Joint Modeling of Dense and Incomplete Trajectories for Citywide Traffic Volume Inference

Xianfeng Tang[1], Boqing Gong[2], Yanwei Yu[3], Huaxiu Yao[1], Yandong Li[4]
Haiyong Xie[5], Xiaoyu Wang[6]
[1]The Pennsylvania State University, USA
[2] Tencent AI Lab
[3] Yantai University, China
[4]University of Central Florida, USA
[5]University of Science and Technology of China, China
[6]Intellifusion Inc.
{tangxianfeng,boqinggo,lyndon.leeseu}@outlook.com,haiyong.xie@ieee.org
{yuyanwei0530,yhx662012,fanghuaxue}@gmail.com

ABSTRACT

Real-time traffic volume inference is key to an intelligent city. It is a challenging task because accurate traffic volumes on the roads can only be measured at certain locations where sensors are installed. Moreover, the traffic evolves over time due to the influences of weather, events, holidays, etc. Existing solutions to the traffic volume inference problem often rely on dense GPS trajectories, which inevitably fail to account for the vehicles which carry no GPS devices or have them turned off. Consequently, the results are biased to taxicabs because they are almost always online for GPS tracking. In this paper, we propose a novel framework for the citywide traffic volume inference using both dense GPS trajectories and incomplete trajectories captured by camera surveillance systems. Our approach employs a high-fidelity traffic simulator and deep reinforcement learning to recover full vehicle movements from the incomplete trajectories. In order to jointly model the recovered trajectories and dense GPS trajectories, we construct spatiotemporal graphs and use multi-view graph embedding to encode the multi-hop correlations between road segments into real-valued vectors. Finally, we infer the citywide traffic volumes by propagating the traffic values of monitored road segments to the unmonitored ones through masked pairwise similarities. Extensive experiments with two big regions in a provincial capital city in China verify the effectiveness of our approach.

CCS CONCEPTS

• **Information systems → Spatial-temporal systems**; **Data mining**; *Information extraction.*

KEYWORDS

Traffic volume inference; spatiotemporal; smart city transportation

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313621

ACM Reference Format:
Xianfeng Tang[1], Boqing Gong[2], Yanwei Yu[3], Huaxiu Yao[1], Yandong Li[4] and Haiyong Xie[5], Xiaoyu Wang[6]. 2019. Joint Modeling of Dense and Incomplete Trajectories for Citywide Traffic Volume Inference. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313621

1 INTRODUCTION

With the vast deployment of sensors, such as loop detectors and surveillance cameras, and the rapid development of data storage techniques, an intelligent city can collect and store a large amount of traffic volume data on the daily basis. It is vital to not only monitor "visible" traffic volumes in real time but also infer those unseen by the sensors. Information extracted from the traffic volume data may contribute to a wide range of urban applications, such as route selection, traffic control, and urban planning, etc.

However, the sensors are often displaced away from each other to a certain distance, and some of them could fail now and then. As a result, the citywide traffic volume data inevitably have lots of missing values, causing challenges to many downstream applications. In this paper, we aim to develop techniques for the citywide traffic volume inference — in other words, to estimate the traffic volume of each and every road segment in every time interval in the city.

The citywide traffic volume inference is a very challenging problem for two major reasons. Firstly, sensors to monitor the traffic are not only sparse but also non-uniform, resulting in relatively low spatial coverage [15]. It is difficult to obtain any historic traffic data for the currently unmonitored roads. Secondly, traffic patterns evolve over time and could suddenly change rapidly due to big events. The high dynamism of traffic makes it difficult to infer the traffic volume data by drawing similarities between different road segments or across various time intervals.

Existing works on inferring the traffic volume mostly fall into the category of missing (spatiotemporal) data inference. A basic and representative framework is to estimate the missing values by linear interpolation [3, 9, 26, 43]. However, those approaches fail to maintain the global consistency of the inferred results and assume simple structures underlying the historic spatiotemporal data. Another frequently used method is collaborative filtering [39].

However, this approach fails for the unmonitored road segments which have no historical information.

Some recent studies use dense trajectories from mobile devices for traffic volume inference. Zhan et al. estimate the citywide traffic volume by combining GPS trajectories with road networks, point-of-interest information, and weather conditions. Meng et al. characterize the similarities between roads using travel speeds extracted from taxi trajectories. However, the dense trajectories, no matter extracted from GPS or taxi, are biased representations of the real traffic. After all, taxi is only a small portion of the total vehicles and not all vehicles carry or turn on the GPS devices.

Partially observed trajectories, which are captured by the static and discrete sensors (e.g., a vehicle tracked by a camera network), are notably under-explored by the existing works. Unlike the dense trajectories (e.g., generated by "active" GPS devices), the partially observed trajectories are incomplete because they are obtained from the "passive" static sensors (e.g., cameras). As a result, they are available for almost all kinds of vehicles no matter they have GPS devices installed or not. Compared with dense trajectories, such incomplete trajectories are collected in a much bigger scale everyday. For instance, the surveillance system recognizes and provides the incomplete trajectories of more than 1.1 million unique vehicles in a provincial capital city in China every single day at the time of this paper written. To the best of our knowledge, the incomplete trajectories have not been utilized before for the traffic volume inference.

Of course, it is more involved to model the spatiotemporal patterns of incomplete trajectories than their dense counterparts. The uncertainties between monitor points have to be taken care of, as well as the noisy measurements at the monitor points. There exist some research tackling the uncertainty problem. For example, Zheng et al. investigate how to reduce the uncertainty in low-sampling-rate trajectories by inferring the possible routes therein. Banerjee et al. employ road-network constraints to summarize all probable routes in a holistic manner. Li et al. use the GPS snippets to help recover the full trajectories. However, these methods require that every road segment has appeared at least once in the past in order to guide the recovering procedure of the full trajectories. Besides, their performance significantly degrades when the traffic pattern changes, e.g., due to big events or severe weather.

To address the aforementioned challenges, we propose a novel framework to **J**ointly **M**odel the **D**ense and **I**ncomplete trajectories (JMDI) for the citywide traffic volume inference. Specifically, we first recover detailed vehicle movements given the incomplete trajectories using a high-fidelity city traffic simulator. We tune the simulator's parameters using deep reinforcement learning so as to automatically identify the proper simulator state which matches the real data. After that, two spatiotemporal graphs are constructed using the recovered trajectories and the dense GPS trajectories, respectively. The graphs are a natural choice for modeling the dynamism of traffic volumes of road segments and over time. Meanwhile, the graphs enable us to conveniently embed the road segments (nodes of the graph) to continuous-value vectors. Finally, we use these embeddings to construct pairwise similarities between the road segments and then propagate the known traffic volumes to the road segments of unknown traffic.

