orange shading is showing the standard deviation of the uncertainty. As can be seen in Fig. 5.3, the model has detected an *hour-of-day effect* and a quite smaller *day-of-week effect*, as well as an external regressor. The autoregressive process is responsible for the majority of the predictive uncertainty, which is based on its estimate of the unmodeled (residual) variance in the observed series.

5.2 Evaluation Metric

The practice of using known data to estimate the loss that a predictive model would get in available but unused observations is referred to as performance evaluation. Evaluating a predictive model's performance is a critical stage in any machine learning effort. Performance evaluation is used by practitioners to select the best model and its parameters. Performance evaluation is one of the most reliable approaches to assessing predictive model generalization capacity. Such analysis is necessary not only to choose the optimal model but also to ensure that the model solves the underlying forecasting objectives. While there are many metrics for evaluating the accuracy of a prediction, not all of them are adequate for time series data.

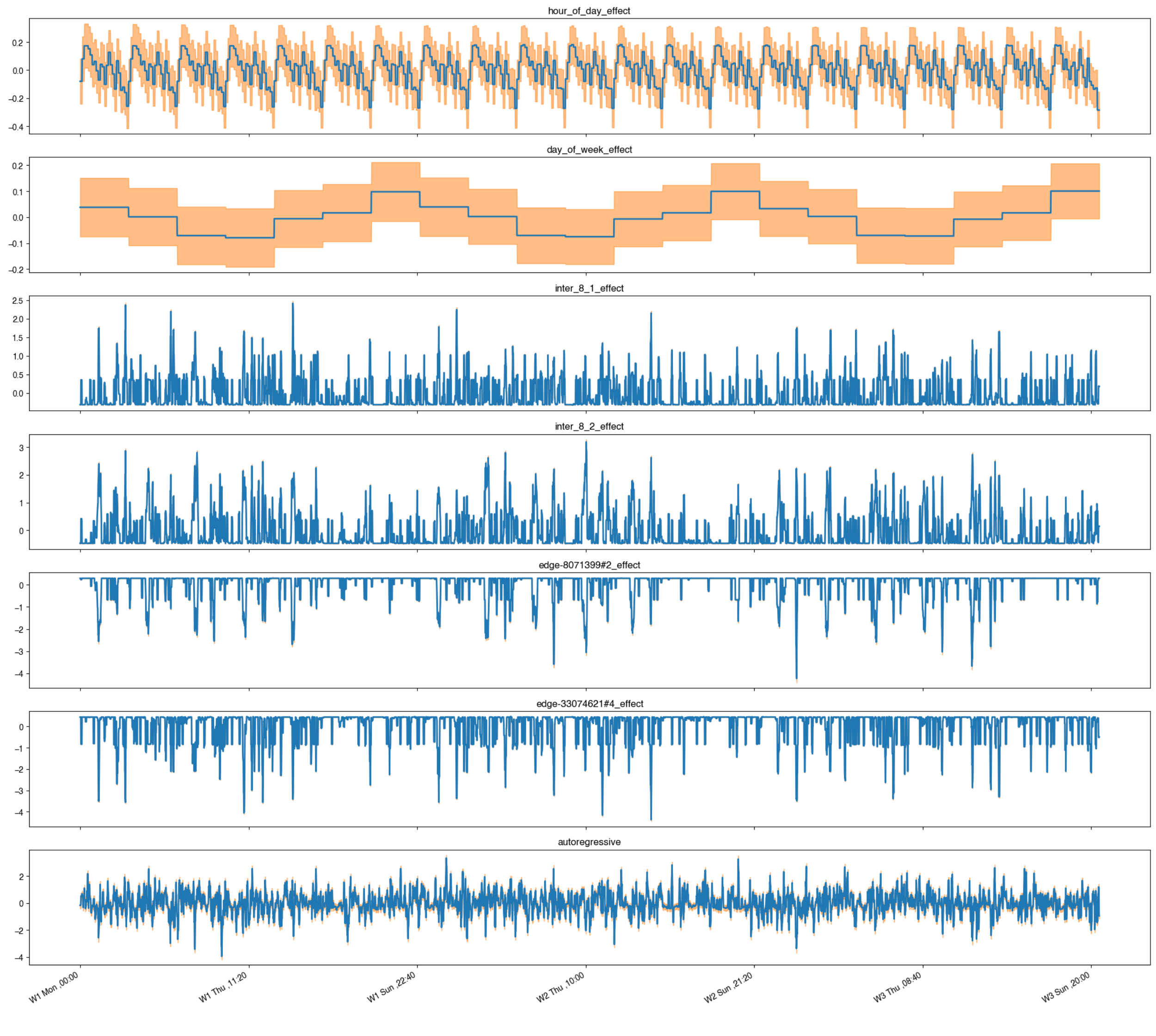


Figure 5.3: Effective components decomposition.

That is because of the chronological dependency amongst samples in the time series. For instance, standard cross-validation, the most commonly used method for evaluating prediction models, is not adequate for time series prediction. However, it can be adjusted so it can be applied to time series models as well [7]. You can see the standard cross validation technique in Fig. 5.4. In the standard version, say a 5-fold cross validation means the dataset is split into 5 parts and in each iteration one part is held out as a test set. An alternative version of cross validation [7] that is compatible with the time series is depicted in Fig. 5.5. In this version, for a 5-fold cross validation, the dataset is split into 6 parts. Then at the n-th iteration the union of the first n parts is used as the training set and the next part as the test set. So, there will be 5 errors (one for each part) and their average is reported as the model's performance. As a measure of error, we employ the *mean absolute error* (MAE)[3]. This is the easiest to calculate and interpret, and it is defined as follows.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{5.1}$$

where $\hat{y}_i$ is the prediction and y_i the true value.

