Citywide Traffic Volume Inference with Surveillance Camera Records

Yanwei Yu*, *Member, IEEE,* Xianfeng Tang*, Huaxiu Yao, Xiuwen Yi, and Zhenhui Li, *Member, IEEE*

Abstract—Real-time traffic monitoring becomes an essential part of an intelligent city. In recent years, the adoption of surveillance cameras is rapidly growing because they are helpful to manage and control the traffic. However, it is impossible to install cameras on every road in a city due to the high costs of deployment and maintenance. Given the information from limited surveillance cameras, can we infer the citywide traffic volume accurately? This is a challenging question because we have no historical data on the roads without cameras. It requires us to design a method that goes beyond the inference using nearby traffic data. Moreover, a nice property of surveillance camera data is that these AI-equipped cameras can recognize individual vehicles. So we can recover incomplete trajectories for vehicles using plate numbers in surveillance camera records. However, for road segments without cameras, we do not know whether those vehicles pass through them or not. How can such incomplete trajectories be effectively used to help citywide traffic inference? In this paper, we propose a framework named **CityVolInf** to infer citywide traffic volume based on surveillance camera records. Our framework combines a semi-supervised learning-based similarity module with a novel simulation module to address the above challenges. While the similarity module focuses on spatiotemporal correlations of traffic volume between road segments, the simulation module utilizes incomplete trajectories to model transitions of traffic volume between adjacent road segments. Our framework bridges the conventional data-driven approach and transportation domain knowledge from the simulator. We conduct extensive experiments on a real-world dataset, containing $405,370,631$ camera records collected from $1,704$ surveillance cameras over a period of 31 days in a provincial capital in China. The experimental results demonstrate the effectiveness of **CityVolInf** compared with existing methods.

Index Terms—Traffic volume inference, spatiotemporal data, urban computing.

1 Introduction

Real-time traffic monitoring can benefit a variety of urban applications such as traffic management, route planning, and public safety. In recent years, surveillance cameras are widely deployed to monitor traffic situations. For example, more than two thousand traffic surveillance cameras are installed in Beijing, China. These AI-equipped cameras can recognize individual vehicle information (e.g., license plate, speed, driving direction, etc.) and count the overall traffic volume. They are also used to automatically detect violations such as speeding. In addition, police can monitor the traffic conditions of the whole city and quickly respond to abnormal events (e.g., traffic accidents and congestion).

Despite the growing adoption of traffic surveillance cameras, their coverage in the city is still limited because of the cost of installment and maintenance. The traffic surveillance system requires high-resolution cameras, a high-speed interconnection network, and video processing techniques. For example, only about $1,500$ of more than $5,000$ major road segments in Jinan, China are monitored by surveillance cameras. Can we infer the city-wide traffic volume using camera records collected from a small portion of road segments?

In this paper, we aim to address the problem of traffic volume inference using the surveillance cameras records. This problem falls into the category of spatiotemporal missing data inference. The most naive approach is to estimate the missing values by linear interpolation [1], [2], [3], [4]. However, those simple approaches fail to optimize the problem globally and do not well utilize spatiotemporal characteristics obtained from historical data. Another frequently used method is to infer the missing values with collaborative filtering approach [5]. The method treats each location as a user, each timestamp as an item, and the traffic volume of a location at a time as the values for the user-item matrix. Traffic volume values for the missing locations can be estimated using the similar locations (measured by historical similarity). However, in our problem definition, there is no historical information for unmonitored road segments at all. Thus collaborative filtering cannot be applied to our problem because it relies on historical information to define similar road segments.

Recent spatiotemporal data inference study [6] proposes to construct an affinity graph, which defines nodes as locations, edge weights as the similarities between pairs of locations, and values on nodes as the traffic volume values. The objective is to minimize the aggregated differences between the values of two nodes, weighted by the edge weights, using graph-based semi-supervised approach. However, it is extremely difficult to incorporate some important road properties, such as the number of left-turn lanes and speed limits, when defining similarities in the semi-supervised

- *Y.W. Yu is with the Department of Computer Science and Technology, Ocean University of China, Qingdao 266100, China. E-mail: yuyanwei0530@gmail.com*
- *X.F. Tang, H.X. Yao and Z.H. Li are with the College of Information Sciences and Technology, Pennsylvania State University, University Park, PA 16802. E-mail: {xianfeng, huaxiuyao, jessieli}@ist.psu.edu*
- *X.W. Yi is with JD Urban Computing Business Unit, Beijing 100176, China. E-mail: xiuwenyi@foxmail.com*

Manuscript received January 29, 2019; revised June 25, 2019; accepted August 6, 2019.

*(*Authors contribute equally to this work. Corresponding author: Yanwei Yu.)*

learning framework.

Traffic surveillance cameras enable us to consider the transitions of traffic volume between road segments. This is possible because the cameras can recognize vehicle license plates. We can construct the trajectory for any specific vehicle by concatenating the locations where the vehicle is recognized in chronological order. Millions of such trajectories obtained from surveillance camera records can be used to estimate the transitions. However, such trajectories do not cover the road segments without cameras. We need to address the challenge of estimating the complete routes based on the incomplete camera based trajectories. Note that, some existing literature [4], [6], [7], [8] use densely sampled trajectory data (e.g., trajectories tracked by GPS embedded in vehicles) for traffic volume inference. However, those dense trajectories are sampled from specific kinds of vehicles (e.g., taxi trajectories used in [6]), leading to biases compared with the actual distribution of trajectories.

In this paper, we propose a novel framework named **CityVolInf** for citywide traffic volume inference with surveillance camera data. **CityVolInf** leverages the spatiotemporal similarities and transitions of traffic volume for inference. On the one hand, we introduce a similarity module that utilizes graph-based semi-supervised learning method to model spatiotemporal similarities of traffic volume between road segments. On the other hand, we propose a simulation module to leverage the incomplete trajectories recognized by cameras. The simulation module incorporates a traffic simulator (i.e., *SUMO* [9]), which can simulate multiple vehicles' movements jointly at a citywide scale. When estimating complete routes using *SUMO*, many properties of the road network, such as the effect of multiple left-turn lanes and speed limits, can be modeled precisely and effectively through the simulation. Conventional data-driven approaches are not capable of handling those heterogeneous properties and factors, given data from limited cameras. Further learned transitions from those routes benefit the traffic volume inference.

Combining the similarity module and the simulation module, our proposed **CityVolInf** bridges data-driven methods and transportation domain knowledge. To the best of our knowledge, this is the first attempt to utilize incomplete trajectories for citywide traffic volume inference, and the first attempt to combine simulators with existing data-driven approaches in this problem.

We use a large-scale real-world dataset collected from a provincial capital in China during the whole August 2016. The $405,370,631$ camera records contain more than 11 million unique vehicles, identified by plate numbers. We conduct comprehensive experiments to demonstrate the effectiveness of our proposed method.

To summarize, we make the following contributions:

- We propose **CityVolInf** for citywide traffic volume inference with surveillance camera records data.
- We design a similarity module by constructing an affinity graph to model spatiotemporal similarities of traffic volume between road segments.
- We propose a novel simulation module that leverages *SUMO* and incomplete camera based trajectories for modeling traffic volume transitions between adjacent road segments.
- We conduct extensive experiments on a large-scale real-world traffic surveillance dataset. Experimental results demonstrate that our proposed framework outperforms state-of-the-art methods.

The remainder of this article is organized as follows. We discuss related work which is related to our method in next section. We define the necessary concepts and formulate the focal problem of this paper in Section 3. We present our proposed **CityVolInf** framework in Section 4. Section 5 reports the experimental observations. Section 6 concludes the article.