We conduct extensive experiments on large-scale real-world traffic datasets collected from a provincial capital city in China. The dense trajectories are available for GPS-enabled taxi. For the incomplete trajectories, we extract partially observed information from the surveillance cameras as a result of the plate number identification technique [7]. We evaluate the proposed framework JMDI against competitive baselines on two selected regions of the city and during different time periods. We also study some ablations of our approach to highlight the effectiveness of main components of JMDI.

We summarize our main contribution as follows.

- We propose a novel framework, called JMDI, to infer citywide traffic volumes using both dense and incomplete traffic trajectories.
- We propose to use a high-fidelity traffic simulator to recover full vehicle movements from incomplete trajectories. Moreover, in order to efficiently tune the simulator towards the real data, we design a deep reinforcement learning algorithm to automatically set the parameters of the simulator.
- We present a joint embedding method to learn meaningful representations of road segments using spatiotemporal graphs. We further use a semi-supervised propagation method to infer citywide traffic volume values.
- We conduct extensive experiments on large-scale real-world traffic datasets to validate the proposed approach.

To the best of our knowledge, this work is the first to jointly model the dense and incomplete trajectories for the citywide traffic volume inference. Since the dense GPS trajectories are biased to taxi, the incomplete trajectories obtained from other sensors show strong complement to their dense counterparts. Jointly, they give rise to better results than either individually.

2 RELATED WORK

2.1 Traffic Volume Inference

Some studies [9, 26, 43] use linear regression models to infer missing traffic speed or travel time based on taxi trajectories. Aslam et al. [3] 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., [2, 13, 22, 23]) or collaborative filtering (CF) (e.g., [25, 30, 39]) 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 [23], Probabilistic PCA [13, 22] and FPCA [2]. CF-based methods recover missing values by decomposing spatiotemporal data into the product of low-rank matrices. Yi et al. [39] fill missing values in geo-sensory time series using CF from multiple spatial and temporal perspectives. Ruan et al. [25] use tensor completion to recover missing values in spatiotemporal sensory data. Wang et al. [30] propose a tensor factorization method to estimate the missing travel time for drivers on road segments. *However, both PCA-based and CF-based methods rely on historical data when filling in. They are unable to handle our problem since the traffic volume of unmonitored road segments are totally missing.*

Graph-based Semi-supervised learning (SSL) method [6] has been widely applied for unlabeled data inference, which can be used for inferring missing values. Label propagation [46], a classic semi-supervised method, infers unobserved labels by propagating existing labels on an affinity graph. Other SSL based methods [1, 35, 45] have been proposed to model the similarities of vertices in the affinity graph. *Although Graph-based 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 [12, 16, 34], surveillance cameras [40], or float car trajectories [3, 8, 41]. Studies [12, 34, 40] 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. [41] 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. [16] propose ST-SSL that predicts city-wide traffic volume values using loop detector incorporating taxi trajectories. They first build a spatiotemporal affinity graph based on travel speed pattern and spatial correlations extracted from loop detector and taxi trajectories. Then they infer traffic volume using semi-supervised learning method on the spatiotemporal affinity graph. *However, both [41] and ST-SSL [16] require full observation of trajectories, thus they are unable to utilize incomplete counterparts.*

2.2 Trajectory Recovery

Several methods (e.g., [4, 33, 44]) have been proposed to predict complete trajectories from partial observations. Zheng et al. [44] investigate how to reduce the uncertainty in low-sampling-rate trajectories. This approach requires every road segment appearing in historical data, which is impossible for incomplete trajectories. Banerjee et al. [4] infer uncertain trajectories from network-constrained partial observations by summarizing all probable routes in a holistic manner. However, their method ignores many important factors such as vehicle interaction and road constraints. *More important, these methods require historical trajectories as input, which cannot be applied to recovering routes in our problem.*

3 NOTATIONS AND PROBLEM STATEMENT

Let $\{r_1, r_2, \ldots, r_m\}$ denote road segments of a road network in a city region. Each road segment is a directed short portion of a road. The traffic volume of a road segment can be represented using the total number of vehicles traversing through it during a fixed time interval. We split the whole time period (e.g, one week) into equal-length continuous time intervals and use vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ to denote the traffic volume values of the road segment r_i during the past n continuous time intervals. Note that the traffic volume values are only available at monitored road segments.

Different sources of trajectories (i.e., incomplete observations from cameras or GPS trackers of taxi) are denotes by $\Delta^{\mathrm{S}} = \{\delta_i\}$, where the superscript S stands for the source S. A trajectory $\delta_i = p_{i,1} \rightarrow p_{i,2} \rightarrow \cdots \rightarrow p_{i,s}$ is a list of s points in chronological order, where each point $p = \langle l, t \rangle$ is a location coordinate l (i.e., latitude and longitude) and a time stamp t. We denote by Δ^{D} and Δ^{I} dense and incomplete trajectories, respectively. Note that points contained in the incomplete trajectories Δ^{I} are restricted to monitor points — therefore they form a fixed finite set.

The citywide traffic volume inference problem is defined as follows: Given the road network, observed traffic volume values $\{x_{it}\}$ at monitored road segments, and multiple sources of trajectories Δ^{S} ($\mathrm{S} \in \{\mathrm{D}, \mathrm{I}\}$), our goal is to infer the traffic volume values of any unmonitored road segment at any time interval.

4 APPROACH

Figure 1 illustrates the proposed framework JMDI. First, we design an algorithm to recover full trajectory from incomplete observations using a traffic simulator, whose parameters are tuned by deep reinforcement learning. Next, we construct spatiotemporal graphs from the dense and incomplete trajectories. In order to model the spatiotemporal dynamics of traffic volume, we then employ a multi-view graph embedding technique [24] to learn representation vectors for road segments. Finally, we present a propagation method which infers the citywide traffic volume values according to pairwise similarities between different road segments. In the following sections, we introduce each major component of our solution in detail.

4.1 Trajectory Recovery

In this section, we introduce the recovery process for incomplete trajectories. Instead of data-driven algorithms, we incorporate a high-fidelity traffic simulator [11] for the recovery. Different from previous trajectory recovery methods (e.g., [4, 44]), the simulator jointly models the influence of traffic rules as well as vehicle interactions. Furthermore, we propose a novel deep reinforcement learning [19] algorithm that improves the accuracy of trajectory recovery by tuning time-aware hyper-parameters of the simulator.

4.1.1 Trajectory Recovery via Simulator. We first estimate the missing portions of incomplete trajectories by using rich spatiotemporal movement patterns. Unlike the existing methods [4, 14, 36, 44], our solution relies on a traffic simulator [11], which naturally observes road network constraints (e.g., no turn on red at some intersections) and the global consistency of traffic in the road network. It also captures the influence of vehicle interactions.