5.3 Methods

The following sensible methods have been evaluated and compared to our method as baselines:

- NAIVE: It is a forecaster that makes forecasts using simple strategies based on naive assumptions about past trends continuing. To be more precise, it has three strategies: last, mean, and drift. We have chosen the mean strategy to use as a baseline in the

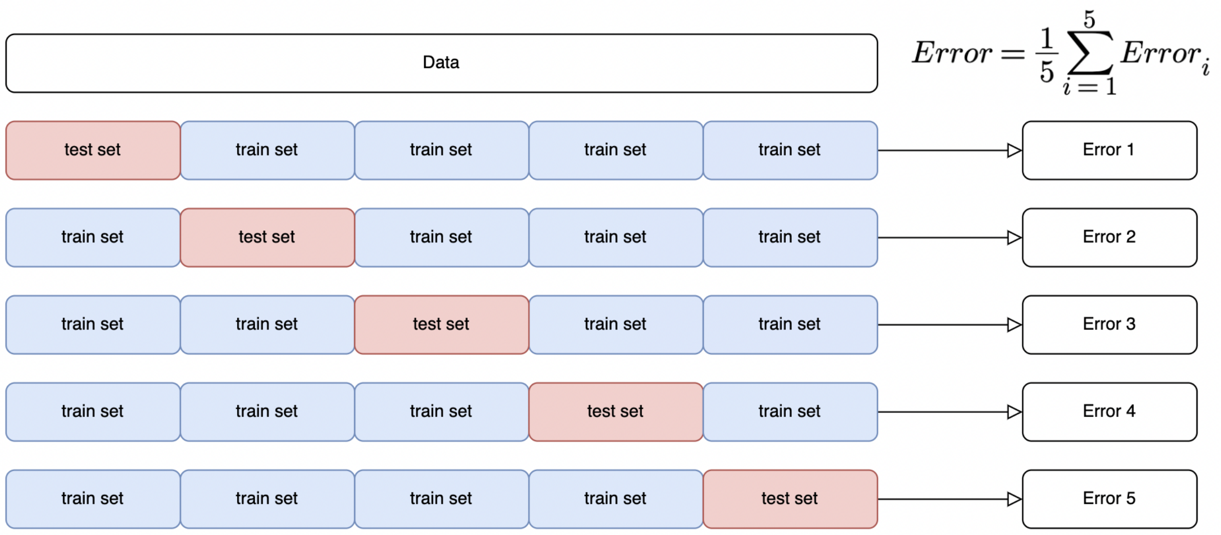


Figure 5.4: Standard cross-validation method.

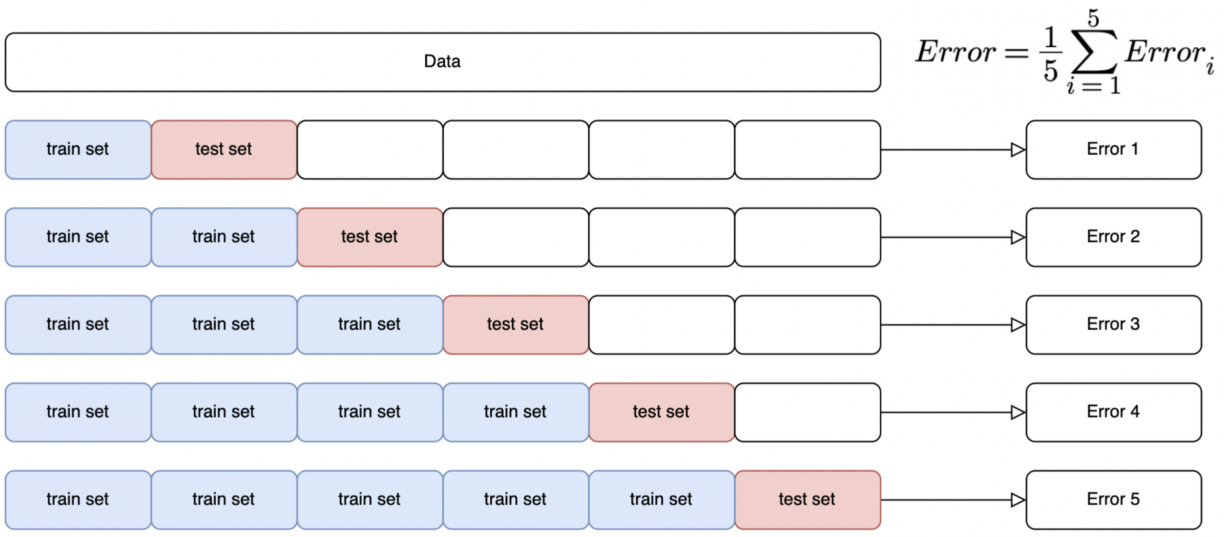


Figure 5.5: Time series cross-validation method.

current work. Basically, the mean of all values corresponding to a time point in all the previous seasons will be forecasted for each point in the future season.

- AutoARIMA: An ARIMA, or autoregressive integrated moving average, is a modification of an ARMA that is fitted to time-series data in order to estimate future points. The "AR" in ARIMA denotes that the developing variable of interest is regressed on its own lag (i.e., previously observed) values. The "MA" portion shows that the regression error is a linear mixture of error terms for which values happened both concurrently and at different points in the past. The letter "I" (brief for integrated) denotes that each value has been changed to the difference between its value and the prior values. Each of these features exists to make the model suit the data as closely as possible. Auto-ARIMA performs differencing tests like Phillips–Perron, Kwiatkowski–Phillips–Schmidt–Shin, or Augmented Dickey-Fuller to establish the order of differencing, d. Then it fits models within defined start p, max p, start q, and max q ranges. After that, it will also try to identify the best P and Q hyper-parameters after performing the Canova-Hansen to determine the best order of seasonal differencing, D.

- PolyTrend: This model Forecasts time series data with a polynomial trend. Our settings train a linear regression model with a 3rd degree polynomial transformation of the feature.

- ExpSm: Exponential smoothing is a univariate time series forecasting approach. In that a prediction is a weighted average of past observations, exponential smoothing forecasting methods are similar, but the model explicitly utilises an exponentially decreasing weight for past data.

- Prophet: Prophet is a procedure for forecasting time series data based on an additive

Table 5.1: Accuracy performance of the forecasting models.

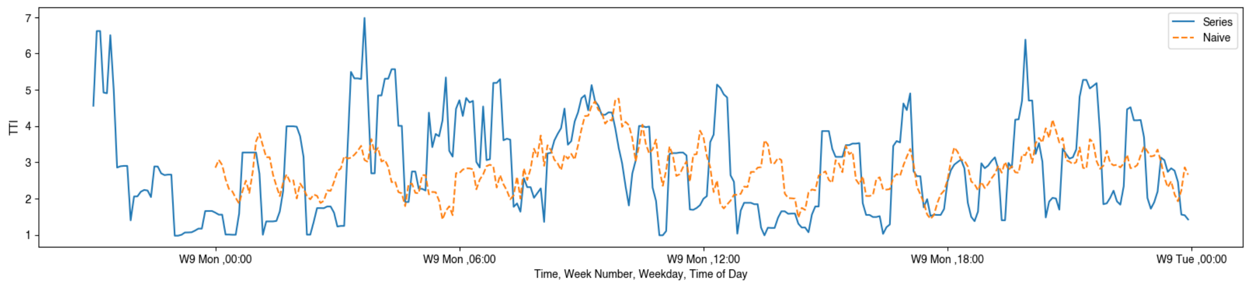
Naive	AutoArima	PolyTrend	ExpSm	Prophet	STS (ours)
1.24	1.22	1.25	2.25	1.19	**0.66**

model where non-linear trends are fit with seasonality.

