Evaluating and Forecasting the Operational Performance of Road Intersections

Ali Nematichari
Lassonde School of Engineering
York University
Toronto, Canada
alinemat@eecs.yorku.ca

Tilemachos Pechlivanoglou
Lassonde School of Engineering
York University
Toronto, Canada
tipech@eecs.yorku.ca

Manos Papagelis
Lassonde School of Engineering
York University
Toronto, Canada
papaggel@eecs.yorku.ca

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.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; *Model development and analysis*; *Simulation types and techniques*; • **Applied computing** → **Transportation**; **Forecasting**; • **Mathematics of computing** → Probabilistic inference problems.

KEYWORDS

trajectory data mining, stream mining, road network, road intersection, machine learning, time series

ACM Reference Format:
Ali Nematichari, Tilemachos Pechlivanoglou, and Manos Papagelis. 2022. Evaluating and Forecasting the Operational Performance of Road Intersections. In *The 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '22), November 1–4, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3557915.3560965

SIGSPATIAL '22, November 1–4, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9529-8/22/11…$15.00
https://doi.org/10.1145/3557915.3560965

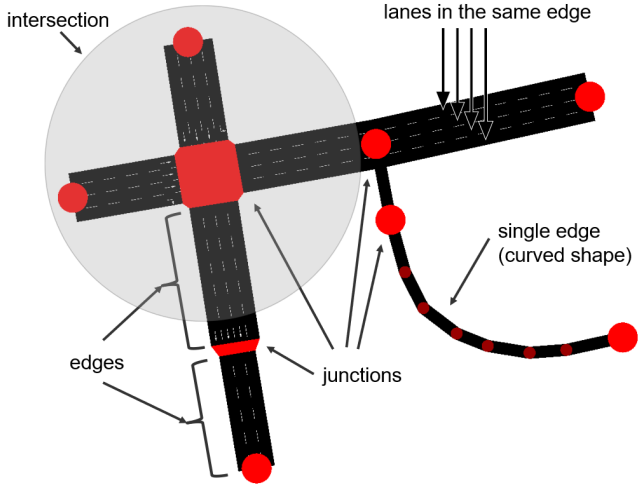


Figure 1: Illustration of *edges*, *junctions* and *lanes* forming a *road intersection*. Our system evaluates and forecasts the operational performance of road intersections by mining streams of trajectory data. Metrics are based on well-defined measures of effectiveness (MOEs) for traffic operations.

1 INTRODUCTION

Motivation. 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). It is characterized by slower speeds, longer trip times, and increased vehicle queues and is associated to significant social, economic and environmental costs. *Road intersections* represent one of the most complex configurations encountered when traversing road networks and a high percentage of accidents occur at these locations. It is therefore of vital importance to improve their operational performance, as that can significantly contribute towards the efficiency of the whole transport network. 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.

Limitations and our approach. Traditional approaches to improve the efficiency of intersections are based on analysis of static

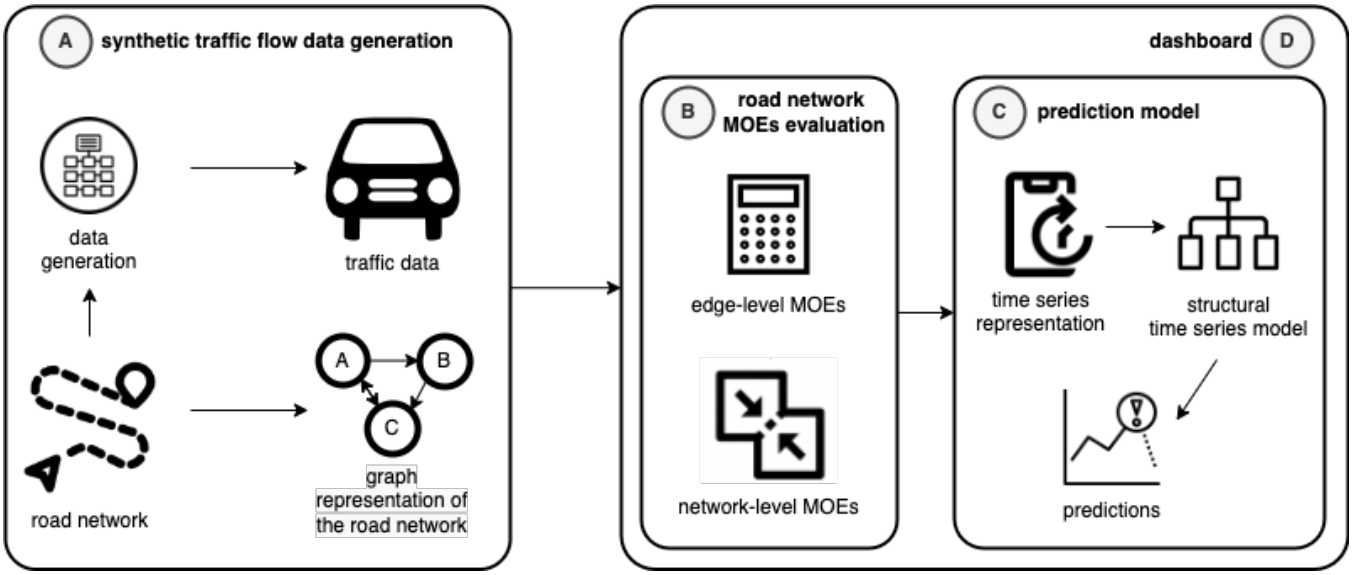


Figure 2: Illustration of the high-level system architecture.