Specifically, we simulate all vehicles simultaneously in real-world road networks using the traffic simulator named *SUMO* [11], conditioning on observations in the incomplete trajectories. Given an incomplete trajectory $\delta_i = p_{i,1} \rightarrow p_{i,2} \rightarrow \cdots \rightarrow p_{i,s}$, the simulator initializes a vehicle at the corresponding time and location of each point $p_{i,j}$. Then, the vehicle moves toward $p_{i,j+1}$ along the most possible route selected by the simulator. Multiple factors, such as distance, speed, traffic jams, etc., are considered jointly to find the best route. The simulated results for all points $p_{i,j} \rightarrow p_{i,j+1}$ are collected to fill in the missing values in the incomplete trajectories.

However, blindly using the simulator for trajectory recovery without careful validation is not rigorous. Therefore, a novel metric is proposed for trajectory recovery evaluation. Taking incomplete trajectory δ_i as an example, we first measure the relative arrival time $t_{i,j}$ for $p_{i,j} \in \delta_i$, using time stamps provided by δ_i. Here $t_{i,j}$ equals to the time consumption of traversing between $p_{i,j}$ and $p_{i,1}$ ($t_{i,1} = 0$) captured by real-world sensors. Since vehicles would

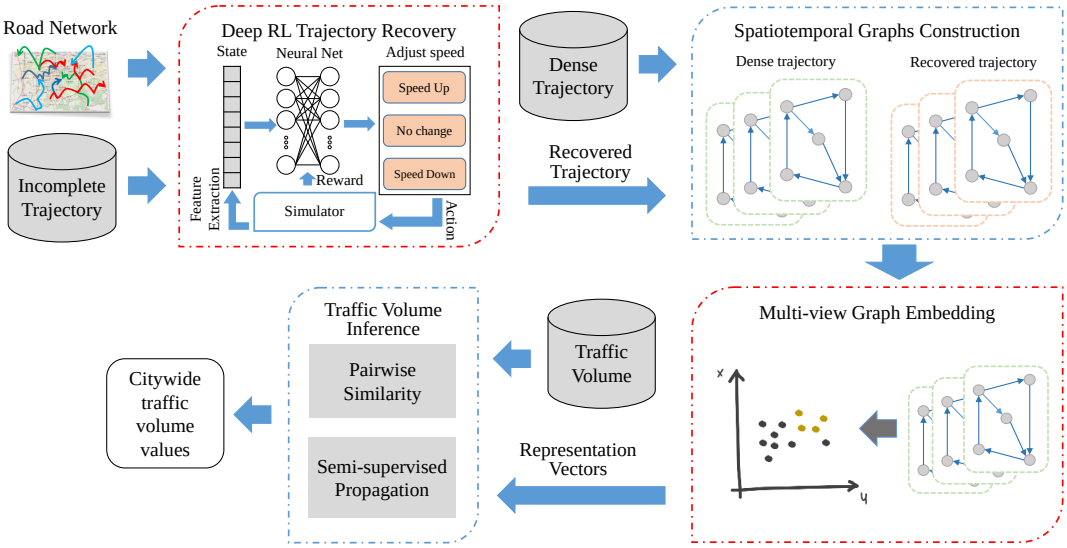

Figure 1: Overview of the proposed JMDI framework.

Table 1: Examples of parameters in *SUMO*.

Parameters	Descriptions
stop	Lets the vehicle stop at the given edge
change lane	Change to the lane with the given index
slow down	Change the speed smoothly
speed	Sets the vehicle speed to the given value
change target	Change the vehicle's destination edge
change route	Assigns the vehicle's new route
speed factor	Sets the vehicle's speed factor
max speed	Sets the vehicle's maximum speed

traverse through each monitor point successively in the simulating environment, we compute the relative arrival time at each $p_{i,j}$ again, using time stamps from simulating results. Similarly, let $t^{\text{simu}}_{i,j}$ denote the relative arrival time at $p_{i,j}$ of the simulated vehicle. Given those two kinds of relative arrival time $t_{i,j}$ and $t^{\text{simu}}_{i,j}$, the trajectory recovery accuracy can be estimated using the following equation:

$$err = \sum_{\delta_i \in \Delta^{\text{I}}} \sum_{p_{i,j} \in \delta_i} |t^{\text{simu}}_{i,j} - t_{i,j}| / (\sum_{\delta_i \in \Delta^{\text{I}}} \sum_{p_{i,j} \in \delta_i} \mathbf{1}). \quad (1)$$

The intuition behind Equation 1 is that larger errors will be generated if the recovered trajectories differ a lot from the real ones (i.e., larger $|t^{\text{simu}}_{i,j} - t_{i,j}|$). And the recovered trajectories have low *err* if the simulated vehicles arrive at monitor points sharply.

4.1.2 *Deep Reinforcement Learning for Parameter Tuning.*

There are a few major parameters of the simulator (see Table 1) we have to tune in order to achieve good simulation results (in other words, small *err*). For example, changing the speed of vehicles could directly affect the arrival time at each monitored point in the simulating environment, and modifying the lane-changing rules could lead to traffic jams. In order to simulate as realistic as possible, parameter tuning is key. Unfortunately, it is nontrivial to tune the parameters because they are inter-correlated and their effects on the traffic volume are delayed. For example, we may speed up a vehicle anytime but the results of the acceleration cannot be validated until the vehicle arrives at the next monitor point. Besides, the change of speed of one vehicle may change the speed of others.

To address the above challenges, we propose a reinforcement learning (RL) algorithm to tune the simulator's parameters such that the simulated trajectories can better match the sparsely monitored real data. The RL algorithm successfully avoids the manual configuration of the simulator in a tedious trial-and-error manner.

The general goal of reinforcement learning is to maximize the expected cumulative rewards in a sequential decision making process. Given a set of states $\{s_i\}$ and actions $\{a_i\}$, an action a is taken by the agent following a certain policy π after seeing the current state s in each step. The expected return of each potential action a under state s is evaluated using the Bellmen equation [31],

$$Q_\pi(s, a) = \mathbf{E}(\sum_{i=1}^{\infty} \gamma^{i-1} R_i | S_0 = s, A_0 = a, \pi), \quad (2)$$

where γ is the discount factor for future rewards. The goal of reinforcement learning is to maximize the expectation of accumulative future rewards through the sequential process.