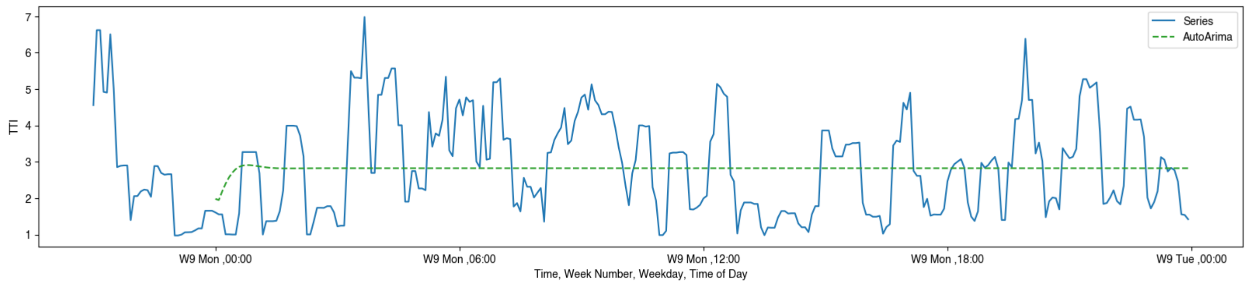
- STS (ours): Our proposed method based on structural time series, as described in Section 4.3.

5.4 Results

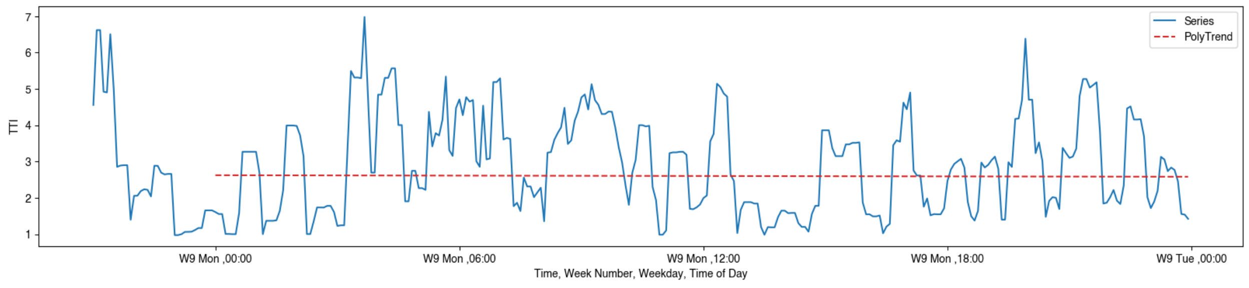
The results of the evaluation of the different forecasting models are shown in Table 5.1 and visualized for clarity in Fig. 5.6 and Fig. 5.7. Our structural time series model (STS) outperforms all the other sensible baseline time series forecasting models. Large fluctuations of the MOE time series data points cause many prediction models to predict a straight line that tries to fit the average MOE value in the training data. Other ones like the Naive approach have a problem detecting the peak areas because they solely see the same data on the previous windows which don't give you the spikes as they don't happen on the exact time in each day. On the other hand, the structural time series model is able to better capture the data dynamics and is able to provide more accurate results.



(a) Naive

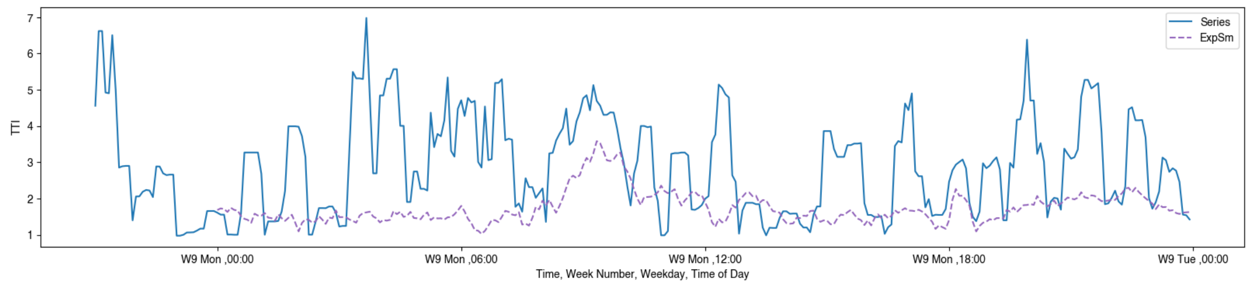


(b) AutoARIMA

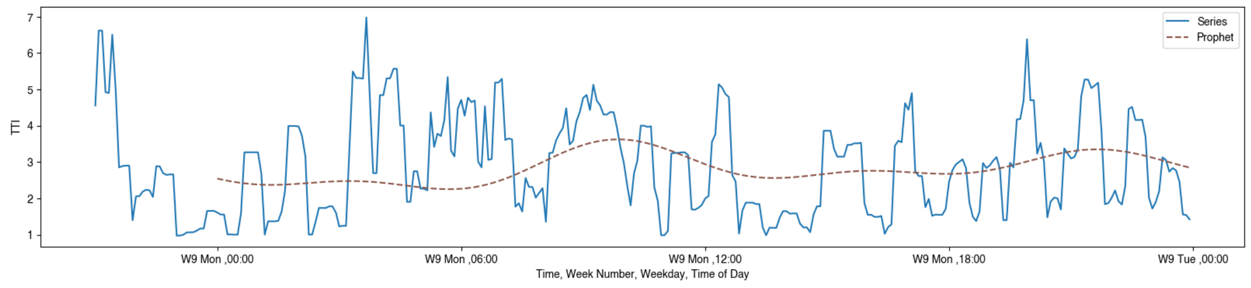


(c) PolyTrend

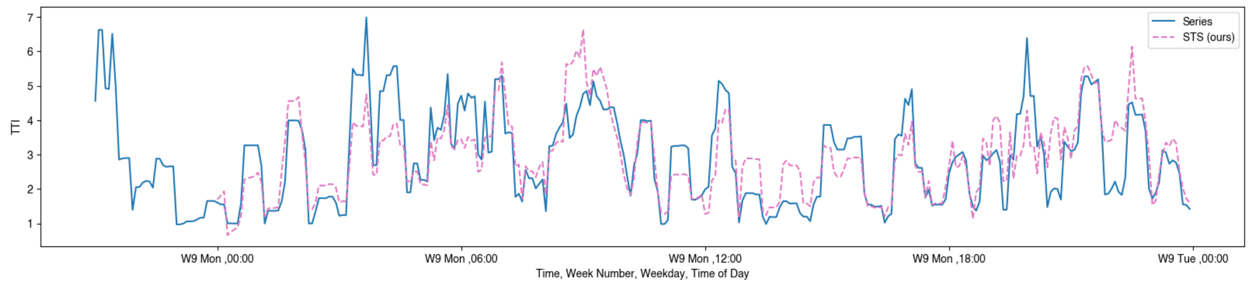
Figure 5.6: Accuracy performance of the forecasting models (visualization) (part 1).