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.

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). MOEs for each road intersection are reported in a dashboard (see Fig. 6 (a)). Critical components of the system have been ported and tested by our industrial partner for production. So, while there is no direct user base at the moment, it is designed and deployed as a more full fledged system to serve real needs of the industrial partner's clients/customers.

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 design methods for network-level aggregation of MOEs that are collected at edge-level.
- We design realistic traffic data generation tool for evaluating the time series models.
- We propose a method for forecasting the operational performance of a road intersection in the future that is based on a *structural time series* machine learning model.
- We conduct a comprehensive empirical study on both synthetic and realistic road networks to evaluate the proposed method against sensible baselines.

Organization. The remainder of the paper is organized as follows. Section 2 provides preliminaries and formally defines the problem of interest and engineering challenges. Section 3 describes the system architecture and its components. Section 4 discusses our experimental evaluation. The related work is discussed in Section 5. We conclude in Section 6.

2 PRELIMINARIES AND PROBLEM DEFINITION

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

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.).

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

Figure 3: Road network extraction and graph representation.

- $\mathcal{E}$ 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).

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).

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.

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.

3 SYSTEM ARCHITECTURE

In this section, we provide an overview of the system that allows to evaluate and forecast the operational performance of road intersections. The complete process is depicted in the diagram of Fig. 2 and includes four 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) **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.

(C) **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.

(D) **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.

3.1 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.

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

SUMO also supports tools for specific tasks such as route finding, visualization, network import and emission calculation.

3.1.1 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. 3(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. 3(b).

Graph representation of a road network. To work with the road network, an efficient 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. 3(c).

3.1.2 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 and any random prediction model would perform equally well as a more sophisticated time series model. 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 and it allows to generate more realistic traffic scenarios that include common traffic patterns (e.g., more congestion during rush hours, etc.). 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.

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

Figure 4: Different levels of system abstraction.

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.

3.2 Road Network MOEs Evaluation

In Section 2 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). For example, Fig. 4(a) depicts a *micro-level* system (a single road segment), Fig. 4(b) a *meso-level* system (a single intersection), and Fig. 4(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.

3.2.1 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 2 is represented as a time series, a sequence of data points occurring in successive order in $[0, T]$.

[2] urlhttps://www.openstreetmap.org/
[3] https://sumo.dlr.de/docs/Netedit/index.html
[4] https://sumo.dlr.de/docs/activitygen.html

3.2.2 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 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. 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.

3.3 Prediction Model

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*.

3.3.1 Time Series Smoothing. 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:

$$\bar{y_t} = \frac{y_t + y_{t-1} + \ldots + y_{t-w-1}}{w} \tag{1}$$

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. 7.

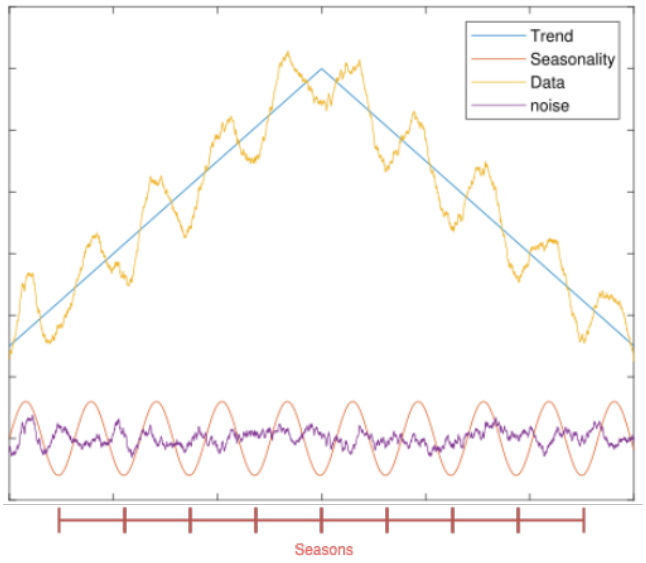


Figure 5: Structural time series decomposition.