2 Related Work

Many prior studies focus on data-driven approach for modeling city traffic. For example, to predict future traffic flow, autoregressive integrated moving average (ARIMA) and its variants have been widely applied [10], [11], [12], [13], [14]. Further studies start leveraging external context data such as venue information, weather conditions, and local events [3], [15], [16], [17], [18]. Recently, deep learning based methods [19], [20], [21], [22] reveal their strong capabilities in modeling complex non-linear spatiotemporal relations in traffic flow data. *However, all these models are region-based traffic prediction, which is completely different from our problem.* A line of studies are conducted to model urban human mobility based GPS records or geo-social data [23], [24], [25]. *Such methods aim to model human mobility with users' check-ins, which is also completely different from our problem.*

Varieties of research focus on filling-in missing value in spatiotemporal data. Basic methods learn a linear model to estimate the missing values. Some studies [1], [2], [3] use linear regression models to infer missing traffic speed or travel time based on taxi trajectories. Aslam et al. [4] learn a regression model with taxi GPS trajectories to estimate traffic volume. *However, regression methods require a great amount of labeled training data, which is unavailable in our problem setting.*

Another category of prior studies apply principal component analysis (PCA) (e.g., [26], [27], [28], [29]) or collaborative filtering (CF) (e.g., [5], [30], [31]) to fill in missing values in spatiotemporal data. PCA-based methods extract traffic patterns from observed data using various PCA techniques, such as Bayesian PCA [26], Probabilistic PCA [27], [28] and FPCA [29]. CF-based methods recover missing values by decomposing spatiotemporal data into the product of low-rank matrices. *However, both PCA-based and CF-based methods rely on historical data when filling in. They are unable to handle our problem since traffic volume of unmonitored road segments are totally missing.*

Semi-supervised learning (SSL) method [32] has been widely applied for unlabeled data inference, which can be used for inferring missing values. Label propagation [33], a classic semi-supervised method, infers unobserved labels by propagating existing labels on an affinity graph. Other SSL based methods [34], [35], [36] have been proposed to model the similarities of vertices in the affinity graph. *Although SSL methods can be applied to our problem, they only consider the similarities between vertices. Therefore, they fail to utilize rich traffic flow information for volume inference.*

Other existing work aim to infer traffic volume values of road segments using loop detector [6], [37], [38], surveillance cameras [39], [40], or float car trajectories [4], [7], [8]. [41] tries to model the characteristics of urban vehicular mobility using camera vehicular mobility images. But they only extract traffic densities from the images by simply counting the number of pixels in the images. [42] aims to understand urban mobility patterns by identifying the most popular routes through GPS trajectory clustering. Studies [37], [38], [39] tackle the volume estimation of a single road segment with loop detectors or surveillance cameras. Thus their methods cannot infer citywide traffic volume. Zhan et al. [7] propose a method to estimate citywide traffic volume using probe taxi trajectories. They estimate travel speeds for volume inference using full taxi trajectories. Recently, Meng et al. [6] propose ST-SSL that predicts city-wide traffic volume values using loop detector incorporating taxi trajectories. *However, both [7] and ST-SSL [6] require full observation of trajectories, which is not available from surveillance system. Therefore, those methods cannot be applied to tackle our problem.*

Several methods have been proposed to predict complete trajectories from partial observations. Zheng et al. [43] investigate how to reduce the uncertainty in low-sampling-rate trajectories. More specially, they aim to infer the possible routes for a given low-sampling-rate trajectory. Banerjee et al. [44] infer uncertain trajectories from network-constrained partial observations by summarizing all probable routes in a holistic manner. Yang et al. [45] investigate the problem of reconstructing hidden trajectories from a collection of separate spatial-temporal points where trajectory links are unknown. *However, these methods require historical trajectories as input, which cannot be applied to recovering routes in our problem.*

3 Data and Problem

In this section, we first introduce the data used in this paper, then formulate the traffic volume inference problem.

3.1 Data

We use a real-world dataset collected from a provincial capital in China (denoted City A). The data consists of two parts: road network and surveillance camera records.

- **Road Network.** The road network contains heterogeneous information such as road structures, road properties, traffic signals, etc. We obtain the road network from *OpenStreetMap* [46], which is public-available. The road network information is used for learning similarity and simulating citywide vehicle movements in our framework.
- **Surveillance Camera Records.** Surveillance camera records contain all vehicles detected by all surveillance cameras in the traffic monitoring system. When any vehicle passes by a camera, the camera will send a record containing recognized information about the vehicle to the server. The format of each record follows $\langle veh_{id}, cam_{id}, ts\rangle$, which represents a vehicle with plate id veh_{id} passing the road segment monitored by camera cam_{id} at timestamp ts. Note that due to limitations in computer vision techniques, a small portion of vehicle plates are not recognized. A virtual plate number (i.e., *unknown*) is assigned to those vehicles. In our collected dataset, about 88% of records contain a real vehicle plant number.

3.2 Problem Definition

We first define the key data structures and notations used in the paper. We define road segment as follows:

Definition 1 (Road Segment). We split roads into short road segments. We use intersections as natural separations of roads. Each road segment connects two adjacent intersections. Note that road segments are directed. Let $R = \{r_1, r_2, \ldots, r_M\}$ denote all road segments in a city, where M is the number of road segments.

Each camera is deployed at a corresponding road segment, namely, the vehicles captured by camera cam_{id} pass through its corresponding road segment. Since the camera can recognize vehicles and their license plate number, traffic volume of road segments can be identified by aligning cameras with road segments.

We define time interval as follows:

Definition 2 (Time Interval). We split time as N non-overlapping equal-length time intervals and use $T = \{t_1, t_2, \ldots, t_N\}$ to denote all time intervals.

Definition 3 (Traffic Volume). The traffic volume for road segment r_i during time interval t_j is defined as the total number of vehicles passing through r_i during the time interval t_j.

Definition 4 (Camera Based Trajectory). The camera-based trajectory of a vehicle with plate number id is a sequence of tuples in chronological order, denoted by $\{\langle id, r, ts\rangle\}$, where each tuple $\langle id, r, ts\rangle$ represents that the vehicle id appears on road segment r at timestamp ts.

Obviously, camera-based trajectories of vehicles are incomplete because only partial road segments are deployed with surveillance cameras.

We now state our problem as below:

Problem 1 (Citywide Traffic Volume Inference). Given the road network, camera-based trajectories, and observed traffic volume, our goal is to infer citywide traffic volume of any road segment at any time interval.

4 The Framework of CityVolInf

4.1 Overview

Figure 1 illustrates the overall framework of **CityVolInf**, which takes the road network and surveillance camera records as input, and outputs the inference result of citywide traffic volume values. In particular, our framework consists of three key components: (1) data preprocessing module that extracts road network features, traffic volume values and camera-based trajectories from the raw input data; (2) similarity module, which utilizes graph-based semi-supervised learning method to model spatiotemporal similarities of traffic volume between road segments, and (3) simulation module, which generates transitions between adjacent road segments using *SUMO* and observed incomplete trajectories to enhance inference results.

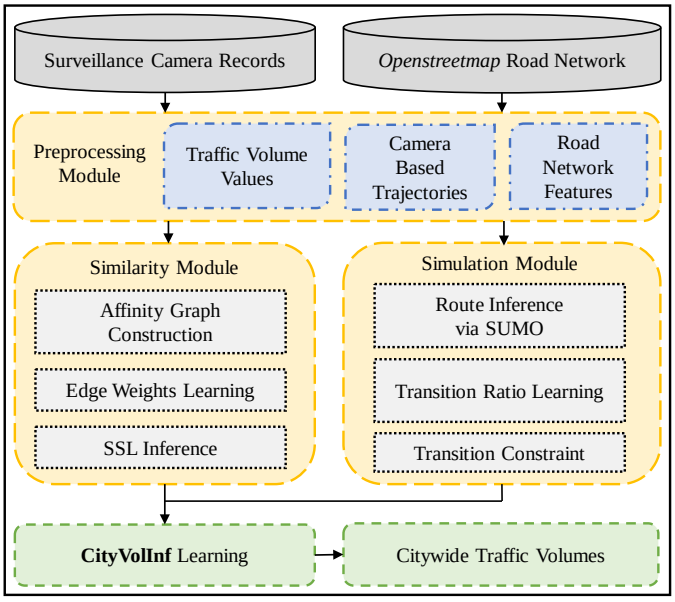


Fig. 1: Overall framework of **CityVolInf**.

In what follows, the data preprocessing module is introduced in Section 4.2. Then the similarity module is presented in Section 4.3. Next in Section 4.4 we discuss the simulation module. Finally, we elaborate the optimization in Section 4.5.