(a) ExpSm



(b) Prophet



(c) STS (ours)

Figure 5.7: Accuracy performance of the forecasting models (visualization) (part 2).

Chapter 6

System Proof of Concept

In this section, we provide an overview of the system that allows to generate traffic flow data. The complete process is depicted in the diagram of Fig. 6.1 and includes two main components.

(A) **Synthetic traffic flow data generation**. This component is responsible for generating realistic synthetic traffic flow data for our setting. Traffic flow data consists of (*i*) the road network, and (*ii*) traffic data for a period of time.

(B) **Dashboard**. This component is responsible for visualization of the road network, as well as the MOE real-time analytics and forecasting (dashboard). It also allows for interactive exploration of the various metrics for different road network abstractions (i.e., road segment, road direction, road intersection).

We elaborate on these components in the next paragraphs.

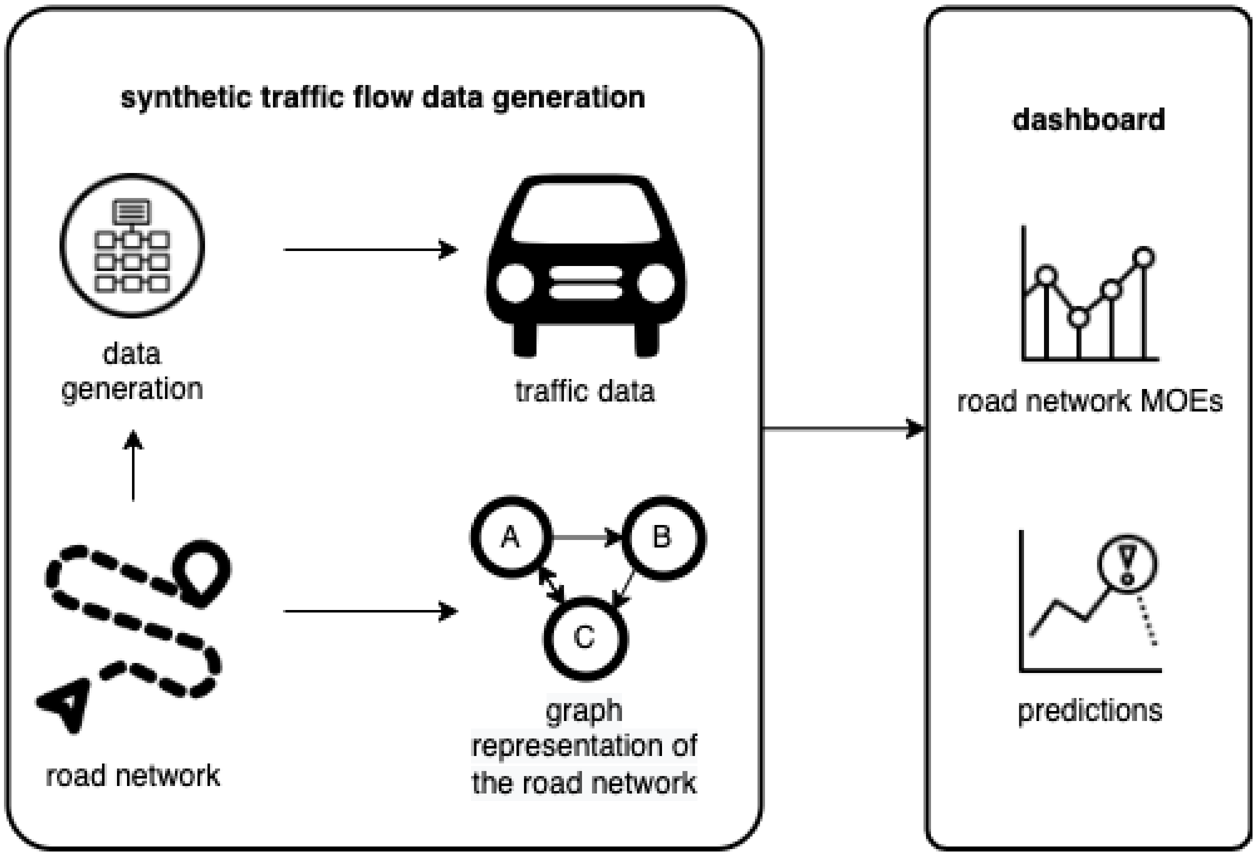


Figure 6.1: Illustration of the high-level traffic flow data generation system architecture.

6.1 Synthetic Traffic Flow Data Generation

The objective of the synthetic data generator is to provide realistic traffic flow data for a given urban area. Traffic flow data generation requires access to a road network and information about the traffic over it as a function of time. We rely on the Simulation of Urban MObility (SUMO[1]) package for simulating traffic flow data. SUMO is an open source, portable, microscopic and continuous multi-modal traffic simulation package designed to handle large networks. SUMO also supports tools for specific tasks such as route finding, visualization, network import and emission calculation.