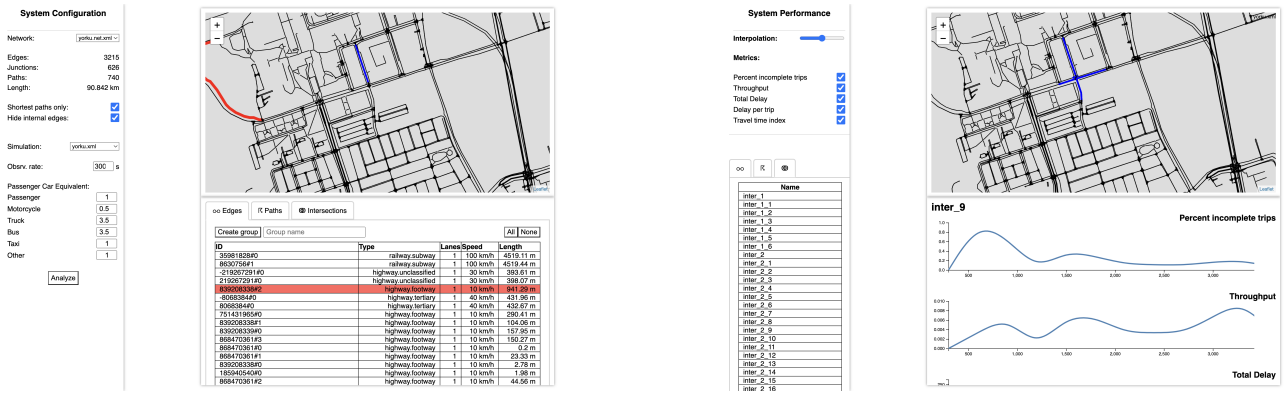
3.3.2 Structural Time Series. Although forecasts of future happenings are essentially uncertain, forecasting is an important aspect of long-term planning. Predicting time series methods can be used to infer the potential influence of a feature launch or any other intervention on customer engagement measurements [4], to extract the current value of challenging-to-observe quantities like the rate of unemployment from more available datasets [7], and in order to identify anomalies in time series data. In our system, we have designed and developed *structural time series* models [12] 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. (2).

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

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 provide a formal measurement of uncertainty.

Our model, considers the following decomposed components, also shown in Fig. 5 for a more clear intuition:

- `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.*



(a) Interactive exploration of a network. **(b) MOE visual analytics.**

Figure 6: Dashboard: Monitoring, analysis and interactive exploration of road intersections.

- `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 [36] and variational inference technique [22] 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.

More technical details can be found in Appendix A.

3.4 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 (a). 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 (b). The top-left pane has options for showing the MOEs. Identified systems are shown at the bottom-left pane, while at the bottom-right pane, the MOEs of a selected system are depicted.

4 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.

4.1 Data

We need data of all the traffic data in an intersection for around two month. Because there is no public data that gives that to us and the conditions don't allow us to gather such data, we decided to synthesize the data. To evaluate the forecasting models, we employ a two-month (9 weeks) synthetic traffic flow dataset using the data generator described in Section 3 and the map of Fig. 3(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. 7 demonstrates the time series of the `TTI` measure and the effect of the smoothing based on the moving average technique. Fig. 8 shows the prediction against the ground truth; the orange shading is showing the standard deviation of the uncertainty. As can be seen in Fig. 9, 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.

4.2 Evaluation Metric

While there are many metrics for evaluating the accuracy of a prediction, not all of them are adequate for time series data. 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 [5]. 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 [5] that is compatible with the time series is depicted in Fig. 10. 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)[2]. This is the easiest to calculate and interpret, and it is defined as follows.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

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

4.3 Methods

The following sensible methods have been evaluated:

- Naive: In a naive forecaster, 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 is an autoregressive integrated moving average. An Auto ARIMA model performs differencing tests to establish the order of differencing.
- PolyTrend: This model forecasts time series data with a 3rd degree polynomial trend.
- ExpSm: An exponential smoothing model where a prediction is a weighted average of past observations; it utilises an exponentially decreasing weight for past data.
- Prophet: Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with seasonality.
- STS (ours): Our proposed method based on structural time series, as described in Section 3.3.

Table 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**

4.4 Results

The results of the evaluation of the different forecasting models are shown in Table 1 and visualized for clarity in Fig. 11. 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 constant line that tries to fit the average MOE value in the training data. 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. We have also experimented with other intersections with smaller datasets (shorter periods) and the behavior was consistent. We reported only on the larger experiment with one intersection.

5 RELATED WORK

Our work is mostly related to *traffic generation* and the problem of *travel time estimation* and it also deals with *trajectory mining*. One of the most important topics in traffic management is travel time estimation. In this section, we try to review the literature about data generation and travel time prediction and we introduce some works around trajectory mining.

5.1 Traffic Generation

Traffic generation allows to simulate commuting scenarios and provides enough information to study traffic related problems. Techniques for traffic generation can be organized into two categories: *data simulation* and *data augmentation*.