4.2 Data Preprocessing Module

The first step of **CityVolInf** is to preprocess the raw input data. That is, we extract road network features, traffic volume values and camera-based trajectories using road network information from *Openstreetmap* and collected surveillance camera records.

- **Road Network Features.** We extract features from the road network. More specifically, we identify several types of features from the heterogeneous *OpenStreetMap* data for each road segment, including starting/ending locations, road segment length, road width, road type, and speed limit. Let $\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_M\}$ denote the feature vector set, where $\mathbf{f}_i$ is the feature vector of road segment r_i.
- **Traffic Volume Values.** According to Definition 3 in Section 3.2, the traffic volume values for a given road segment are counted using captured vehicles at the aligned camera during each time interval. Because vehicle detection technique is very accurate, counted traffic volume can be used as ground truths. The traffic volume value of road segment r_i during time interval t_k is represented as x_i^k, which is a non-negative integer. Moreover, let $\mathbf{X}^k \in \mathbb{R}^{M\times 1}$ denote the traffic volume vector for all road segments at time interval t_k, and $\mathbf{X} \in \mathbb{R}^{M\times N}$ denote traffic volume values for all segments in all time intervals. Namely, $\mathbf{X} = [\mathbf{X}^1, \mathbf{X}^2, \cdots, \mathbf{X}^N]$.
- **Camera Based Trajectories.** We extract large amounts of camera-based trajectories from the records with valid real plate numbers. More specifically, the trajectory Ω_{id} for vehicle id can be identified by listing all surveillance camera records containing the same id in chronological order. We use $\mathbf{\Omega} = \{\Omega_{id}\}$ to denote all such camera based trajectories.

4.3 Similarity Module

Spatial and temporal correlations on road network play important roles in the traffic inference. In other words, the traffic volume values of different road segments are correlated with each other in spatial and temporal perspectives. For example, the traffic volume value of one road segment tends to be larger if all neighboring road segments have higher volume at the same time. The traffic volume of one road segment is likely to be lower during a time interval if we observed low traffic volume during the corresponding time interval in the past several days. Inspired by above observations, we build an affinity graph to describe the correlations between road segments, where each road segment during a time interval is a node in the graph. The edges in the affinity graph represent correlations and similarities between road segments at different time intervals. After defining the affinity graph, we can infer city-wide traffic volume values by using graph-based semi-supervised learning (SSL) approach [33]. We will first introduce how to build the affinity graph, then infer city-wide traffic volume values using graph-based semi-supervised learning approach.

Affinity Graph: an affinity graph can be represented by a multi-layer weighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathcal{W} \rangle$ (as shown in Figure 2). $\mathcal{V} = \{\mathcal{V}^1, \cdots, \mathcal{V}^N\}$ is the set of all road segments at all time intervals, where $\mathcal{V}^k = \{v_1^k, v_2^k, \cdots, v_M^k\}$ is a layer that contains all nodes at time interval t_k, and v_i^k represent road segment r_i at time interval t_k. $\mathcal{E}$ is the set of all edges, which can be divided into two types. The first type is *spatial edges* between nodes at the same time interval. This type of edges reflect spatial correlations between road segments. The second type is *temporal edges* between two nodes that represent the same road segment at different time intervals (i.e., edges between different v_i^k and v_i^{k-1}). Those edges represent temporal similarities of the same road segment at different time intervals. $\mathcal{W}$ is the set of weights on the edges $\mathcal{E}$. For simplicity, let $\mathcal{V} = \mathbf{L} \bigcup \mathbf{U}$, where $\mathbf{L}$ denotes the set of road segments with observed traffic volume (labeled nodes), and $\mathbf{U}$ is the set of road segments without traffic volume information (unlabeled nodes). Traffic volume for the nodes in $\mathbf{U}$ need to be further inferred.

To construct edges in the affinity graph, we consider the following five types of correlations from spatial and temporal perspectives. Figure 2 shows an example of the affinity graph. We build multiple layers to model spatiotemporal similarities.

- **Edges between adjacent road segments**. Previous study [6] tells that traffic volume patterns of adjacent road segments are more similar to each other because vehicles traverse between them frequently. For example, a traffic peak on a certain road segment would affect its neighbors via vehicle movements. Therefore, traffic volume values of adjacent road segments in the same time interval are likely to be similar. Inspired by this, we add an edge for each pair of nodes which represent adjacent road segments.
- **Edges between reachable road segments**. In addition to adjacency, reachable road segments are likely to have

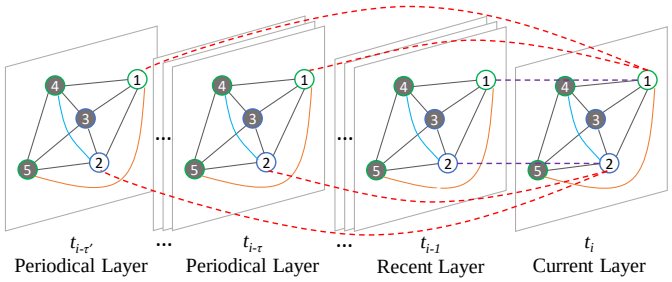


Fig. 2: An example of the affinity graph. There are five road segments, including three monitored segments (i.e., 3, 4 and 5 in black) and two unmonitored segments (i.e., 1 and 2 in white). The boundary color of node indicates road type. For example, nodes 1, 4 and 5 are of the same type. Solid lines within each layers are spatial edges, where gray, cyan and yellow represent edges between adjacent, reachable and same-type road segments respectively. Dash lines connecting nodes from different layers denotes temporal edges, where purple ones are between recent layers, and red ones are between periodical layers.

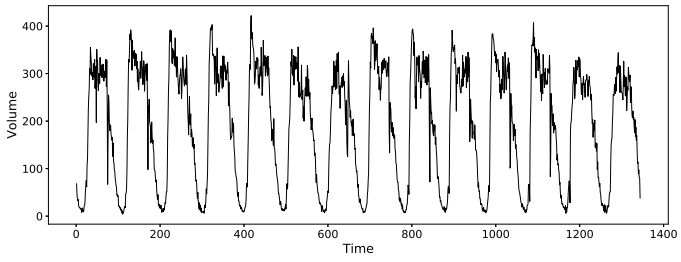


Fig. 3: Traffic volume of a road segment in City A.

similar traffic patterns. For example, two nearby road segments may have similar traffic volume patterns even if they are not adjacent, especially when vehicles can traverse from one to another easily.

To model the similarities between reachable road segments, we first find reachable road segments using Euclidean distance. Then we select top λ nearest neighbors for each unmonitored road segment, and add edges between the road segment and selected neighbors.

- **Edges between same type road segments**. According to domain experts, road segments that belong to the same type (e.g., highway, trunk, primary.) tend to have similar traffic patterns. Because we aim to leverage few road segments with traffic volume data for inference, we connect every unmonitored road segment to all monitored road segments of the same type.
- **Edges between recent time intervals**. Traffic volume patterns have strong temporal dependencies. Generally, the volume on a certain road segment would not change dramatically during a relatively short time period. We connect every node to the corresponding node in the previous time intervals.
- **Edges between periodical time intervals**. Daily traffic patterns have strong temporal periodicity. The traffic volume at a specific time interval should be close to its relevant time intervals in previous time (e.g., daily, weekly and monthly periods). For example, Figure 3 shows the traffic volume of a road segment in two weeks. We observe the strong daily and weekly periodicities from the traffic volume pattern. To incorporate such periodicities, we connect each node to previous corresponding ones with the most periodic similarities (e.g., nodes one day before, one week before, etc.). In this paper, we consider daily and weekly periodicity.