6.1.1 Road Network Extraction

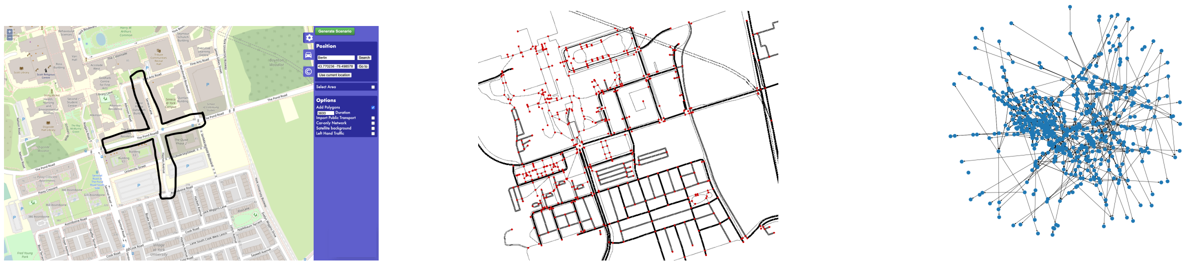
In order to generate traffic flow data of an urban area, we need to extract the road network map of that area. For that purpose, we use the OpenStreetMap (OSM) platform. OpenStreetMap[2] is a collaborative (crowdsourced) project that provides geodata underlying maps of the world. An example map extracted from OpenStreetMap can be seen in Fig. 6.2(a). Once a road network's geodata and meta information is extracted, we use NetEdit to obtain a simplified abstraction of it. NetEdit[3] is a visual network editor tool, included with SUMO, and it can be used to create customized road networks from scratch or to modify an existing one. An example road network after editing is shown in Fig. 6.2(b).

Graph representation of a road network. To work with the road network, an efficient

[1]`https://www.eclipse.org/sumo/`

[2]urlhttps://www.openstreetmap.org/

[3]`https://sumo.dlr.de/docs/Netedit/index.html`



(a) a road network (OpenStreetMap) (b) extracted network (NetEdit) (c) graph representation of the network

Figure 6.2: Road network extraction and graph representation.

data structure representation of it is required. Recall that in our system, we model a road network as a directed multigraph (multidigraph) $G := (\mathcal{V}, \mathcal{E}, s, t)$. In this model, nodes represent junctions, edges represent links between junctions, and each edge has its own identity, defined by functions s and t that assign the source and target node of the edge. In practice, each edge represents a road lane. An example graph of a road network can be seen in Fig. 6.2(c).

6.1.2 Traffic Data Generation

Traffic data provides information about the speed with which vehicles travel road segments over time. It is important in network analysis because traffic affects travel times. Obtaining real-world traffic data is challenging, mainly due to privacy concerns. We therefore had to resort to simulated traffic data for evaluating our system. SUMO has a comprehensive set of scenario-creation tools, where given a road network, one can configure a traffic scenario, run it, and visualize it in the SUMO-gui. By default, simulations in SUMO are randomized. However, random traffic exhibits no biases/patterns and it appears as if all destinations (edges or intersections) are equally important. This is problematic for our setting, since we would like to learn interesting patterns of traffic in the road network and forecast the operational performance of road intersections in the future. To address this issue, we had to resort to a tool within SUMO, called Activitygen[4] that can be parameterized to generate custom traffic data. The output of Activitygen is a file in FCD format.

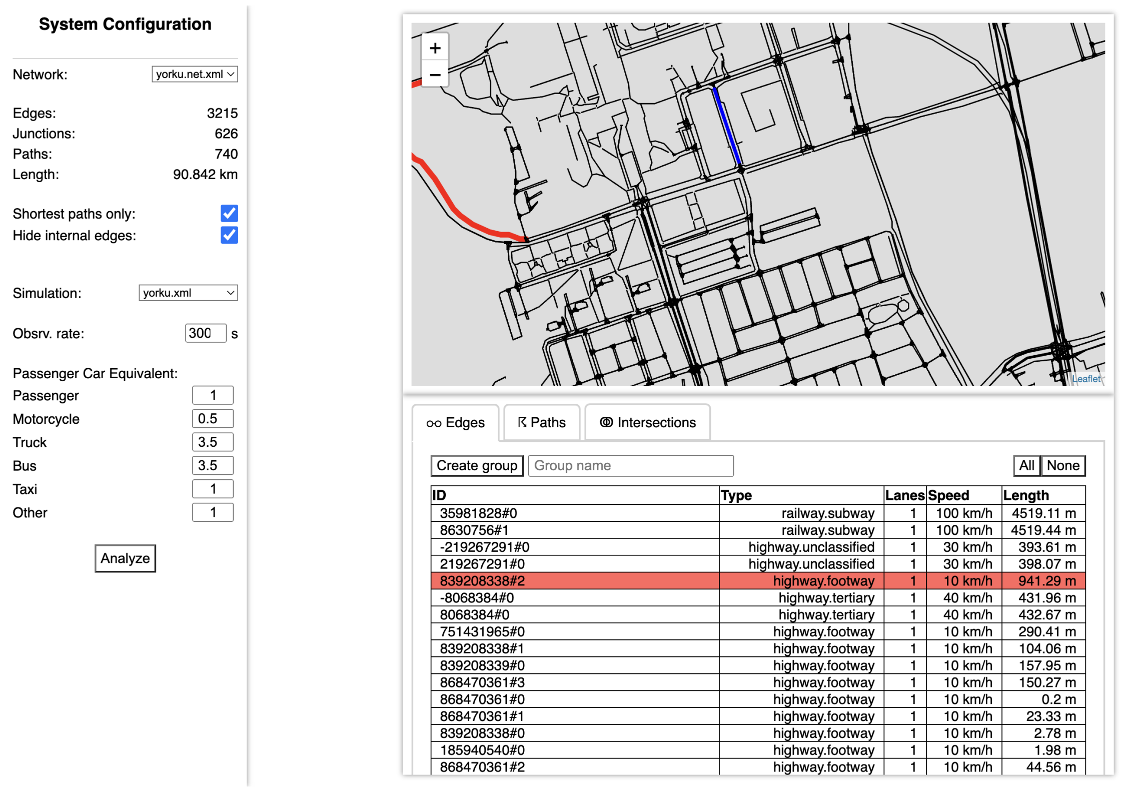
Activitygen. Activitygen generates demand from a description of the population in the network. It uses a simple activity-based traffic model, and supports going to work, school, park either by foot, a bike, a car, or a bus. Cars may have their start or stop locations outside the map. SUMO also handles all routing in an ad-hoc way, based on the current network status. Activitygen takes as input a configuration file that provides information about the population distribution inside the given network (the age ranges in the population, where they work, go to school, go for hobbies, and so on) and generate trips. Also the city gates for incoming and outgoing cars need to to be specified. Based on the estimated places of work and study, and the population distribution, a real world scenario is run in the network. For instance, people wake up, go to work, come back and do some random chores; they go to schools and come back home based on the hours specified, and so on. Once all trips are generated, they are given to SUMO for simulation; an output file in FCD format is created, which is explained in the next section.