Parameter tuning of the simulator can be naturally considered as a sequential decision making process, where states are the snapshots of the simulating environment and actions are all possible ways of parameter tuning. We learn a policy that continuously takes positive action to change the parameters of the simulator in each step. Note that it is intractable to use the Bellmen Equation (Equation 2) to directly optimize the Q-value as it has to track all possible state-action combinations. Instead, we employ Deep Q-Learning (DQN) [19] which approximates the Q-value using a deep neural

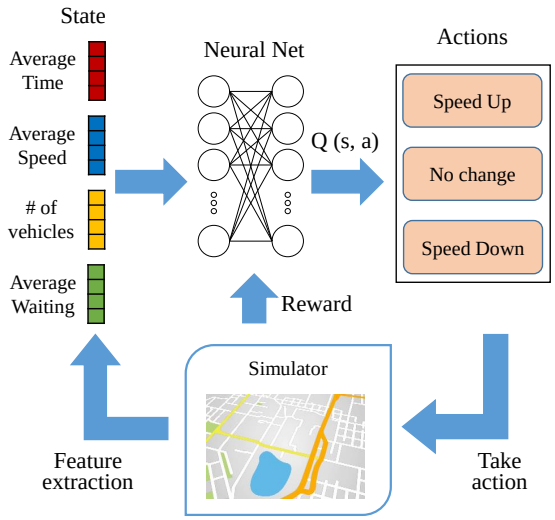

Figure 2: Deep reinforcement learning architecture for tuning the vehicles' speeds in the simulator.

network. Given a state s, the output of DQN is the approximated Q-values for the actions $Q(s, a; \theta)$, where θ are neural network weights in DQN. The greedy action is chosen as $\arg\max_{a \in A} Q(s, a; \theta)$. In our context, we use this action to tune the simulator's parameters.

Figure 2 illustrates the DQN framework for trajectory recovery. Real-valued feature vectors extracted from the simulator are used to describe the states. We construct a two-layer fully-connected neural network for DQN to approximate the Q-value. During the training process, weights of the neural network are optimized iteratively to reduce the discrepancy between the optimal Q-value and the predicted one, using gradient descent of the loss function. The details of states, actions, rewards, and the training are described below:

- **States**: Many existing deep reinforcement learning methods (e.g. [29, 32, 42]) use real-valued vector to represent state. Following the same practice, we first extract representative features using the API of the simulator. Since the number of vehicles on the road segments directly describes the traffic volume distribution, we first retrieve their values from the simulator. Next, average traversing time and speed are acquired as the traversing speed and time are closely related to the traffic volume. In addition, we retrieve average waiting time, which measures the average stopping time of every vehicle in the immediate past simulating step. Those four types of values are retrieved from each road segment and then concatenated into a real-valued vector as the state representation. Given m road segments, the dimension of the state feature vector is $4m$.
- **Actions**: Recall that the action with the largest expected cumulative reward (Q-value) will be selected by the agent in each simulating step. To construct the action set, we change the speeds of vehicles which are one of the most important factors that affect the traversing trajectory. However, directly tuning the speed for every vehicle in the simulator is prohibitive because there could be thousands of vehicles traveling simultaneously. To simplify the setting, we do not change the speed continuously for every vehicle. Instead, the speed limits (meters/second) for different vehicles are modified within a certain range in each simulation step. The actions include increasing the speed limit by $+1$, 0, or -1 (i.e., decrease by 1) for each vehicle. To reduce the action space, we group the vehicles according to their sizes (i.e., sedans, SUVs, and large trucks) and assign the same speed limit to a group of vehicles. Hence, the action space consists of 9 possible actions pairing the speed limits with the groups. Such a setting actually changes the behavior of every group of vehicles by limiting their speeds into a certain range.
- **Rewards**: In reinforcement learning, positive or negative rewards are provided by the environment for each action by the agent. The agent can utilize the reward to calculate the optimal Q-value, and to optimize parameters in the neural network using the discrepancy induced by the Bellman equation. We construct short-term rewards according to the following equation:

$$R = \sum_{v \in \Omega} (e^{-|t_v^{\text{simu}} - t_v|}), \tag{3}$$

where Ω denotes vehicles that have arrived at the monitor points at the simulating step. That is, the simulating results of those vehicles can be evaluated at this time. The arrival time stamps t_v^{simu} and t_v are of the simulated ones and the real vehicles, respectively. The definition is straightforward. Obviously, lower absolute time differences are received if and only if more vehicles arrive sharply, and the agent receives larger rewards under such conditions.
- **Training**: A simulating step is set to lasting for one minute. A two-layer neural network is initialized and trained to estimate the Q-value. Weights θ of the network are updated every step by gradient descent, using the discrepancy between estimated Q-value and the feedback from the simulating environment. Before the training, we first set up a replay memory $\mathcal{M}$ to store state transitions (i.e., (s, a, s', R) where s' denotes the next state following s after taking action a). The deep reinforcement learning algorithm keeps sampling mini-batches from the replay memory and updates its network weights in each simulating step by minimizing the following loss function:

$$\mathcal{L}(\theta) = \mathbf{E}_{s,a}[(Q^* - Q(s, a; \theta))^2], \tag{4}$$

where $Q^* = R + \gamma \max_{a'} Q(s', a'; \theta)$ is the target optimal Q-value, which is the sum of the short-term reward R and the optimal Q-value of the next step. The decay term γ gives penalty to future rewards. The expectation is computed over the mini-batches. Furthermore, we adopt ϵ-greedy strategy [20] to balance the trade-off between exploration and exploitation. That is, with probability ϵ, a random action is selected for exploration during training, instead of the optimal one. After calculating the loss function, the network

Algorithm 1: Training of the Deep RL Architecture

Randomly initialize network parameters θ and the replay memory $\mathcal{M}$;
for *each episode* **do**
  for *each simulation step* **do**
    /* Action and simulating */
    if *random()* $< \epsilon$ **then**
      Select a random action a;
    else
      Select $a = \max_a Q(s, a; \theta)$;
    end
    Obtain short-term reward R and new state s' by executing a in the simulator;
    Store state transition (s, a, s', R) in $\mathcal{M}$;
    /* Updating Q-network parameters θ */
    Sample mini-batch state transitions $\{(s, a, s', R)\}$ from $\mathcal{M}$;
    if *simulator terminated* **then**
      $Q^* = R$;
    else
      $Q^* = R + \gamma \max_{a'} Q(s', a'; \theta)$;
    end
    Use Equation 5 to perform gradient descent on the loss $\mathcal{L}(\theta)$ based on Q^*;
  end
end

weights θ are updated using gradient descent. The gradient of θ is given as follows,

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{s,a}[(Q^* - Q(s, a; \theta))\nabla_\theta Q(s, a; \theta)]. \quad (5)$$

The overall training procedure of the deep reinforcement learning architecture is given in Algorithm 1. Once the simulator is stable, we fill in the missing values of the real and incomplete trajectories using the traffic volume values in the simulator *SUMO*. Compared with taxi trajectories collected from GPS devices, these recovered trajectories, denoted by $\Delta^{\mathrm{I}+}$, are more representative of a variety of vehicle types. Both kinds of trajectories will be employed in the following steps.