The next step of building affinity graph is to learn weights on edges, which represent similarities between nodes. If two nodes are more similar, the weight on the edge between them should be larger. We use features on nodes to learn weights, following three steps:

1) *Feature Scaling*: To avoid the similarity being governed by any particular features, we first standardize $\mathbf{F}$ by applying Z-normalization on the values of all feature vectors so that the values have zero-mean and one-variance.
2) *Spatial Similarity*: Intuitively, if the feature vectors of two connected nodes within the same time interval are more close, their similarity weight should be higher. To generate weights for spatial edges within each layer, we define a linear function over the difference of feature vectors from two nodes connected by spatial edges. Suppose v_i and v_j are two nodes from the same time interval, and $\mathbf{f}_i$ and $\mathbf{f}_j$ are their normalized feature vectors respectively, we define the weight between v_i and v_j as:
$$w_{i,j} = w(\mathbf{f}_i, \mathbf{f}_j) = \exp\big(-\mathbf{a}\cdot(\mathbf{f}_i - \mathbf{f}_j) + \mathrm{c}\big),$$
where $\mathbf{a}$ is a row vector of parameters, c is the bias parameter, and "$\cdot$" denotes the inner product of vectors. To estimate $\mathbf{a}$ and c, we minimize the following loss function that defines on the sub-network of nodes with observed traffic volume:
$$\begin{aligned}\mathbf{a}, \mathrm{c} &= \arg\min_{\mathbf{a},\mathrm{c}} \mathcal{L}_0 \\ &= \arg\min_{\mathbf{a},\mathrm{c}} \sum_{k=1}^{N} \sum_{\substack{i,j=1 \\ v_i^k, v_j^k \in L}}^{M} \exp\big(-\mathbf{a}\cdot(\mathbf{f}_i - \mathbf{f}_j) + \mathrm{c}\big)(x_i^k - x_j^k)^2.\end{aligned}$$
3) *Temporal Similarity*: Temporal similarities could be modeled by weights on temporal edges between different layers of the affinity graph. To explicitly capture temporal dependencies, we manually set all weights to 1 on edges between different layers, which reflects the temporal characteristic of traffic volume values.

Given the affinity graph with known weights, we can use a graph-based semi-supervised learning (SSL) model to infer citywide missing traffic volume values. The basic idea of this SSL model is that if two nodes are connected by an edge with larger weight, their traffic volume values tend to be more similar. To model such similarities from the view of edge weights, we propose the following loss function:
$$\begin{aligned}\mathcal{L}_1 &= \mathcal{L}_S + \alpha\mathcal{L}_T \\ &= \frac{1}{2}\sum_{k=1}^{N}\sum_{i,j=1}^{M} w_{i,j}(x_i^k - x_j^k)^2 + \frac{\alpha}{2}\sum_{k=1}^{N-1}\sum_{i=1}^{M}(x_i^{k+1} - x_i^k)^2,\end{aligned} \tag{1}$$

where $\mathcal{L}_S$ and $\mathcal{L}_T$ are loss functions for spatial and temporal similarities respectively, and α is the coefficient parameter.

4.4 Simulation Module

Although the similarity module successfully models spatial and temporal similarities through graph-based semi-supervised learning approach, it still fails to capture dynamic characteristics of traffic volume. That is, transitions of traffic volume between adjacent road segments are ignored when using conventional semi-supervised learning method. Despite similarities, traffic volume also affect each other explicitly via a "weighted-sum" process (i.e., transition). Because a lot of vehicles are traversing continuously on roads, most vehicles on one road segment will finally arrive at adjacent road segments. For example, if 100 vehicles are moving alongside one road segment towards a crossroad, we aim to answer the following question: how many vehicles will go straight ahead, how many of them will turn left, and how many of them will turn right? Suppose we can answer the question for every road segment, we can approximate traffic volume values by propagating those at road segments with data.

Inspired by above discussion, we model the transition of traffic volume in our framework. More specifically, we first *learn transition ratio matrices from camera-based trajectories*, then *propose a simulation module utilizing learned transition ratios*.

4.4.1 Transition Probability

To model the transition between adjacent road segments, an intuitive approach is to generate vehicle numbers directly from camera-based trajectories. If two adjacent road segments both are monitored by cameras, the transition between them can be identified from counting. However, it is unpractical to model traffic volume transitions from camera-based trajectories directly. Because camera-based trajectories fail to cover all vehicles in the city. Only 88% of records contain valid plate numbers, some trajectories are missing even if the vehicle (counted towards traffic volume) is detected. Therefore, directly counting transition will produce inaccurate results.

To tackle the aforementioned limitations, **CityVolInf** learns average transition probability as an alternative approach. Transition probability can be considered as the conversion rate of traffic volume between adjacent road segments. We use a real value $p^k_{i,j} \in [0,1]$ to represent the average transition probability. That is, $p^k_{i,j}$ denotes the conditional probability of vehicles traversing from r_j to r_i in time interval t_k:

$$p^k_{i,j} = P(r_i \backslash r_j \mid r_j, t_k) = \frac{\#\text{ of } r_i \backslash r_j \text{ in } t_k}{x^k_j}, \tag{2}$$

where $P(\cdot|\cdot)$ denotes conditional probability, $r_i \backslash r_j$ means "traversing from r_j to r_i".

Transition probability can be learned from partial missing trajectories. Since cameras are almost uniformly deployed in the city, extracted camera based trajectories can be considered as unbiased samples of all vehicle trajectories. That is, the collected vehicle trajectories are a coarse representation of full observed vehicle movements.

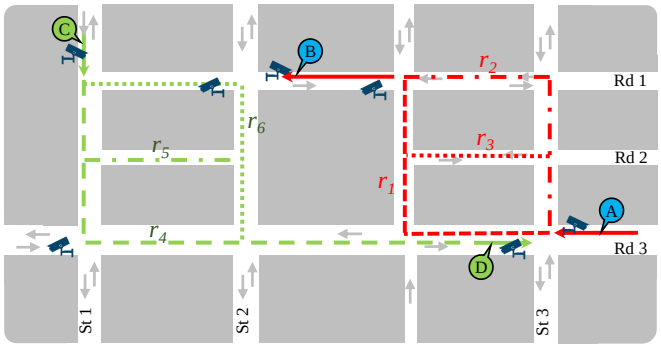


Fig. 4: An example of potential routes given segments of camera based trajectories.

Given transition probability related to road segment r_i and its neighbors (denotes by $r_{n_1}, r_{n_2}, \ldots r_{n_q}$), the traffic volume of r_i can be approximated using volume values of its neighborhoods at last time interval:

$$x^k_i = \sum_{l=1}^{q} p^{k-1}_{i,n_l} x^{k-1}_{n_l}.$$

For simplicity, we define transition matrices as follows:

Transition Matrices. Transition matrices $\mathcal{P} = \{\mathbf{P}^1, \mathbf{P}^2, \cdots, \mathbf{P}^N\}$ are a series of matrices, where $\mathbf{P}^k \in \mathbb{R}^{M \times M}$, $\mathbf{P}^k_{i,j} = p^k_{i,j}$, and M and N are the number of the road segments and time intervals respectively. Namely, $\mathbf{P}^k$ contains transition probability in time interval t_k.

4.4.2 Learning Transition Probability

It is still unfeasible to learn transition probability directly because camera-based trajectories only provide partial observations at monitored points, and details about vehicle movements between cameras are missing if either of two adjacent road segments is not monitored. That is, transition probability related to any unmonitored road segments is missing in camera-based trajectories. We use "route" to denote detailed vehicle movements between nearby cameras. Compared with camera-based trajectories, routes contain detailed vehicle movements between adjacent road segments. To bridge the gap between trajectories and routes, our next step is to estimate missing portions of camera based trajectories. Figure 4 shows two examples of the route inference. $A \rightarrow B$ and $C \rightarrow D$ are two parts of camera based trajectories. The detail traversing routes between A and B (or C and D) remain unknown and need further inference.

A basic approach for route inference given camera based trajectories is formulating it as a path searching problem on the road network. Following this direction, the shortest path can be used as an approximation for route inference. To improve the inference results, restrictions and constraints can be also proposed and added to the searching algorithm to consider complex real-world correlations. Inspired by [44], sampling-based method can be also applied to infer possible routes given camera-based trajectories. However, many dynamical factors and properties of roads are hard to formulate in sampling, such as multiple lanes and interchanges.