[4]`https://sumo.dlr.de/docs/activitygen.html`

FCD output explanation. This file contains the traffic flow data that we need as input for the current work. It starts from time step 0 and goes on until the simulation has ended. The simulation time increases in increments/steps of one (1), representing one second in real world. Each second can be associated with several items containing people, cars, buses, and bikes. Since we focus on cars in this work, the only elements related to our work are the cars. Each car element has the following features: *id*, x, y, *angle*, *speed*, *position*, and *lane*. With information about the position of every car in the network at every second, we have realistic traffic flow data in the provided road network.

6.2 Dashboard

A key feature of our system is the design of a *dashboard* that provides visual analytics of the operational performance of a road network. In particular, the user interface provides at-a-glance views of the edge-level and network-level MOEs of the road network represented as time series. An illustration of the dashboard is shown in Fig. 6.3. In the left pane, one can select the network to be analyzed, and setup the parameters of the simulation. In the top-right pane, there is a visual of the entire network. In the bottom-right pane, there is a breakdown of the different system abstraction levels that have been automatically identified. These include all edges, paths representing road intersection directions, and custom networks. The system also allows for interactive exploration of the various system abstractions. Once a system (edge, path, or network) has been selected the results of the analysis are presented. An illustration can be seen in Fig. 6.4. The top-left pane has options for showing the MOEs.



System Configuration

Network:	yorku.net.xml
Edges:	3215
Junctions:	626
Paths:	740
Length:	90.842 km
Shortest paths only:	✓
Hide internal edges:	✓
Simulation:	yorku.xml
Obsrv. rate:	300 s

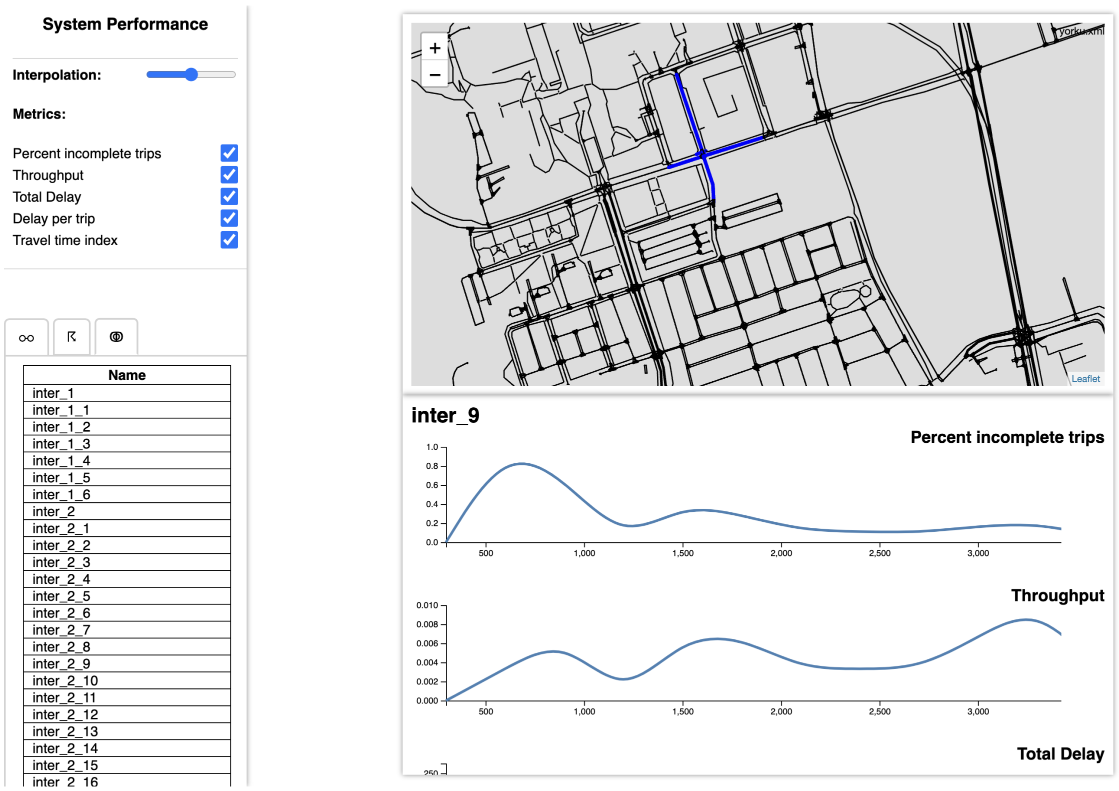
Passenger Car Equivalent:

Passenger	1
Motorcycle	0.5
Truck	3.5
Bus	3.5
Taxi	1
Other	1

ID	Type	Lanes	Speed	Length
35981828#0	railway.subway	1	100 km/h	4519.11 m
8630756#1	railway.subway	1	100 km/h	4519.44 m
-219267291#0	highway.unclassified	1	30 km/h	393.61 m
219267291#0	highway.unclassified	1	30 km/h	398.07 m
839208338#2	highway.footway	1	10 km/h	941.29 m
-8068384#0	highway.tertiary	1	40 km/h	431.96 m
8068384#0	highway.tertiary	1	40 km/h	432.67 m
751431965#0	highway.footway	1	10 km/h	290.41 m
839208338#1	highway.footway	1	10 km/h	104.06 m
839208339#0	highway.footway	1	10 km/h	157.95 m
868470361#3	highway.footway	1	10 km/h	150.27 m
868470361#0	highway.footway	1	10 km/h	0.2 m
868470361#1	highway.footway	1	10 km/h	23.33 m
839208338#0	highway.footway	1	10 km/h	2.78 m
185940540#0	highway.footway	1	10 km/h	1.98 m
868470361#2	highway.footway	1	10 km/h	44.56 m

Figure 6.3: Dashboard: interactive exploration of a network.

Identified systems are shown at the bottom-left pane, while at the bottom-right pane, the MOEs of a selected system are depicted.



System Performance

Interpolation:

Metrics:

- Percent incomplete trips ✓
- Throughput ✓
- Total Delay ✓
- Delay per trip ✓
- Travel time index ✓

Name
inter_1
inter_1_1
inter_1_2
inter_1_3
inter_1_4
inter_1_5
inter_1_6
inter_2
inter_2_1
inter_2_2
inter_2_3
inter_2_4
inter_2_5
inter_2_6
inter_2_7
inter_2_8
inter_2_9
inter_2_10
inter_2_11
inter_2_12
inter_2_13
inter_2_14
inter_2_15
inter_2_16

Figure 6.4: Dashboard: MOE visual analytics.