4.2 Spatiotemporal Graph Construction

Temporal correlations of different road segments change over time. For example, an expressway connecting business areas and residential areas may exhibit reversed traffic flows from morning to evening. Moreover, the spatial relationship between road segments is extremely complicated (e.g., overpasses at the intersection of the main road). To model such dynamism of traffic in the road network, we construct spatiotemporal graphs, as shown in Figure 3 and detailed below.

A spatiotemporal graph in this paper is a multi-layer graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V} = \{v_i^t\}$ denotes time-enhanced nodes representing road segments $\{r_i\}$, and $\mathcal{E}$ represents edges between nodes. Nodes in the same time interval are grouped to the same *layer*. To preserve spatial restrictions of the road network, edges in $\mathcal{E}$ are limited to $v_i^t \rightarrow v_j^{\{t+1\}}$ where vehicles could travel from the road segment r_i to r_j. The edge's weight is assigned using the traffic flow value, i.e., number of trajectories traversing through the edge.

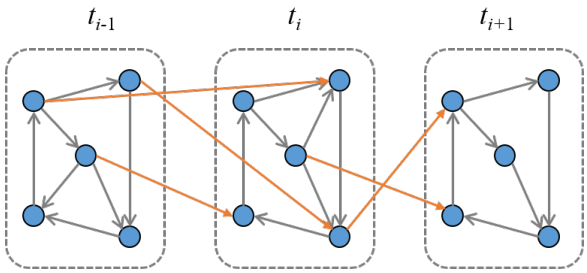

Figure 3: An example of spatiotemporal graph. Three layers are shown in the figure. Gray arrows denote edges within the same layer and yellow arrows denote edges between current and next layers.

To differentiate mobility behaviors and traffic patterns in different sources of trajectories, two spatiotemporal graphs are constructed respectively using the dense trajectories Δ^{D} and recovered $\Delta^{\mathrm{I}+}$ from incomplete trajectories.

4.3 Multi-view Graph Embedding

The objective of citywide traffic volume inference is to estimate the unknown traffic volume values from the observed ones. This problem is challenging due to multiple interwoven factors such as the road network, vehicles' physical condition, road condition, time, etc. Our solution relies on the spatiotemporal graphs constructed from the macro trajectories, effectively factoring out those individual factors and allowing us to conduct the inference on the graphs instead.

We first embed the time-enhanced road segments (nodes of the graph) to real-valued vectors using the two spatiotemporal graphs obtained above. The embedding captures multi-hop correlations [27] between road segments at different times. Moreover, joint embeddings from the two graphs naturally account for both the GPS-tracked taxi and other vehicles monitored in the incomplete trajectories. We first introduce the embedding process on a single graph, followed by the joint embedding algorithm from the two graphs by a multi-view graph embedding method [24].

We adopt the Skip-gram model [17] to learn the embedding vector $\mathbf{u}_i^t$ for each node v_i^t in a spatiotemporal graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$. Given a node in the graph, the skip-gram model aims to predict the probability of the node's context. We first define a path in G as $P_i = v_{i_1} v_{i_2} \dots v_{i_\sigma}$, where v_{i_j} in P_i is corresponding to a node in G. We omit the time superscript here, which can be easily acquired from each layer. Given a specific node v_i^t, the set of all paths containing v_i^t formulates a set $\mathcal{P}(v_i^t)$. We define set $\mathcal{N}(v_i^t) = \{v_c | \exists P_i \in \mathcal{P}(v_i^t), v_c \in P_i\} \backslash \{v_i^t\}$ as the context of node v_i^t, which refers to all multi-hop neighbors of v_i^t (i.e., nodes closely surround v_i^t in space and time) [28]. Specifically, the following log-probability

is maximized:

$$O = \sum_{v_i^t \in G} \log Pr(\mathcal{N}(v_i^t)|v_i^t)) = \sum_{v_i^t \in G} \log \prod_{v_c \in \mathcal{N}(v_i^t)} P(v_c|v_i^t)$$
$$= \sum_{v_i^t \in G} \sum_{v_c \in \mathcal{N}(v_i^t)} \log P(v_c|v_i^t).$$

and $P(v_c|v_i^t)$ is further estimated by $P(v_c|v_i^t) = \sigma({\mathbf{u}_i^t}^{\mathrm{T}} \mathbf{u}_c')$ where $\sigma(\cdot)$ is the sigmoid function, and $\mathbf{u}_c'$ denotes the auxiliary vector of v_c when it is treated as "context" of other nodes.

To achieve better performance and reduce computational costs, negative sampling technique [18] is applied. For each node v_i^t and a neighbor node v_c, we sample negative nodes $NG(v_i^t) = \{z|z \notin \mathcal{N}(v_i^t)\}$ which denotes unrelated road segments. The objective function for network embedding with negative sampling is:

$$O = \sum_{v_i^t \in G} \sum_{v_c \in \mathcal{N}(v_i^t)} \Big\{ \log P(v_c|v_i^t) + \sum_{z \in NG(v_i^t)} \log(1 - P(v_c|v_i^t)) \Big\}$$
$$= \sum_{v_i^t \in G} \sum_{v_c \in \mathcal{N}(v_i^t)} \Big\{ \log(\sigma({\mathbf{u}_i^t}^{\mathrm{T}} \mathbf{u}_c')) + \sum_{z \in NG(v_i^t)} \log(1 - \sigma({\mathbf{u}_i^t}^{\mathrm{T}} \mathbf{u}_z')) \Big\}.$$

The objective reads that, for each node v_i^t, we aim to maximize the probability of its co-occurrence with nodes in $\mathcal{N}(v_i^t)$, while minimize the probability of it being neighbor with nodes from $NG(v_i^t)$.

Since we have two spatiotemporal graphs of the same set of road segments, we may learn the embedding by a trade-off between the objective functions above over the two graphs. Namely,

$$O = \alpha O_{G^{\mathrm{D}}} + (1 - \alpha) O_{G^{\mathrm{I+}}}, \tag{6}$$

where G^{D} and $G^{\mathrm{I+}}$ are the graphs constructed from dense and recovered trajectories, respectively, and α balances the two terms.

4.4 Citywide Traffic Volume Inference

After learning representations for road segments that preserve spatiotemporal dynamics, we borrow a semi-supervised learning formulation [46] to propagate the traffic volume values to the unknown road segments. At first glance, one may train a regression model taking as input the road segment's embedding to predict the missing traffic volume data. However, such methods fail when only a small portion of the entire road network is monitored. We instead employ a formulation which is popular in semi-supervised learning for the inference.