To overcome those limitations, we propose a novel method by incorporating a traffic simulator (i.e., *SUMO*) for conditional route inference. *SUMO* is a citywide traffic simulator, which generates vehicles and models their movements on road network in a simulating manner. *SUMO* also considers multiple factors simultaneously during the simulation, such as real-time traffic volume, differences between lanes and traffic signals. So the dynamicity of traffic is fully explored. Moreover, since different routes are planned jointly, the interactions are taken into account as well. Therefore, routes from *SUMO* are more capable to represent real-world vehicle movements. Note that although micro-view results (e.g., one inferred route for a specific vehicle) could be different from ground truths, macro-view statistics (e.g., transition ratios) collected from inferred routes are stable and helpful because they reflect global behaviors. Our experiment in Section 5 demonstrate the effectiveness of *SUMO* compared with path searching algorithm.

Given the road network of a city and the camera-based trajectories observed at all cameras, the simulator can generate complete routes through dynamic simulation following three steps: (1) road network and segments of camera-based trajectories (e.g., $A \rightarrow B$ in Figure 4) are given as the input for *SUMO*; (2) based on these input data, *SUMO* will generate individual vehicles at beginnings of each partial trajectory at corresponding time. Those vehicles will traverse towards their destination road segments (i.e., ends of partial trajectories); (3) detail routes for each vehicle can be collected for the outputs of *SUMO* after the simulation. Given inferred routes, transition ratios in $\mathcal{P}$ can be estimated by Eq. (2). The transition probability inference process is showed in Algorithm 1.

Algorithm 1 Transition Probability inference using *SUMO*

Input: Road Network $\mathcal{G}$ and Camera Based Trajectories Ω.
Output: Transition Matrices $\mathcal{P}$.

$TraSeg \leftarrow \emptyset$
for $\Omega_i \in \Omega$ **do**
  $TraSeg \leftarrow TraSeg \cup Split(\Omega_i)$
$InfRout \leftarrow SUMO(\mathcal{G}, TraSeg)$
for road segment r_i **do**
  for road segment $r_j \in \mathcal{N}(r_i)$ **do**
    for time interval t_k **do**
      $\mathbf{P}^k_{i,j} \leftarrow \frac{\#\ r_i \backslash r_j \text{ in } t_k \text{in } InfRout}{x^k_j}$

4.4.3 Transition Based Volume Inference

The transition matrices $\mathcal{P}$ bridges traffic volume values at the last time interval and those at the current time interval. Namely, $\mathbf{X}^k$ can be linearly transformed to $\mathbf{X}^{k+1}$ using corresponding $\mathbf{P}^k$. We incorporate the transition probabilities $\mathcal{P}$ in our framework to model transitions of citywide traffic volume. Specifically, we propose the simulation module whose loss function is formulated as below:

$$\mathcal{L}_2 = \frac{1}{2}\sum_{k=1}^{N-1} \|\mathbf{X}^{k+1} - \mathbf{P}^k\mathbf{X}^k\|_2^2, \tag{3}$$

where $\|\cdot\|_2^2$ denotes $l2$-norm.

4.5 Optimization

Combining Eq. (1) and Eq. (3), we have the final loss function of the framework **CityVolInf**:

$$\mathcal{L} = \mathcal{L}_1 + \beta\mathcal{L}_2 = \frac{1}{2}\sum_{k=1}^{N}\sum_{i,j=1}^{M} w_{i,j}(x_i^k - x_j^k)^2 + \frac{\alpha}{2}\sum_{k=1}^{N-1}\sum_{i=1}^{M}(x_i^{k+1} - x_i^k)^2 + \frac{\beta}{2}\sum_{k=1}^{N-1}\|\mathbf{X}^{k+1} - \mathbf{P}^k\mathbf{X}^k\|_2^2, \tag{4}$$

where α and β are hyper-parameters that give different emphases on temporal and simulation terms.

Then our goal is to find traffic volume values $\mathbf{X}$ for all road segments, such that the loss function (4) is minimized:

$$\mathbf{X} = \arg\min_{\mathbf{X}} \mathcal{L} \tag{5}$$

To solve the above problem, we adopt an iterative updating algorithm, where each time all values in $\mathbf{X}$ are updated incrementally. Because Eq. (4) is convex, and $\mathbf{X}$ belongs to a convex set (i.e., $\mathbb{R}^{+\ M\times N}$ where $\mathbb{R}^+$ denotes the set of non-negative real number), $\mathbf{X}$ will achieve global optimal value in the end. Specifically, the update rule for $\mathbf{X}$ is as follow:

$$\mathbf{X} \leftarrow \mathbf{X} - \eta\frac{\partial\mathcal{L}}{\partial\mathbf{X}},$$

where η is the learning rate. We repeat above steps until convergence or reaching the maximum number of iterations ψ. The final result of $\mathbf{X}$ contains inferred city-wide traffic volume values.

The time complexity of the optimization is $O(\psi NM^3)$, where ψ is the maximum iteration number of optimization, N is the total number of time intervals, and M is the number of road segments.

5 Experiment

5.1 Datasets

We evaluate **CityVolInf** on a real-world large-scale dataset collected from a provincial capital in China. Table 1 shows the statistics of the dataset in detail.

- **Road Network.** We select the downtown area ($15km \times 10km$) in the provincial capital city as the road network. Except road segments, the road network also contains heterogeneous information such as intersections, ramps, crosswalks, etc. Note that we fix the road network for all experiments except in Section 5.6.2, where we study the performance w.r.t the size of the road network.
- **Surveillance Camera Records.** The dataset contains $405,370,631$ records from $1,704$ surveillance cameras over the period of 08/01/2016 - 08/31/2016. More than 88% records contain valid vehicle plate number according to our statistic.

5.2 Experiment Setting

5.2.1 Preprocessing

Spatial features and contextual information are extracted from the select road network. Discrete features such as road type are incorporated using one-hot representation. $1,248$

TABLE 1: Statistics of surveillance camera dataset

Time span	08/01/2016 - 08/31/2016
Num. of surveillance cameras	1,704
Num. of records	405,370,631
Num. of total vehicles	11,299,927
Avg. num. of vehicles per day	1,155,415

road segments are collected from the selected downtown area after preprocessing, and 257 among them are labeled with observed traffic volume during the 31 days. We filter out samples when ground truth traffic volume values are smaller than 5 when testing our framework. This is a common practice in industry and academy [22]. Road segments with very low traffic volume are of little interests. Moreover, $11,299,927$ camera-based trajectories are extracted from the records.

5.2.2 Hyper-parameter Setting

We set the length of time intervals to 15 minutes. We randomly select 80% monitored road segments and use all their traffic volume as the training set. The traffic volume from the last week (08/25/2016 to 08/31/2016) of remained 20% road segments is used as the testing set (the traffic volume from the time period before 08/25/2016 are used for neither training nor testing). We set maximum iteration number ψ to $1,000$. We set coefficient parameters α, β and η to 4.6, 8.3, and 25, respectively based on grid searching. We use the default setting of *SUMO* in our experiment.

5.3 Evaluation Methods and Metrics

5.3.1 Baselines

CityVolInf is compared with the following baselines: (1) two average-based methods; (2) three regression methods; and (3) two semi-supervised learning methods. For methods that require features, we use the same spatial features extracted from the road network, as described in Section 4.2.

- **Spatial *k*NN.** *Spatial kNN* (S*k*NN) simply selects nearest top k road segments with traffic volume data, and uses the average of their volume values at each time interval as the prediction.
- **Contextual Average.** *Contextual Average* (CA) use the average result of volume values from same-type road segments (e.g., primary, and express way) as the prediction.
- **Linear Regression.** *Linear Regression* (LR) is trained on all road segments with traffic volume. We use the same spatial features for this model. We train one regression model for one time interval.
- **XGBoost.** *XGBoost* (XGB) [47] is a boosting-tree-based method which is popular in data mining community. Similar to LR, we train one model for one time interval.
- **Multiple Layer Perception.** *Multiple Layer Perception* (MLP) is a four-layer fully connected neural network. The hidden units of each layer are 64, 128, 128, 64. The training and testing of MLP are the same to LR and XGB.
- **Basic SSL** [33]. We implement a classical graph-based semi-supervised learning method with loss function $\mathcal{L} = \sum a_{i,j}(x_i^k - x_j^k)^2$. The *Basic SSL* does not consider temporal correlation. We learn the weight $a_{i,j}$ based on the distance between road segments.
- **Spatio-Temporal Semi-Supervised Learning (ST-SSL)** [6]. ST-SSL is state-of-the-art method which applies semi-supervised learning to inferring citywide traffic volume with loop detector and taxi trajectories. We remove the taxi trajectory part because dense trajectories are unavailable in our problem.