5.1.1 Data Simulation. The majority of researchers are interested in the platform construction of transportation settings. They use the Bayes method to compute similar data distributions. For instance, a simulator is presented in [14] to assist in the preparation and execution of traffic scenario simulation, which involves a network, demand, and traffic generation. Similarly, Lon et al. [31] created a specialized framework for testing pickup-and-delivery algorithms. Adnan et al. [1] created a simulator to simulate millions of agents

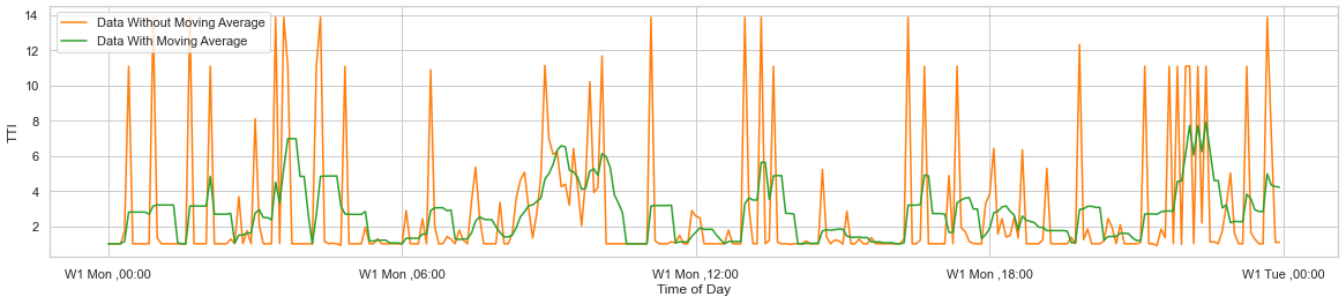


Figure 7: The effect of smoothing on time series data.

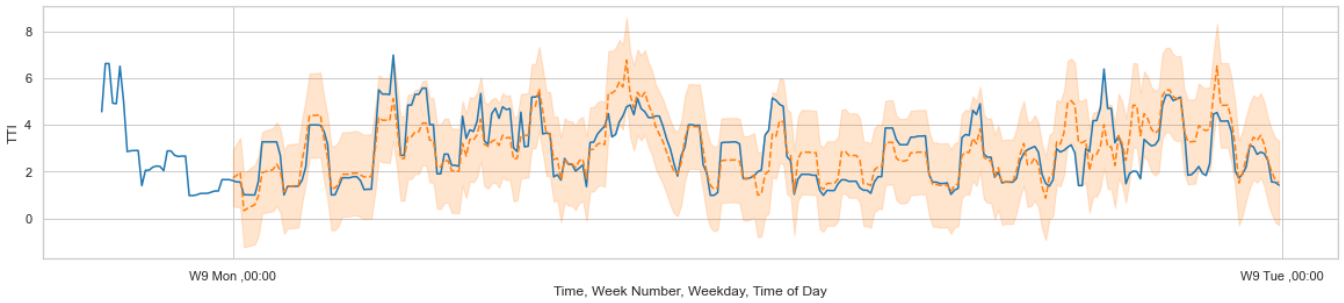


Figure 8: The prediction for the first day of the ninth week.

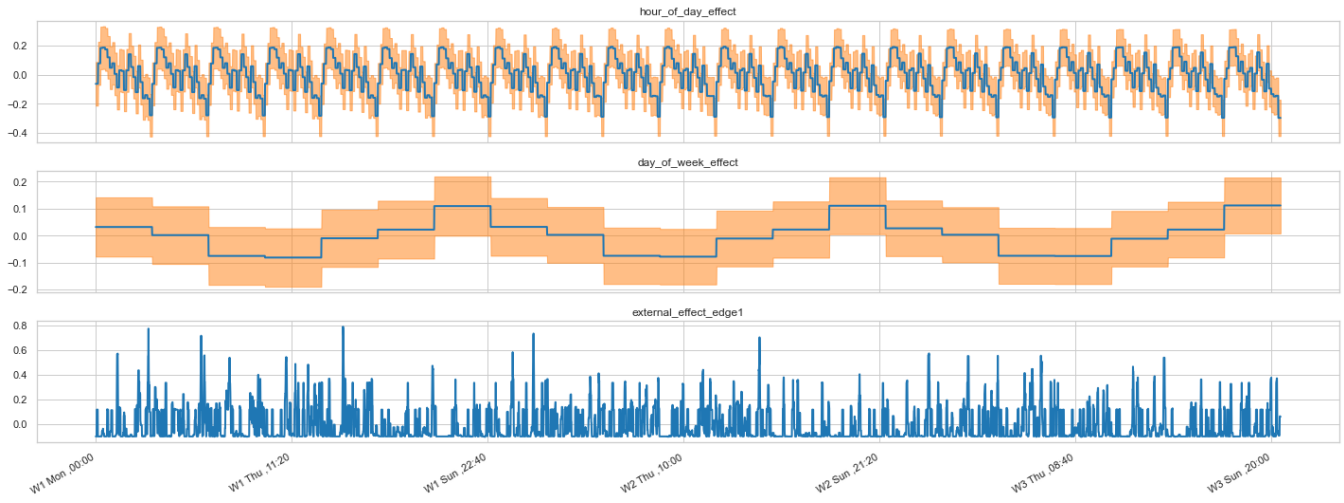


Figure 9: Effective components decomposition.

through a wide variety of mobility decisions. The simulator proposed in [20] 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 [15]. The notions of a Markov random walk, which outlines the movement of a single car, and Markov traffic, which characterizes the traffic on a road network are introduced in [3].