Chapter 7

Conclusion and Future Work

Being able to evaluate and forecast road intersection congestion is highly advantageous for traffic management tasks and to inform travel time prediction models. We have designed, developed and evaluated a system that is able to evaluate and forecast the operational performance of road intersections, which represent complex road network structures. Our system is based on continuous evaluation of industry standard measures of effectiveness (MOEs). In addition, we have evaluated several forecasting models that are informed by the computed MOEs. Our proposed structural time series model outperforms other sensible baselines methods for time series forecasting. Our system leverages advancements in streaming analytics, trajectory data mining and graph mining. The broader impact of our work is twofold: (*i*) it provides a simple, yet powerful data-driven approach to evaluating road intersection operational performance, and (*ii*) it contributes to an increased safety and efficiency of road intersections by enabling situational awareness of vehicle drivers. In addition, we provide an infrastructure to generate traffic data in any extracted or synthetic map network that is close to real-world and can be used to perform a broad range of experiments requiring traffic data. There were some limitations in the way of our research. For instance, congestion detection criteria are not very easy to define for the intersections, and we had to come up with something meaningful for the matter. In this regard, We found out that MOEs are being used in the industry, they can be aggregated, and they are good criteria for this problem. Additionally, there was difficulty in collecting the data because there was no real data satisfying the conditions we needed for this work like precision, being long-term, and in detail. so we created a platform to generate an acceptable data that is also close to real-world traffic.

There are various ideas on how to extend this work. At the moment, VtoI communication is still not easy so collecting data from the real world is a challenge but eventually, the technology is going to improve. Then, this work can be tested on real world data and may be used in the industry. Another idea is that instead of evaluating intersections to add them to other prediction works as a second element of knowledge, they can be reduced into roads to be merges with the mentioned works. The pathlets entering to each entrance of an intersection and exiting from each exit of the intersection can be considered as a separate road and be added to the features of the forecasts. Some experiments are needed to see if this idea will make those works easier and help with the accuracy or not. The other thing that comes into mind is that this work can be used to evaluate the overall operational performance of an area, if one is using network summerization or if one wants to compare operational performance of two areas in the city, they can use the aggregation methods of this work. In addition there are other traffic flow identifiers that can be explored as the measures of operational performance for example fundamental diagrams [40].

Bibliography

[1] Adnan, M., Pereira, F. C., Azevedo, C. M. L., Basak, K., Lovric, M., Raveau, S., Zhu, Y., Ferreira, J., Zegras, C., and Ben-Akiva, M. Simmobility: A multi-scale integrated agent-based simulation platform. In *95th Annual Meeting of the Transportation Research Board Forthcoming in Transportation Research Record* (2016).

[2] Asghari, M., Emrich, T., Demiryurek, U., and Shahabi, C. Probabilistic estimation of link travel times in dynamic road networks. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2015), pp. 1–10.

[3] Bergmeir, C., Hyndman, R. J., and Koo, B. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis 120* (2018), 70–83.

[4] Besenczi, R., Bátfai, N., Jeszenszky, P., Major, R., Monori, F., and Ispány, M. Large-scale simulation of traffic flow using markov model. *Plos one*

16 (2021), e0246062.

[5] Brinkhoff, T. Generating network-based moving objects. In *Proceedings. 12th International Conference on Scientific and Statistica Database Management* (2000), IEEE, pp. 253–255.

[6] Brodersen, K. H., Gallusser, F., Koehler, J., Remy, N., and Scott, S. L. Inferring causal impact using bayesian structural time-series models. *The Annals of Applied Statistics 9* (2015), 247–274.

[7] Cerqueira, V., Torgo, L., and Mozetič, I. Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning 109* (2020), 1997–2028.

[8] Chen, K., Chen, F., Lai, B., Jin, Z., Liu, Y., Li, K., Wei, L., Wang, P., Tang, Y., Huang, J., et al. Dynamic spatio-temporal graph-based cnns for traffic prediction. *arXiv preprint arXiv:1812.02019* (2018).

[9] Choi, H., and Varian, H. Predicting the present with google trends. *Economic record 88* (2012), 2–9.

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

[11] Dai, J., Yang, B., Guo, C., Jensen, C. S., and Hu, J. Path cost distribution estimation using trajectory data. *Proceedings of the VLDB Endowment 10* (2016), 85–96.

[12] Fang, S., Zhang, Q., Meng, G., Xiang, S., and Pan, C. Gstnet: Global spatial-temporal network for traffic flow prediction. In *IJCAI* (2019), pp. 2286–2293.

[13] Gal, A., Mandelbaum, A., Schnitzler, F., Senderovich, A., and Weidlich, M. Traveling time prediction in scheduled transportation with journey segments. *Information Systems 64* (2017), 266–280.

[14] Goldberger, J., Gordon, S., Greenspan, H., et al. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. In *ICCV* (2003), vol. 3, pp. 487–493.

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

[16] Hao, N., Feng, Y., Zhang, K., Tian, G., Zhang, L., and Jia, H. Evaluation of traffic congestion degree: An integrated approach. *International Journal of Distributed Sensor Networks 13* (2017).

[17] Harvey, A. C. Forecasting, structural time series models and the kalman filter.

[18] Hunter, T., Herring, R., Abbeel, P., and Bayen, A. Path and travel time inference from gps probe vehicle data. *NIPS Analyzing Networks and Learning with Graphs 12* (2009), 2.

[19] Idé, T., and Sugiyama, M. Trajectory regression on road networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2011), vol. 25.

[20] Jin, X., Zhang, Y., and Yao, D. Simultaneously prediction of network traffic flow based on pca-svr. In *International Symposium on Neural Networks* (2007), Springer, pp. 1022–1031.

[21] Jindal, I., Chen, X., Nokleby, M., Ye, J., et al. A unified neural network approach for estimating travel time and distance for a taxi trip. *arXiv preprint arXiv:1710.04350* (2017).

[22] Krajzewicz, D., Erdmann, J., Behrisch, M., and Bieker, L. Recent development and applications of sumo-simulation of urban mobility. *International journal on advances in systems and measurements 5* (2012).

[23] Li, S., Zhang, J., Zhong, G., and Ran, B. A simulation approach to detect arterial traffic congestion using cellular data. *Journal of Advanced Transportation 2022* (2022).

[24] Li, X., Cong, G., Sun, A., and Cheng, Y. Learning travel time distributions with deep generative model. In *The World Wide Web Conference* (2019), pp. 1017–1027.

[25] Li, Y., Fu, K., Wang, Z., Shahabi, C., Ye, J., and Liu, Y. Multi-task representation learning for travel time estimation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), pp. 1695–1704.

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

[27] Liao, B., Zhang, J., Wu, C., McIlwraith, D., Chen, T., Yang, S., Guo, Y., and Wu, F. Deep sequence learning with auxiliary information for traffic prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), pp. 537–546.