We first define pairwise similarities between any two road segments. The similarity of r_i at time interval t and r_j at time interval t' is measured by:

$$sim(r_i^t, r_j^{t'}) := \omega_{i,j}^{t,t'} = \langle \mathbf{u}_i^t \cdot \mathbf{u}_j^{t'} \rangle, \tag{7}$$

where $\langle \cdot \rangle$ denotes the inner product of two vectors.

We then prune out irrelevant pairs of road segments by constructing 0-1 masks based on geospatial features (i.e., absolute distance between road segments) as well as temporal distance. The intuition behind such pruning is that traffic volume values of faraway locations usually do not have similarities, and the temporal dependencies of traffic pattern at the same location will decrease as the time lag becomes longer [38]. The "masked" pairwise similarities between road segments are defined as:

$$\omega^{\star\, t,t'}_{i,j} = \begin{cases} \omega_{i,j}^{t,t'}, & \text{if } adj(r_i, r_j) \wedge |t - t'| \leq 1, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Here $adj(r_i, r_j)$ is *true* if and only if r_i and r_j are adjacent on the road network. The above mask ensures only the correlations between road segments that are close enough in time and space are considered. The citywide traffic volume values $\{x_{it}\}$ are inferred by minimize the following with respect to (w.r.t.) the unknown values:

$$\mathcal{L}^\star = \sum_{1 \leq i,j \leq m, 1 \leq t,t' \leq n} \omega^{\star\, t,t'}_{i,j} (x_{it} - x_{jt'})^2. \tag{9}$$

Since the objective function is convex w.r.t. the unknown traffic volume values, gradient descent is applied to find the global solutions.

5 EXPERIMENTS

In this section, we conduct extensive experiments to answer the following research questions.

- How does the proposed JMDI perform compared with the baseline methods?
- How do each component of JMDI affect the overall performance?
- How accurate is the trajectory recovery via the deep RL architecture?
- How to balance different types of trajectories through the hyper-parameter α?
- How is the inference error distributed over space (i.e., different road segments) and time (i.e., different time periods of a day)?

5.1 Datasets

We compile a traffic volume dataset using the vehicle counting reported by the city surveillance system from a large provincial capital city in China. The incomplete trajectories are acquired using plate numbers identified by surveillance cameras (with an average F1-score of 90%). Dense trajectories are collected from GPS sensors equipped by public taxicabs. The average reporting frequency of the taxicab-carried GPS devices is 5 seconds. All those datasets are collected over the period of Aug. 2016. The road network is extracted from *Openstreetmap.org* [21]. We split roads into segments using intersections as natural cut-points and set the length of the time interval to 5 minutes. Moreover, geospatial features such as starting/ending locations, road segment length, road width, number of lanes, road type, speed limit, etc., are constructed for each road segment. We select two regions in the city. The detailed data descriptions are shown in Table 2.

5.2 Experiment Settings

We cut trajectories recursively at any $p_{i,j} \rightarrow p_{i,j+1}$ if the time interval between $p_{i,j}$ and $p_{i,j+1}$ is larger than 30 minutes. Because vehicles may stop and park during such long duration, their behaviors are different from the constantly moving vehicles.

We randomly split the road segments with traffic information into training (80%) and testing (20%), respectively. We further select 20% of the training randomly as validation. The same split is used

Table 2: Dataset statistics.

	Region 1	Region 2
Time period	Aug. 1 - Aug 4	Aug. 14 - Aug. 20
Time interval	5 minutes	5 minute
Longitude range	116.8988 - 116.9457	117.0635 - 117.1053
Latitude range	36.6507 - 36.6926	36.6624 - 36.7052
# of all road seg.	322	143
# of monitored seg.	45	32
# of incomplete traj.	366,645	1,798,782
# of dense taxi traj.	12,562	97,315

for all experiments. We run each baseline and ablation 10 times to report the average results.

We set the memory buffer size to $10,000$ and the discount factor γ to 0.8 for the deep RL framework. A feed-forward neural network with two 256-dimension hidden layers is trained using the Adam optimizer [10]. The size of a mini-batch is 128 and the learning rate is 0.0001. Parameters of the network are updated at every step of the simulator. The probability ϵ in the deep RL training is reduced linearly from 0.5 to 0.01 over $2,000$ epochs. The speed limits for every type of vehicle are restricted between $1m/s$ and $40m/s$ in the simulating environment. We set the dimension size for embedding vectors to 50. The window size for sampling in Skip-gram is 10.

5.3 Metrics

We use Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) to evaluate the performance of all methods.The detailed definitions of the two metrics are stated as below:

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

where $x_l \in \{x_{it}\}$ is a test sample, $\hat{x}_l$ is the ground truth of x_l, and s denotes the total number of test samples.

While RMSE receives more penalties from larger values, MAPE focuses on the error of smaller values. Therefore, combining these two metrics can lead to more comprehensive conclusions. Similar to [37, 38], to evaluate relative error using MAPE, we filter out samples whose traffic volume is lower than 5 in testing, since road segments with very low traffic volume are of little interest for our problem.

5.4 Baselines

Our solution is compared with the following baselines: (1) two geospatial-based methods, (2) two supervised learning methods, and (3) two semi-supervised learning methods.

- **Spatial *k*NN** selects top k nearest road segments with traffic data and uses the average of their volume values at each time interval as the traffic volume prediction for unmonitored road segments.
- **Contextual Average** (CA) uses the averaged volume values from same-type road segments (e.g., primary, expressway, etc.) as the prediction.
- **Linear Regression** (LR) is trained on all road segments with traffic volume using geospatial features. We train regression models for each time interval.
- **XGBoost** [5] is a boosting-tree-based method which is popular in data mining community. We train XGBoost separately on each time interval.
- **Graph SSL** Classic graph-based semi-supervised learning method with loss function $\mathcal{L} = \sum_t \sum_{i,j} a_{i,j}(x_{it} - x_{jt})^2$, which is implemented following Zhu et al.. The similarity weight $a_{i,j}$ is defined using the Euclidean distance between the geospatial feature vectors r_i and r_j.
- **ST-SSL** [16] is the state-of-the-art method for inferring citywide traffic volume using dense trajectories. ST-SSL utilizes static features and estimated speed patterns from taxi trajectories to model the correlations between road segments. The citywide traffic volume is then inferred by a semi-supervised learning method.

5.5 Result Analysis of Different Methods

The comparison of JMDI with other methods in two regions is shown in Table 3. As we can see, JMDI achieves best performances in both regions. In particular, JMDI significantly outperforms geospatial-based methods due to its capability of well modeling the dynamism of traffic volume. In addition, JMDI achieves tremendous improvement compared with supervised methods. This is because supervised methods are trained on road segments with traffic data, they ignore correlations between other road segments. Moreover, both geospatial and regression-based methods fail to model spatiotemporal patterns in multi-sources trajectories, and do not well utilize information from unmonitored road segments.