5.1.2 Data Augmentation. Another approach looks on how to produce individual data to solve estimation problems. Wang et al. [33], for example, created routes to predict the travel time from an origin to a destination. They use the kNN method to find the nearest historical path. Song et al. [28] used GAN (generative adversarial

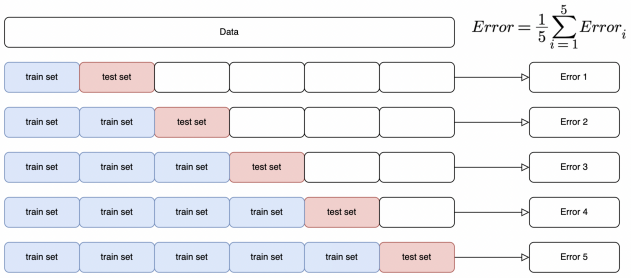


Figure 10: Time series cross-validation method.

networks) to create human mobility routes to address the same problem. On a different context, Wu et al. [37] generate fake spatiotemporal data that can substitute real data to prevent mobility privacy disclosure.

5.2 Travel Time Prediction

Work related to the travel time prediction problem can be organized into three categories: *origin-destination (OD) travel time*, *path travel time*, and *network flow*.

5.2.1 Origin-Destination (OD) Travel Time. The goal of *the OD travel time problem* is to predict the travel time for a given origin-destination input. The authors of [33] first used the kNN method to select historical trajectories with origin point and destination point that are close to the given origin-destination input. In [13], the multilayer perceptron (i.e., a multilayer fully connected neural network) is used to predict the origin-destination travel time. Li et al. [17], for example, utilized the residual neural network (ResNet) to encode given origin-destination input. Given the utility of historical trajectories, Yuan et al. [40] used LSTM and CNN methods to create a model that encodes historical trajectories, allowing an approximate travel time to be associated with a trajectory.

5.2.2 Path Travel Time. The *path travel time problem* is characterized as estimating travel time on road networks for a given route. Some machine learning models, such as decision tree [9] and hidden Markov model [38] have been proposed to address the problem of managing non-linearity. A few works concentrate on using deep learning methods (e.g., CNN and LSTM) to address the problem. For example, in [32, 41], the authors first consider a given path as a series of segments. Then, they use the CNN model to encode every segment to capture local spatio-temporal correlations; finally, used the RNN model to encode the entire path. Another work addresses the problem using the Wide-Deep model [35]. They split inputs into various components, which are encoded by various wide (e.g., affine transformation) and deep models. The authors of [16] used deep learning methods to produce probability parameters for associated generative models.

5.2.3 Network Flow. The *network flow problem* is concerned with optimizing traffic flow on a transport network. Tang et al. [29] use the SVR approach and improve it with denoising strategies. To enhance estimation accuracy, they merge a type of denoising algorithm and the fuzzy C-means neural network [30]. The authors of [6, 8, 11] use GCN models to forecast network flows by taking into account the nature of road networks. Fang [8] created a spatio-temporal module to encode historical traffic information, which includes a multi-resolution temporal module and a global correlated spatial module. In [34] authors address the sequential aspects of network flows using RNN techniques to encode past data. Li et al. [18] characterize network traffic as a diffusion process on a directed graph and propose DCRNN. Wang et al. [6] use a spatial GNN to encode historical information and a GRU model to encode the entire sequence. Meta-learning is used to capture the dynamic relationship between traffic flow data [21] taking into account the spatial features of road networks.

5.3 Trajectory Mining

The process of extracting useful information such as movement patterns, travel patterns, and traffic anomalies from trajectory databases is called *trajectory mining*. Sawas et al. [25, 26] 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. [24] devise a method that is able to simultaneously evaluate node importance metrics for all moving objects in a trajectory network. Sawas et al. [27] focuses on identifying pedestrian group patterns by analyzing moving pedestrian trajectory data to identify behaviours like group dispersion and group gathering. Mehmood et al. [19] 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. By introducing a data-driven model for the spread of the disease in a group that takes into account people's mobility patterns, Pechlivanoglou et al. [23] make use of GPS-enabled digital contact traces of individuals to inform a more thorough analysis and modelling of disease spreading.

6 CONCLUSIONS

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, based on industrial standard MOEs, and
(ii) it contributes to increased safety and efficiency of road intersections by enabling situational awareness of vehicle drivers.