5.3.2 Variations.

We further propose three variations of our proposed framework to study the effectiveness of the data-driven module, the simulation module, and the simulation module with data-driven route inference, respectively.

- **CVISimi.** This is a variant of **C**ity**V**ol**I**nf that only contains the **Simi**larity module (i.e., no simulation module).
- **CVISumo.** This variant of **C**ity**V**ol**I**nf only maintains the simulation module (i.e., *SUMO*), while excludes the similarity module.
- **CVIPS.** Inspired by [44], we propose a variant of **C**ity**V**ol**I**nf that samples dense trajectories using **P**ath **S**earching algorithms. More specifically, we first sample potential route candidates using path searching algorithm (i.e., routes no longer than 1.5 times the length of the shortest path). Then the traversing time for each candidate is estimated using speed limits of road segments. Next, traversing time of routes are punished by adding a half minute per left or right turn. Finally, the route with least traversing time is selected. Compared to **CityVolInf**, the difference is that CVIPS alters the way to generate $\mathcal{P}$.

5.3.3 Metrics

We use Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) of inferred traffic volume values to evaluate the performance of inference methods, which are defined as follow:

$$RMSE = \sqrt{\frac{1}{S}\sum_{l=1}^{S}(x_l - \hat{x}_l)^2}, \quad MAPE = \frac{1}{S}\sum_{l=1}^{S}\frac{|x_l - \hat{x}_l|}{\hat{x}_l},$$

where $x_l \in \{x_i^k\}$ is a test sample, $\hat{x}_l$ is the ground truth of x_l, and S denotes the total number of test samples.

Note that RMSE focuses more on larger values, while MAPE receives more punishments from smaller values. Therefore, the combination of two metrics evaluates the performance of inference methods more comprehensively.

5.4 Effectiveness Comparison

5.4.1 Overall Analysis.

We first evaluate the overall performance of **CityVolInf** on collected dataset compared with all baselines. We run all methods 50 times and report the average result and variance for each method. Table 2 shows the comparison results among all methods. Based on the experiment results, we make following observations:

TABLE 2: Comparison with different baselines.

Method	Metrics	
	RMSE	MAPE
Spatial kNN (top 10)	137.441 ± 0.000	0.679 ± 0.000
Spatial kNN (top 100)	131.622 ± 0.000	0.678 ± 0.000
Contextual Average	127.107 ± 0.000	0.696 ± 0.000
Linear Regression	140.366 ± 0.000	0.893 ± 0.000
XGBoost	124.425 ± 0.004	0.669 ± 0.003
MLP	126.210 ± 1.264	0.797 ± 0.013
Basic SSL	133.661 ± 0.001	0.730 ± 0.000
ST-SSL	123.753 ± 0.004	0.601 ± 0.001
CityVolInf	**109.035 ± 0.003**	**0.497 ± 0.001**

TABLE 3: Comparison with different variations.

Method	Metrics	
	RMSE	MAPE
CVISimi	126.215 ± 0.000	0.621 ± 0.000
CVISumo	125.264 ± 0.001	0.651 ± 0.001
CVIPS	122.417 ± 0.001	0.561 ± 0.000
CityVolInf	**109.035 ± 0.003**	**0.497 ± 0.001**

(1) **CityVolInf** significantly outperforms average-based methods (i.e., SkNN and CA) and regression methods (i.e., LR, XGB and MLP) w.r.t both RMSE and MAPE. Obviously, those methods ignore spatiotemporal correlations and transition correlations, thus resulting in poor accuracy.

(2) **CityVolInf** is superior to the Basic SSL. The performance of Basic SSL is limited because the distance-based similarity fails to capture a variety of information including temporal correlations and traffic volume transitions.

(3) **CityVolInf** also significantly outperforms the state-of-the-art ST-SSL method in terms of RMSE and MAPE. ST-SSL incorporates specific spatiotemporal characteristics of traffic volume by building a spatiotemporal affinity graph, thus outperforms other methods by a considerable margin. However, **CityVolInf** reduces RMSE and MAPE by 11.89% and 17.30% compared with ST-SSL, respectively. The huge improvements are mainly attributed to our simulation module, which incorporates incomplete camera based trajectories for modeling dynamic transition relationships explicitly via a traffic simulator.

5.4.2 Variations Study

We further study the effectiveness of the similarity module and the simulation module. The results of different variations are shown in Table 3.

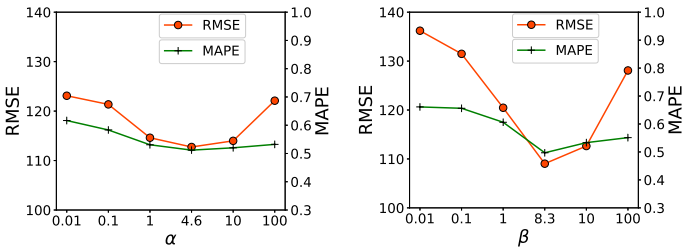
The performances of CVISimi and CVISumo are limited because they both ignore important characteristic of traffic volume. Both spatiotemporal similarities and transition relationships between road segments are essential for a higher accurate inference. CVISimi achieves similar performance compared with ST-SSL in MAPE, while CVISumo has a relatively lower accuracy. One potential reason is that because of the sparsity of cameras, inferring purely with transitions may magnify existing errors, especially at road segments far away from monitored ones. CVIPS replaces the simulator with a path searching based route inference algorithm. As mentioned in Section 4.4, the searching algorithm approximates volume transitions between adjacent road segments using a naive approach, which fails to model dynamical factors. Compared to CVIPS, the simulation approach in **CityVolInf** exhibits better capability in modeling complex real-world scenarios.

5.5 Parameter Sensitivity

5.5.1 Coefficient Parameters.

We explore how the performance of **CityVolInf** changes with respect to regularization coefficients. To evaluate the impacts of α and β on inference performance, we perform the experiments by varying one of either α and β when fix another one. The analysis of temporal parameter α in similarity module and simulation module parameter β w.r.t. RMSE and MAPE are shown in Figure 5. We use the grid search method to find the best parameter settings. As we can see, both RMSE and MAPE of **CityVolInf** first decrease to the minimal values and then increase as the coefficient parameters increasing. This is intuitive because both temporal correlations and transition relationships are essential for a precise inference. As shown in Figure 5(a), the RMSE and MAPE reach low values when α falls around 4.6. Similarly, the RMSE and MAPE achieve minimum values when β is 8.3 in Figure 5(b).

In addition, it is clear that the inference error decreases rapidly with β increasing from 0. This suggests our proposed simulation module contributes a lot to the overall performance.



(a) Parameter α (b) Parameter β

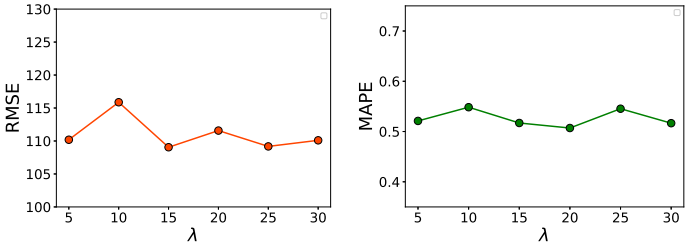
Fig. 5: Results of varying the coefficient parameters.

5.5.2 Number of Reachable Road Segments λ.

We study the impact of hyper-parameter λ that controls the number of reachable nearby road segments when building affinity graph. From Figure 6, we observe that **CityVolInf** nearly keeps stable performance with respect to λ in terms of both RMSE and MAPE. This is may because **CityVolInf** incorporates multiple spatiotemporal factors simultaneously when building the affinity graph, resulting in lower sensitivity with respect to one specific kind of spatial edges. Although the number of reachable edges is altered, other kinds of spatial edges could contribute more when learning spatial weights. Moreover, temporal correlations also play important roles in inference as well. Finally, the entire similarity module contributes consistently to the overall performance.