JMDI significantly outperforms both Graph SSL and ST-SSL. There are two potential reasons. First, JMDI utilizes vast incomplete trajectory data to recover spatiotemporal similarities among road segments. The proposed deep reinforcement learning recovery method successfully transforms partial observations to complete route samples. Second, the joint embedding algorithm enhances the representation of road segments. Both multi-hop correlations and multi-view graph information are preserved in learned embedding vector, resulting in better performance. Furthermore, Graph SSL achieves lower errors compared with ST-SSL, a possible explanation is that the strong assumption of speed similarity does not hold in those regions. For example, most drivers will drive close to the speed limit unless the road capacity reaches the upper limit. In such condition, speeds are almost stationary while traffic volume could change depending on the number of vehicles. Moreover, it suffers from the spatial sparsity of taxi trajectories since the coverage of taxi trajectory in the city is limited.

5.6 Ablation Study

To validate each component of JMDI, the following variations of JMDI are further studied. Despite the changed part(s), all variations have the same framework structure and parameter settings. The results of all variations are listed in Table 4.

- **JMDI-Reg** trains XGBoost models using representation of road segments. It can be considered as appending embedding vectors to the geospatial features in previous XGBoost.

Table 3: Comparison of baseline methods.

Methods	Region 1		Region 2	
	RMSE	MAPE	RMSE	MAPE
Spatial kNN	7.1404	0.6348	8.5259	0.6053
CA	8.0866	0.5251	8.1317	0.5423
LR	13.2857	1.0911	11.5781	1.0219
XGBoost	9.2157	0.6711	9.2847	0.6954
Graph SSL	6.1743	0.4457	6.3640	0.4643
ST-SSL	6.5603	0.5631	7.1528	0.5975
JMDI (Ours)	**5.5877**	**0.3675**	**5.6680**	**0.3915**

- **JMDI-Tr** replaces the trajectory recovery part using algorithm in [4]. The method infers uncertain trajectories from network-constrained partial observations by summarizing all probable routes in a holistic manner.
- **JMDI-Df** recovers trajectories using default simulator, i.e., we directly collect results from *SUMO* without training the deep reinforcement learning component.
- **JMDI-Semi** directly applies graph-based semi-supervised learning [46] on the spatiotemporal graphs to propagate known traffic volume without the embedding process. The similarity between any time-enhanced road segments (nodes of the graph) is assigned according to the edge weights (i.e., traffic flow value from trajectory data).
- **JMDI-Um** proceeds semi-supervised learning without pruning out irrelevant pairs of road segments. Namely, the raw similarity in Equation (7) is used.

JMDI-Reg performs the worst among all variations. However, it still outperforms the original XGBoost. In addition, the performance gap between JMDI-Reg and JMDI reflects the advantage of the semi-supervised method, which successfully leverages large amounts of unmonitored road segments.

The comparison between JMDI-Tr and JMDI highlights the effectiveness of the proposed trajectory recovery method. Sampling-based methods are unable to handle complex road network constraints, vehicle interactions, and other factors, thus fail to recovery highly accurate trajectories. Moreover, JMDI-Df performs worse compared with JMDI. A possible explanation is that default stationery (time independent) parameters do not describe real-world vehicle movements correctly. Furthermore, since there are fewer vehicles in the simulator, some parameters should still be altered even if the ground truth is known. For example, vehicles traverse faster and arrive earlier in a simulated environment, resulting in a higher error of trajectory recovery. In such condition, speed limits could be explicitly reduced to adjust vehicles' movement.

The performance of JMDI-Semi is obviously overtaken by JMDI. This is because the joint embedding enhances the representation capacity from multi-hop spatiotemporal correlations and multi-view graph information.

The comparison of JMDI-Um and JMDI indicates the effectiveness of proposed 0-1 masks that explicitly remove unrelated road segment pairs. The propagation of traffic volume focuses on relevant pairs of road segments, improving the overall performance of JMDI distinctly.

Table 4: Comparison of variations.

Methods	Region 1		Region 2	
	RMSE	MAPE	RMSE	MAPE
JMDI-Reg	7.0851	0.5182	6.9187	0.5155
JMDI-Tr	6.5091	0.5127	6.8262	0.5390
JMDI-Df	6.1445	0.4104	6.4006	0.4378
JMDI-Semi	7.9121	0.5673	7.5165	0.5366
JMDI-Um	6.0357	0.3781	6.2272	0.4057
JMDI	**5.5877**	**0.3675**	**5.6680**	**0.3915**

Table 5: Averaged arrival time error (s) of simulated vehicles.

Region	Region 1	Region 2
Default Parameters	27.2356	29.2141
Deep RL	**22.2554**	**25.2231**

5.7 Accuracy of Trajectory Recovery

We evaluate the trajectory recovery algorithm. Since the spatial ground truth for incomplete trajectories is unavailable, we construct validation from arrival times at monitor points. First, we manually tune speed limit parameters using greedy search approach over each hour of the simulating procedure to minimize the arrival time error. Those speed parameters are used as the default setting of the compared method. Note that we do not use the default speed limit parameters in *SUMO* (i.e., $22.4m/s$). Following Equation (1), the averaged arrival time error (in second) of the default simulator and that of the optimized simulator are compared. The results are demonstrated in Table 5. The proposed deep reinforcement learning method reduces averaged arrival error significantly, which is relatively lower by 18.29% in Region 1 and 13.61% in Region 2. Moreover, the improvement in Region 1 is relatively higher than in Region 2. A possible reason is that Region 1 has a relatively simpler environment with less traffic. Therefore, modeling the state-action relationship could be easier for the DQN.

5.8 Hyper-Parameter

We further study the hyperparameter α which balances weights between different spatiotemporal graphs. More emphasis is put to recovered trajectories when α is closer to 0, and vise versa when α is closer to 1. The performance of JMDI w.r.t different α is shown in Figure 4. As we can observe, the inference error first drops and then rises in both regions. Moreover, JMDI achieves relatively low inference errors when α falls into a certain range. However, the performance is worse when α is closer to 0 or 1. In addition, the performance of JMDI is significantly improved in both regions by leveraging incomplete trajectories. Compared with the ablation that only uses taxi data (i.e., $\alpha = 1$), JMDI reduces the RMSE value by 8.84%, 5.47%, and the MAPE value by 9.76%, 7.41% in region 1 and region 2, respectively. This suggests the importance of modeling multi-source trajectories jointly. Besides, the relative improvement from recovered trajectories in Region 1 is larger compared with Region 2. A potential explanation is that the taxi trajectories in Region 1 are very sparse, thus brings biased spatiotemporal correlation and results in larger inference error. Moreover, recovered

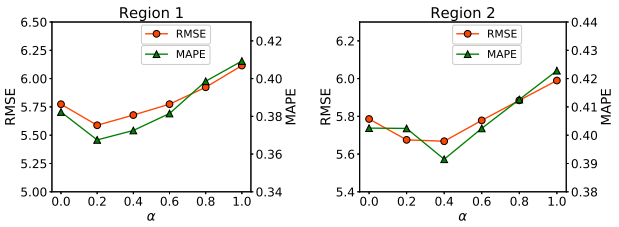

Figure 4: Performance w.r.t different α.