Overall, the method we described is simple to understand and implement, accurate, fast, and general, so it can be easily adopted in a variety of strategies and applications for evaluating the performance of road intersections. We therefore expect our method to be beneficial in diverse settings and disciplines.

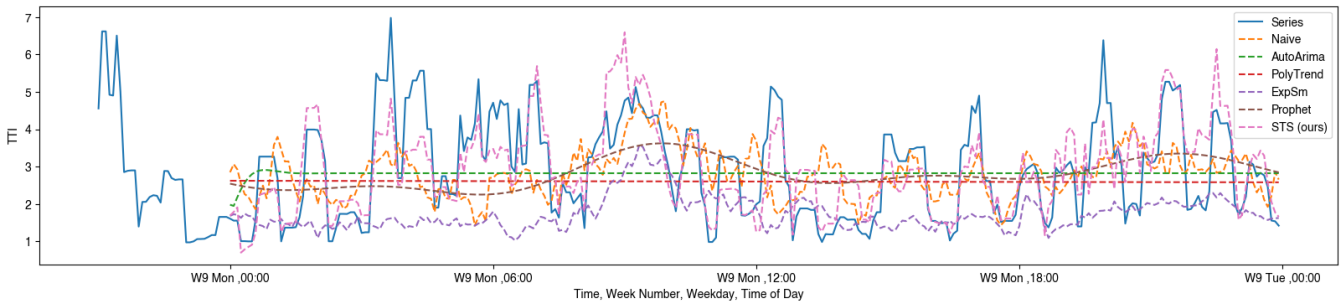


Figure 11: Accuracy performance of the forecasting models (visualization).

REFERENCES

[1] Muhammad Adnan, Francisco C Pereira, Carlos Miguel Lima Azevedo, Kakali Basak, Milan Lovric, et al. 2016. Simmobility: A multi-scale integrated agent-based simulation platform. In *95th Annual Meet. of the Transp. Res. Board.*

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

[3] Renátó Besenczi, Norbert Bátfai, Péter Jeszenszky, Roland Major, Fanny Monori, and Márton Ispány. 2021. Large-scale simulation of traffic flow using Markov model. *Plos one* 16 (2021), e0246062. Issue 2.

[4] Kay H Brodersen, Fabian Gallusser, Jim Koehler, Nicolas Remy, and Steven L Scott. 2015. Inferring causal impact using Bayesian structural time-series models. *The Annals of Applied Statistics* 9 (2015), 247–274. Issue 1.

[5] Vitor Cerqueira, Luis Torgo, and Igor Mozetič. 2020. Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning* 109 (2020), 1997–2028. Issue 11.

[6] Ken Chen, Fei Chen, Baisheng Lai, Zhongming Jin, Yong Liu, Kai Li, Long Wei, Pengfei Wang, Yandong Tang, et al. 2018. Dynamic spatio-temporal graph-based cnns for traffic prediction. *arXiv preprint arXiv:1812.02019* (2018).

[7] Hyunyoung Choi and Hal Varian. 2012. Predicting the present with Google Trends. *Economic record* 88 (2012), 2–9.

[8] Shen Fang, Qi Zhang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2019. GSTNet: Global Spatial-Temporal Network for Traffic Flow Prediction.. In *IJCAI*. 2286–2293.

[9] Avigdor Gal, Avishai Mandelbaum, François Schnitzler, Arik Senderovich, and Matthias Weidlich. 2017. Traveling time prediction in scheduled transportation with journey segments. *Information Systems* 64 (2017), 266–280.

[10] Jacob Goldberger, Shiri Gordon, Hayit Greenspan, et al. 2003. An Efficient Image Similarity Measure Based on Approximations of KL-Divergence Between Two Gaussian Mixtures.. In *ICCV*, Vol. 3. 487–493.

[11] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proc. of the AAAI conf. on Artif. Intellig.*, Vol. 33. 922–929. Issue 01.

[12] Andrew C Harvey. 1990. Forecasting, structural time series models and the Kalman filter. (1990).

[13] Ishan Jindal, Xuewen Chen, Matthew Nokleby, Jieping Ye, et al. 2017. A unified neural network approach for estimating travel time and distance for a taxi trip. *arXiv preprint arXiv:1710.04350* (2017).

[14] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. 2012. Recent development and applications of SUMO-Simulation of Urban MObility. *Inter. journal on advances in systems and measurements* 5 (2012). Issue 3&4.

[15] Shen Li, Jian Zhang, Gang Zhong, and Bin Ran. 2022. A Simulation Approach to Detect Arterial Traffic Congestion Using Cellular Data. *Journal of Advanced Transportation* 2022 (2022).

[16] Xiucheng Li, Gao Cong, Aixin Sun, and Yun Cheng. 2019. Learning travel time distributions with deep generative model. In *The Web Conference*. 1017–1027.

[17] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task representation learning for travel time estimation. In *Proc. of the 24th ACM SIGKDD Intern. Conf. on Knowledge Discovery & Data Mining*. 1695–1704.