[28] Lv, Y., Duan, Y., Kang, W., Li, Z., and Wang, F.-Y. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems 16* (2014), 865–873.

[29] Lv, Z., Xu, J., Zheng, K., Yin, H., Zhao, P., and Zhou, X. Lc-rnn: A deep learning model for traffic speed prediction. In *IJCAI* (2018), pp. 3470–3476.

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

[31] Mehmood, S., and Papagelis, M. Learning semantic relationships of geographical areas based on trajectories. In *2020 21st IEEE International Conference on Mobile Data Management (MDM)* (2020), IEEE, pp. 109–118.

[32] Pan, J. J., Li, G., and Hu, J. Ridesharing: simulator, benchmark, and evaluation. *Proceedings of the VLDB Endowment 12* (2019), 1085–1098.

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

[34] Park, M., and Nassar, M. Variational bayesian inference for forecasting hierarchical time series. In *International conference on machine learning (ICML), workshop on divergence methods for probabilistic inference, Beijing* (2014), Citeseer.

[35] Pechlivanoglou, T., Li, J., Sun, J., Heidari, F., and Papagelis, M. Epidemic spreading in trajectory networks. *Big Data Research 27* (2022), 100275.

[36] Pechlivanoglou, T., and Papagelis, M. Fast and accurate mining of node importance in trajectory networks. In *2018 IEEE International Conference on Big Data (Big Data)* (2018), IEEE, pp. 781–790.

[37] Qureshi, K. N., and Abdullah, A. H. A survey on intelligent transportation systems. *Middle-East Journal of Scientific Research 15* (2013), 629–642.

[38] Rahmani, M., Jenelius, E., and Koutsopoulos, H. N. Route travel time estimation using low-frequency floating car data. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)* (2013), IEEE, pp. 2292–2297.

[39] Sawas, A., Abuolaim, A., Afifi, M., and Papagelis, M. Tensor methods for group pattern discovery of pedestrian trajectories. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)* (2018), IEEE, pp. 76–85.

[40] Siebel, F., and Mauser, W. On the fundamental diagram of traffic flow. *SIAM Journal on Applied Mathematics 66*, 4 (2006), 1150–1162.

[41] Song, H. Y., Baek, M. S., and Sung, M. Generating human mobility route based on generative adversarial network. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)* (2019), IEEE, pp. 91–99.

[42] Sun, X., Lin, K., Jiao, P., and Lu, H. The dynamical decision model of intersection congestion based on risk identification. *Sustainability (Switzerland) 12* (2020), 1–16.

[43] Tang, J., Chen, X., Hu, Z., Zong, F., Han, C., and Li, L. Traffic flow prediction based on combination of support vector machine and data denoising schemes. *Physica A: Statistical Mechanics and its Applications 534* (2019), 120642.

[44] Tang, J., Gao, F., Liu, F., and Chen, X. A denoising scheme-based traffic flow prediction model: Combination of ensemble empirical mode decomposition and fuzzy c-means neural network. *IEEE Access 8* (2020), 11546–11559.

[45] Tang, J., Liu, F., Zou, Y., Zhang, W., and Wang, Y. An improved fuzzy neural network for traffic speed prediction considering periodic characteristic. *IEEE Transactions on Intelligent Transportation Systems 18* (2017), 2340–2350.

[46] van Lon, R. R., and Holvoet, T. Rinsim: A simulator for collective adaptive systems in transportation and logistics. In *2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems* (2012), IEEE, pp. 231–232.

[47] Wang, D., Zhang, J., Cao, W., Li, J., and Zheng, Y. When will you arrive? estimating travel time based on deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2018), vol. 32.

[48] Wang, H., Tang, X., Kuo, Y.-H., Kifer, D., and Li, Z. A simple baseline for travel time estimation using large-scale trip data. *ACM Transactions on Intelligent Systems and Technology (TIST) 10* (2019), 1–22.

[49] Wang, W., and Li, X. Travel speed prediction with a hierarchical convolutional neural network and long short-term memory model framework. *arXiv preprint arXiv:1809.01887* (2018).

[50] Wang, X., Ma, Y., Wang, Y., Jin, W., Wang, X., Tang, J., Jia, C., and Yu, J. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020* (2020), pp. 1082–1092.

[51] Wang, Y., Zheng, Y., and Xue, Y. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), pp. 25–34.

[52] Wang, Z., Fu, K., and Ye, J. Learning to estimate the travel time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), pp. 858–866.

[53] Wei, L., Yu, Z., Jin, Z., Xie, L., Huang, J., Cai, D., He, X., and Hua, X.-S. Dual Graph for Traffic Forecasting. *IEEE Access PP* (2020), 1–1.

[54] Welch, G. F. Kalman filter. *Computer Vision: A Reference Guide* (2020), 1–3.

[55] Wu, H., Chen, Z., Sun, W., Zheng, B., and Wang, W. Modeling trajectories with recurrent neural networks. IJCAI.

[56] Yan, Y., Zhang, S., Tang, J., and Wang, X. Understanding characteristics in multivariate traffic flow time series from complex network structure. *Physica A: Statistical Mechanics and its Applications 477* (2017), 149–160.

[57] Yang, B., Guo, C., and Jensen, C. S. Travel cost inference from sparse, spatio temporally correlated time series using markov models. *Proceedings of the VLDB Endowment 6* (2013), 769–780.

[58] Yang, X. Understanding the variational lower bound. *In: variational lower bound, ELBO, hard attention* (2017), 1–4.

[59] Yang, X., Zou, Y., Tang, J., Liang, J., and Ijaz, M. Evaluation of short-term freeway speed prediction based on periodic analysis using statistical models and machine learning models. *Journal of Advanced Transportation 2020* (2020).

[60] Yu, J., Wang, L., and Gong, X. Study on the Status Evaluation of Urban Road Intersections Traffic Congestion Base on AHP-TOPSIS Modal. *Procedia - Social and Behavioral Sciences 96* (2013), 609–616.

[61] Yuan, H., Li, G., Bao, Z., and Feng, L. Effective travel time estimation: When historical trajectories over road networks matter. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (2020), pp. 2135–2149.

[62] Zhang, H., Wu, H., Sun, W., and Zheng, B. Deeptravel: a neural network based travel time estimation model with auxiliary supervision. *arXiv preprint arXiv:1802.02147* (2018).

[63] Zheng, J., and Ni, L. M. Time-dependent trajectory regression on road networks via multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2013), vol. 27.