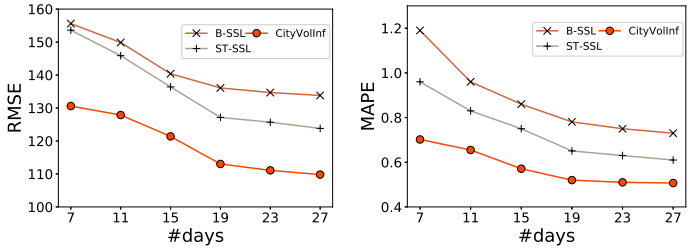
5.5.3 Training and Testing Ratio

Figure 7 shows the results of all methods in terms of RMSE and MAPE by varying the ratio of training set from 50% to 90%. As we can see, our method continuously performs better than all baselines even with very few training data.



(a) RMSE (b) MAPE

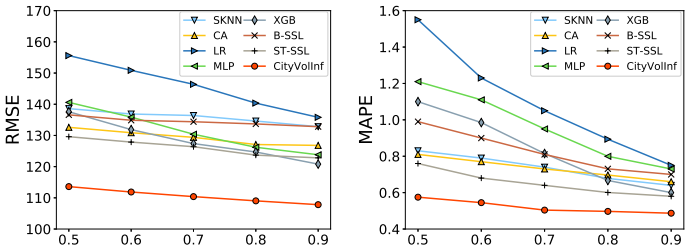
Fig. 6: Results of varying the hyper-parameter λ.



(a) RMSE (b) MAPE

Fig. 8: Results of varying the length of training data.

In addition, the RMSE and MAPE values of all methods increase as the ratio of training data increases. Although the performances of some baseline methods (e.g., LR, XGB, MLP) improves obviously when using more training data, our proposed **CityVolInf** still exhibits significant superiority.



(a) RMSE (b) MAPE

Fig. 7: Results of varying the ratio of training data.

5.5.4 *Training and Testing Length*

We next evaluate the performance of our method in terms of RMSE and MAPE by varying the length of training set from one week to four weeks compared to SSL-based methods. The rest of data is treated as testing data. Namely, the length of testing data varies from three weeks to four days. Figure 8 shows the RMSE and MAPE results of our method and SSL-based baselines with respect to the length of training set. As expected, our method consistently outperforms SSL-based baselines in all cases. As the training set increases, the inference accuracy of our method increases in both terms of RMSE and MAPE, and then keeps stable when the training set reaches about three weeks. Moreover, our method is more better than SSL-based baselines in a training set with less length, which demonstrates that our proposed simulation module is useful for inferring traffic volume as an important complement to the similarity module.

5.6 Scalability

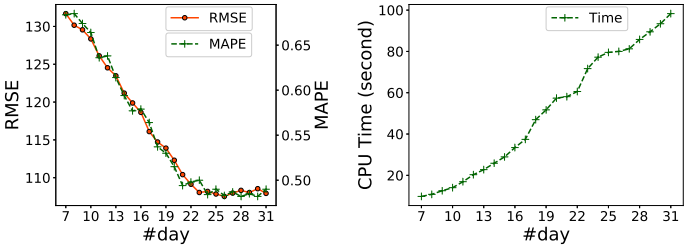
We conduct experiments to study the scalability of **CityVolInf** in terms of inference accuracy and running time by varying the number of time intervals N and the number of road segments M. We fix the testing set in all experiment as described in Section 5.2, such that all results are comparable.

5.6.1 *Varying the Number of Time Intervals N*

First we study the scalability w.r.t N. While fixing the testing set, the total number of time intervals in training set increases from 672 (i.e., 7 days corresponding to testing set) to 2976 (i.e., 31 days) by 96 (i.e., one-day time span) in reverse chronological order. Figure 9 shows RMSE, MAPE and running time of **CityVolInf** with the number of training data.

From Figure 9(a), similarly with experiment in Section 5.5.4, both RMSE and MAPE on the testing set keep declining at first, then convergence after N reaching a certain threshold (about 3 weeks). This indicates that using about 3-week historical data is sufficient for **CityVolInf** to model temporal correlations and transition relationships.

Figure 9(b) shows the total running time of the optimization w.r.t. N. We can see that the running time keeps linearly increasing. This is expected because the time complexity of the optimization is $O(\psi N M^3)$ which is linear with the number of time interval N.

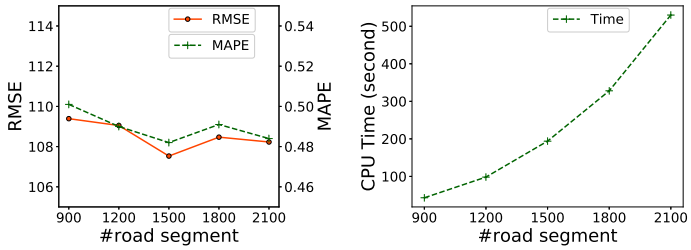


(a) RMSE and MAPE (b) Running time

Fig. 9: Scalability evaluation w.r.t. N.

5.6.2 *Varying the Number of Road Segments M*

We further vary the size of selected road network so that the number of road segments increases from 900 to 2,100 by 300 each time. As shown in Figure 10, the running time nearly keeps linear increasing w.r.t. M^3, which is consistent with our previous analysis on algorithm complexity. The inference accuracy of **CityVolInf** is relatively stable w.r.t. M. The reason is that the inference of traffic volume for a road segment does not rely on faraway road segments. Furthermore, the simulation module defines traffic transitions only on adjacent road segments, which is not affected by size of road networks basically. This also suggests our proposed framework is robust and effective w.r.t. the size of road network.



(a) RMSE and MAPE (b) Running time

Fig. 10: Scalability evaluation w.r.t. M.

6 Conclusion

Surveillance cameras are widely used to monitor urban traffic situations in modern cities. However, the coverage is still limited because of high costs. In this paper, we introduce a novel framework **CityVolInf** for citywide traffic volume inference with road network and surveillance camera records. We first construct an affinity graph of road segments at different time intervals according to their spatial and temporal similarities. We further incorporate a novel simulation module that utilizing rich and complex road network information and incomplete camera based trajectories to model traffic volume transitions between adjacent road segments, which results in a significant improvement of accuracy. We evaluate our method on a real-world large-scale traffic dataset collected in a provincial capital in China. **CityVolInf** exhibits extraordinary accuracy compared with all baselines. Experimental results on parameter sensitivity, robustness and scalability demonstrate the advantages of **CityVolInf** with respect to citywide traffic volume inference.

In future work, we plan to extend our framework to an online inference manner to reduce the running time for working more efficiently in real time systems. In addition, we would like to investigate the relation between the similarity and simulation modules to be generalized to different situations.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China under Grant No. 61773331, the National Science Foundation under Grant Nos. 1544455, 1652525 and 1618448, and the China Scholarship Council under Grant No.: 201608370018. Zhenhui Li would like to acknowledge the support from Haile Family Early Career Professorship. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

[1] H. Hu, G. Li, Z. Bao, Y. Cui, and J. Feng, "Crowdsourcing-based real-time urban traffic speed estimation: From trends to speeds," in *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE, 2016, pp. 883–894.

[2] Z. Shan, D. Zhao, and Y. Xia, "Urban road traffic speed estimation for missing probe vehicle data based on multiple linear regression model," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013, pp. 118–123.

[3] J. Zheng and L. M. Ni, "Time-dependent trajectory regression on road networks via multi-task learning," in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press, 2013, pp. 1048–1055.

[4] J. Aslam, S. Lim, X. Pan, and D. Rus, "City-scale traffic estimation from a roving sensor network," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM, 2012, pp. 141–154.

[5] X. Yi, Y. Zheng, J. Zhang, and T. Li, "St-mvl: filling missing values in geo-sensory time series data," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 2704–2710.

[6] C. Meng, X. Yi, L. Su, J. Gao, and Y. Zheng, "City-wide traffic volume inference with loop detector data and taxi trajectories," in *Proceedings of ACM International Conference on Advances in Geographical Information Systems*, 2017.

[7] X. Zhan, Y. Zheng, X. Yi, and S. V. Ukkusuri, "Citywide traffic volume estimation using trajectory data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 272–285, 2017.