[18] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

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

[20] James J Pan, Guoliang Li, and Juntao Hu. 2019. Ridesharing: simulator, benchmark, and evaluation. *Proc. of the VLDB Endowment* 12 (2019), 1085–1098. Issue 10.

[21] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proc. of the 25th ACM SIGKDD Int. Conf. on Knowl. Disc. & Data Min.* 1720–1730.

[22] Mijung Park and Marcel Nassar. 2014. Variational Bayesian inference for forecasting hierarchical time series. In *International conference on machine learning (ICML), workshop on divergence methods for probabilistic inference*. Citeseer.

[23] Tilemachos Pechlivanoglou, Jing Li, Jialin Sun, Farzaneh Heidari, and Manos Papagelis. 2022. Epidemic Spreading in Trajectory Networks. *Big Data Research* 27 (2022), 100275.

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

[25] Abdullah Sawas, Abdullah Abuolaim, Mahmoud Afifi, and Manos Papagelis. 2018. Tensor methods for group pattern discovery of pedestrian trajectories. In *19th IEEE Intern. Conf. on Mobile Data Management (MDM)*. IEEE, 76–85.

[26] Abdullah Sawas, Abdullah Abuolaim, Mahmoud Afifi, and Manos Papagelis. 2018. Trajectolizer: Interactive analysis and exploration of trajectory group dynamics. In *19th IEEE Intern. Conf. on Mobile Data Management (MDM)*. IEEE, 286–287.

[27] Abdullah Sawas, Abdullah Abuolaim, Mahmoud Afifi, and Manos Papagelis. 2019. A versatile computational framework for group pattern mining of pedestrian trajectories. *GeoInformatica* 23, 4 (2019), 501–531.

[28] Ha Yoon Song, Moo Sang Baek, and Minsuk Sung. 2019. Generating human mobility route based on generative adversarial network. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 91–99.

[29] Jinjun Tang, Xinqiang Chen, Zheng Hu, Fang Zong, Chunyang Han, and Leixiao Li. 2019. Traffic flow prediction based on combination of support vector machine and data denoising schemes. *Physica A: Stat. Mech. and its App.* 534 (2019), 120642.

[30] Jinjun Tang, Fan Gao, Fang Liu, and Xinqiang Chen. 2020. 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.

[31] Rinde RS van Lon and Tom Holvoet. 2012. RinSim: A simulator for collective adaptive systems in transportation and logistics. In *2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 231–232.

[32] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When will you arrive? estimating travel time based on deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32. Issue 1.

[33] Hongjian Wang, Xianfeng Tang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2019. A simple baseline for travel time estimation using large-scale trip data. *ACM Trans. on Intelligent Systems and Technology (TIST)* 10 (2019), 1–22. Issue 2.

[34] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*. 1082–1092.

[35] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to estimate the travel time. In *Proc. of the 24th ACM SIGKDD Int. Conf. on Knowl. Disc. & Data Min.* 858–866.

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

[37] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. 2017. Modeling trajectories with recurrent neural networks. IJCAI.

[38] Bin Yang, Chenjuan Guo, and Christian S Jensen. 2013. Travel cost inference from sparse, spatio temporally correlated time series using markov models. *Proceedings of the VLDB Endowment* 6 (2013), 769–780. Issue 9.

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

[40] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2020. Effective Travel Time Estimation: When Historical Trajectories over Road Networks Matter. In *Proc. of the 2020 ACM SIGMOD Int. Conf. on Management of Data*. 2135–2149.

[41] Hanyuan Zhang, Hao Wu, Weiwei Sun, and Baihua Zheng. 2018. Deeptravel: a neural network based travel time estimation model with auxiliary supervision. *arXiv preprint arXiv:1802.02147* (2018).

A APPENDIX: PREDICTION FORMULAE

More details about the steps of prediction are provided below.

A.1 Local Linear Trend

The local linear trend has the following definition with t as the time-step.

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

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.

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

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

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

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

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.

$$
\begin{aligned}
p(z \mid y) &= \frac{p(y \mid z)p(z)}{\int p(y \mid z)p(z)\mathrm{d}z} \quad \text{the posterior density} \\
z &= [\sigma_{level}, \sigma_{slope}, \sigma_{obs}] \quad \text{the vector of model parameters} \\
Y &= Y_{1:t} \quad \text{the vector of the observations}
\end{aligned}
$$

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.

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

A.2 Likelihood Calculation

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

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

We make the following assumption.

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

We can conclude the following because of the rule of Gaussian linear transformation.

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

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

$$
\begin{aligned}
&p\left(x_{t-1} \mid y_{1:t-1}, z\right) \\
&\quad\downarrow \\
&p\left(x_t \mid y_{1:t-1}, z\right) \\
&\quad\downarrow \\
&p\left(y_t \mid y_{1:t-1}, z\right) \\
&\quad\downarrow \\
&p\left(x_t, y_t \mid y_{1:t-1}, z\right) \\
&\quad\downarrow \\
&p\left(x_t \mid y_{1:t}, z\right)
\end{aligned}
$$

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.

A.3 Posterior Approximation

The variational distribution is used to estimate the posterior distribution p(z|y).

$$
\begin{aligned}
&q(z) \\
&= q\left(\sigma_{\text{level}}, \sigma_{\text{slope}}, \sigma_{obs}\right) \\
&= p\left(\sigma_{\text{level}}\right) \cdot p\left(\sigma_{\text{slope}}\right) \cdot p\left(\sigma_{obs}\right) \\
&= \mathcal{N}\left(\sigma_{\text{level}}; \mu_l, \sigma^2_l\right) \cdot \mathcal{N}\left(\sigma_{\text{slope}}; \mu_s, \sigma^2_s\right) \cdot \mathcal{N}\left(\sigma_{obs}; \mu_o, \sigma^2_o\right)
\end{aligned}
$$

A.4 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. 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).

$$
\begin{aligned}
\sigma_{level} &= e^{u_{level}} \\
\sigma_{slope} &= e^{u_{slope}} \\
\sigma_{obs} &= e^{u_{obs}}
\end{aligned}
$$

From now on σ_{level}, σ_{slope}, σ_{obs}, and p(y|z)p(z) are in the transformed domain.

A.5 PDF Distance Minimization

To quantify the level of overlapping between q(z) and p(z|y), we use the KL-divergence [10].

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

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

The theory of variational inference tells us that minimising the KL-divergence is equivalent to maximizing Evidence Lower Bound, denoted by ELBO(q(z)) [39].

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

A.6 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.

$$\begin{aligned}
&\nabla_{vp} ELBO(q(z)) \\
&= \nabla_{vp} \int q(z) \log \frac{p(y,z)}{q(z)} \mathrm{d}z \\
&\approx \nabla_{vp}\left[\frac{1}{n}\sum_{i=1}^{n} \log \frac{p\,(y, Z_i)}{q\,(Z_i)}\right], \quad Z_i \sim q(z) \\
&= \nabla_{vp}\left[\frac{1}{n}\sum_{i=1}^{n} \left(\log p\,(y, Z_i)) - \log q\,(Z_i)\right)\right] \\
&= \nabla_{vp}\left[\frac{1}{n}\sum_{i=1}^{n} \log p\,(y, Z_i)\right] - \nabla_{vp}\left[\frac{1}{n}\sum_{i=1}^{n} \log q\,(Z_i)\right]
\end{aligned}$$

A.7 Reparameterization Approach

We reparameterize each of the model parameters. You can see the reparameterization of σ_{level} as an example:

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

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

$$\theta = [\theta_{level}, \theta_{slope}, \theta_{obs}]$$

$$q(z) = q(r(\theta))$$

$$\begin{aligned}
&\mathrm{ELBO}(q(z)) \\
&= \int q(z) \log \frac{p(y,z)}{q(z)} \mathrm{d}z \\
&= \int q(r(\theta)) \log \frac{p(y, r(\theta))}{q(r(\theta))} \frac{\mathrm{d}r(\theta)}{\mathrm{d}\theta} \mathrm{d}\theta \\
&= \int \mathcal{N}(\theta; \mathbf{0}, \mathbf{1}) \log \frac{p(y, r(\theta))}{q(r(\theta))} \mathrm{d}\theta \\
&\approx \frac{1}{n}\sum_{i=1}^{n} \log \frac{p\,(y, r\,(\Theta_i))}{q\,(r\,(\Theta_i))}, \quad \Theta_i \sim \mathcal{N}(\mathbf{0}, \mathbf{1})
\end{aligned}$$

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

$$\begin{aligned}
\sigma_l &= e^{\phi_l} \\
\sigma_s &= e^{\phi_s} \\
\sigma_o &= e^{\phi_o}
\end{aligned}$$

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

Now the ELBO becomes a function of μ_l, μ_s, μ_o, ϕ_l, ϕ_s, and ϕ_o with 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.

A.8 Predicting the Future

We can run the Kalman filter now. 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 $y_{T+1}|y_{1:T}, z$.

$$\begin{aligned}
x_{T+1} &= Ax_T + w_{T+1} \quad \text{gives } p\,(x_{T+1} \mid y_{1:T}, z) \\
y_{T+1} &= Hx_{T+1} + v_{T+1} \quad \text{gives } p\,(y_{T+1} \mid y_{1:T}, z) \\
p\,(x_T \mid y_{1:T}, z) &\to p\,(x_{T+1} \mid y_{1:T}, z) \to \cdots \to p\,(x_{T+i} \mid y_{1:T}, z) \\
&\qquad\qquad \downarrow \qquad\qquad\quad \downarrow \\
&\qquad\qquad p\,(y_{T+1} \mid y_{1:T}, z) \to \cdots \to p\,(y_{T+i} \mid y_{1:T}, z)
\end{aligned}$$

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.