[8] A. Gühnemann, R.-P. Schäfer, K.-U. Thiessenhusen, and P. Wagner, "Monitoring traffic and emissions by floating car data," 2004.

[9] "*SUMO:*," http://sumo.sourceforge.net/, 2017.

[10] S. Lee and D. Fambro, "Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1678, pp. 179–188, 1999.

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

[12] B. Williams, "Multivariate vehicular traffic flow prediction: evaluation of arimax modeling," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1776, pp. 194–200, 2001.

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

[14] B. Ghosh, B. Basu, and M. O'Mahony, "Multivariate short-term traffic flow forecasting using time-series analysis," *IEEE transactions on intelligent transportation systems*, vol. 10, no. 2, pp. 246–254, 2009.

[15] D. Deng, C. Shahabi, U. Demiryurek, L. Zhu, R. Yu, and Y. Liu, "Latent space model for road networks to predict time-varying traffic," *Proceedings of the 22th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016.

[16] F. Wu, H. Wang, and Z. Li, "Interpreting traffic dynamics using ubiquitous urban data," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2016, p. 69.

[17] B. Pan, U. Demiryurek, and C. Shahabi, "Utilizing real-world transportation data for accurate traffic prediction," in *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 595–604.

[18] Y. Tong, Y. Chen, Z. Zhou, L. Chen, J. Wang, Q. Yang, and J. Ye, "The simpler the better: A unified approach to predicting original taxi demands on large-scale online platforms," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017.

[19] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

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

[21] H. Yao, X. Tang, H. Wei, G. Zheng, Y. Yu, and Z. Li, "Modeling spatial-temporal dynamics for traffic prediction," *arXiv preprint arXiv:1803.01254*, 2018.

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

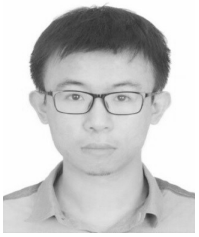
[23] R. Jiang, X. Song, Z. Fan, T. Xia, Q. Chen, S. Miyazawa, and R. Shibasaki, "Deepurbanmomentum: An online deep-learning system for short-term urban mobility prediction." in *AAAI*, 2018.

[24] S. Park, J. Serra, E. F. Martinez, and N. Oliver, "Mobinsight: A framework using semantic neighborhood features for localized

interpretations of urban mobility," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 8, no. 3, p. 23, 2018.
[25] P. Wang, Y. Fu, J. Zhang, X. Li, and D. Lin, "Learning urban community structures: A collective embedding perspective with periodic spatial-temporal mobility graphs," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 9, no. 6, p. 63, 2018.
[26] L. Qu, Y. Zhang, J. Hu, L. Jia, and L. Li, "A bpca based missing value imputing method for traffic flow volume data," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 985–990.
[27] L. Qu, L. Li, Y. Zhang, and J. Hu, "Ppca-based missing data imputation for traffic flow volume: A systematical approach," *IEEE Transactions on intelligent transportation systems*, vol. 10, no. 3, pp. 512–522, 2009.
[28] L. Li, Y. Li, and Z. Li, "Efficient missing data imputing for traffic flow by considering temporal and spatial dependence," *Transportation research part C: emerging technologies*, vol. 34, pp. 108–120, 2013.
[29] M. T. Asif, N. Mitrovic, J. Dauwels, and P. Jaillet, "Matrix and tensor based methods for missing data estimation in large traffic networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1816–1825, 2016.
[30] W. Ruan, P. Xu, Q. Z. Sheng, N. J. Falkner, X. Li, and W. E. Zhang, "Recovering missing values from corrupted spatio-temporal sensory data via robust low-rank tensor completion," in *International Conference on Database Systems for Advanced Applications*. Springer, 2017, pp. 607–622.
[31] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 25–34.
[32] M. Culp and G. Michailidis, "Graph-based semisupervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 1, pp. 174–179, 2008.
[33] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, 2003, pp. 912–919.
[34] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in neural information processing systems*, 2004, pp. 321–328.
[35] A. Argyriou, M. Herbster, and M. Pontil, "Combining graph laplacians for semi–supervised learning," in *Advances in Neural Information Processing Systems*, 2006, pp. 67–74.
[36] Y. Yamaguchi, C. Faloutsos, and H. Kitagawa, "Omni-prop: Seamless node classification on arbitrary label correlation." in *AAAI*, 2015, pp. 3122–3128.
[37] J. Kwon, P. Varaiya, and A. Skabardonis, "Estimation of truck traffic volume from single loop detectors with lane-to-lane speed correlation," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1856, pp. 106–117, 2003.
[38] D. Wilkie, J. Sewall, and M. Lin, "Flow reconstruction for data-driven traffic animation," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 89, 2013.
[39] X. Zhan, R. Li, and S. V. Ukkusuri, "Lane-based real-time queue length estimation using license plate recognition data," *Transportation Research Part C: Emerging Technologies*, vol. 57, pp. 85–102, 2015.
[40] X. Tang, B. Gong, Y. Yu, H. Yao, Y. Li, H. Xie, and X. Wang, "Joint modeling of dense and incomplete trajectories for citywide traffic volume inference," in *The World Wide Web Conference*. ACM, 2019, pp. 1806–1817.
[41] G. S. Thakurzx, P. Huiz, and A. Helmyx, "Modeling and characterization of urban vehicular mobility using web cameras," in *2012 Proceedings IEEE INFOCOM Workshops*. IEEE, 2012, pp. 262–267.
[42] D. Kumar, H. Wu, S. Rajasegarar, C. Leckie, S. Krishnaswamy, and M. Palaniswami, "Fast and scalable big data trajectory clustering for understanding urban mobility," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–14, 2018.
[43] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 2012, pp. 1144–1155.
[44] P. Banerjee, S. Ranu, and S. Raghavan, "Inferring uncertain trajectories from partial observations," in *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 2014, pp. 30–39.
[45] N. Yang and P. S. Yu, "Efficient hidden trajectory reconstruction from sparse data," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 821–830.
[46] "*Openstreetmap*:," https://www.openstreetmap.org/, 2017.
[47] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 785–794.

Yanwei Yu (S'12-M'16) received his Ph.D. degree in Computer Science from University of Science and Technology Beijing, China in 2014. From 2016 to 2018, he was a postdoctoral researcher at the College of Information Sciences and Technology, Pennsylvania State University. He is currently an associate professor at the Department of Computer Science and Technology, Ocean University of China. His research interests include data mining, machine learning and distributed computing.

Xianfeng Tang received his bachelor degree from the School of Computer Science and Technology, University of Science and Technology of China in 2016. He is currently a third-year Ph.D. candidate at College of Information Science and Technology, Pennsylvania State University. His research interests include machine learning and spatial-temporal data mining.



Huaxiu Yao received his bachelor degree from the School of Electronic Science and Engineering at University of Electronic Science and Technology of China in 2016. He worked as research intern at Didi Chuxing AI Labs in 2017 and Tencent AI Lab in 2018. He is currently a second-year Ph.D. candidate at College of Information Science and Technology, Pennsylvania State University. His research spans across data mining and machine learning.

Xiuwen Yi received his bachelor and Ph.D. degrees in computer science from the School of Information Science and Technology, Southwest Jiaotong University in 2013 and 2018, respectively. He interned in Urban Computing Group, Microsoft Research Asia from 2014 to 2017. From 2017 to 2018, he was a visiting scholar at the College of Information Sciences and Technology, Pennsylvania State University. He is currently a data scientist at JD Urban Computing Business Unit. His research interests include urban computing and deep learning.

Zhenhui Li (M'12) received her Ph.D. degree from the Department of Computer Science at University of Illinois at Urbana-Champaign in 2012. Before that, she received her Bachelor degree from Department of Computer Science at Shanghai Jiao Tong University in 2007. She interned at Microsoft Research at Silicon Valley in 2011, Facebook at Palo Alto in 2009, Yahoo! at Santa Clara in 2008 and Google China at Beijing in 2006. She is currently an associate professor with College of Information Sciences and Technology of Pennsylvania State University. Her research interests include data mining, machine learning and artificial intelligent.