Table 6: Error distribution over different groups of road segments.

Group	Region 1		Region 2	
	RMSE	MAPE	RMSE	MAPE
Group 1	6.5151	0.4102	6.6741	0.4170
Group 2	3.5488	0.3368	3.4101	0.3274

trajectories also contain noises and uncertainties, and that's why JMDI achieves the best performance by combining them together.

5.9 Error Distribution Interpretation

We also explore the spatiotemporal performance of JMDI, i.e., error distribution over different types of roads, and over different time periods of a day. Without loss of generality, Region 2 is selected for the interpretation.

5.9.1 Error distribution over the space. To study the performance of JMDI over different road segments, we first split road segments from testing set into two categories using road type information extracted from the map. Namely, Group 1 (i.e., major roads such as highway and primary way) and Group 2 (secondary roads like tertiary route). Generally, road segments from Group 1 have more lanes and higher average traffic volume values.

Table 6 provides a detailed illustration of the spatial performance of JMDI. As we can see, the RMSE values are higher for road segments from Group 1, as they have higher traffic volume values. RMSE values are positively correlated to the original variables themselves. Meanwhile, we also observe that the MAPE values of road segments from Group 2 is relatively lower than those of road segments from Group 1. One potential reason is that traffic volume of road segments from Group 1 has higher uncertainty. For example, traffic jams occur when the traffic volume exceeds road capacity, and this usually happens on major roads. Moreover, local events have a higher probability to happen near major roads. Those factors increase the uncertainty of traffic volume on road segments from Group 1, resulting in a high relative error (i.e., MAPE) of the framework.

5.9.2 Error distribution over different time periods of the day. Since traffic volume value series are closely related to different times of a day, and peak hours are more interesting, we further explore the performance of JMDI at different hours. The averaged RMSE and MAPE values are calculated over each hour of a day in the testing set. Detailed values are illustrated in Figure 5. To

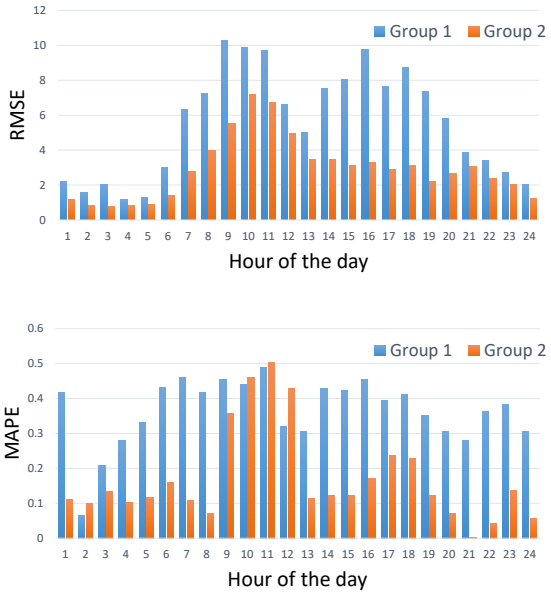

Figure 5: Error distribution over different hours of a day. Number i on x-axis indicates the i-th hour of a day. For example, 7 means the 7th hour, i.e., 6:00 to 7:00.

better interpret those errors, we still use the same group category in Section 5.9.1.

As it reads from the figure, the RMSE values for both groups of road segments have clear patterns. The values are higher during the busy time, such as rush hours in the morning and in the afternoon. This is reasonable since common practice tells us that peak hours would have more traffic. And RMSE values are positively correlated to the ground truth values.

On the other hand, the relative error of JMDI (i.e., the MAPE values) are pretty stable for Group 1 road segments, where the values of MAPE in rush hours are slightly higher than other ones. This is again due to traffic patterns during rush hours are more uncertain, and more accidental events could change the traffic volume dramatically. Those are reasons why inference difficulty is increased, leading to higher percentage error. At the same time, JMDI achieve very low MAPE values for Group 2 road segments at around 0.1 during off-peak hours. And its MAPE values during rush hours in the morning is reasonable. This is explainable since accidental events or traffic jams are less likely to happen near roads from Group 2, because of their less traffic volume.

6 CONCLUSION

In this paper, we propose a novel framework JMDI to citywide traffic volume inference using dense and incomplete trajectories. JMDI combines high-fidelity traffic simulator with deep reinforcement learning to recover full vehicle movements from incomplete trajectories. The masked semi-supervised approach enhanced by multi-view graph embedding is introduced for citywide traffic volume inference. We evaluate JMDI in two regions in a provincial capital city in China to demonstrate its effectiveness.

REFERENCES

[1] Andreas Argyriou, Mark Herbster, and Massimiliano Pontil. 2006. Combining graph Laplacians for semi–supervised learning. In *Advances in Neural Information Processing Systems*. 67–74.

[2] Muhammad Tayyab Asif, Nikola Mitrovic, Justin Dauwels, and Patrick Jaillet. 2016. Matrix and tensor based methods for missing data estimation in large traffic networks. *IEEE Transactions on Intelligent Transportation Systems* 17, 7 (2016), 1816–1825.

[3] Javed Aslam, Sejoon Lim, Xinghao Pan, and Daniela Rus. 2012. City-scale traffic estimation from a roving sensor network. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM, 141–154.

[4] Prithu Banerjee, Sayan Ranu, and Sriram Raghavan. 2014. Inferring uncertain trajectories from partial observations. In *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 30–39.

[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.

[6] Mark Culp and George Michailidis. 2008. Graph-based semisupervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 1 (2008), 174–179.

[7] Tran Duc Duan, TL Hong Du, Tran Vinh Phuoc, and Nguyen Viet Hoang. 2005. Building an automatic vehicle license plate recognition system. In *Proc. Int. Conf. Comput. Sci. RIVF*. Citeseer, 59–63.

[8] Astrid Gühnemann, Ralf-Peter Schäfer, Kai-Uwe Thiessenhusen, and Peter Wagner. 2004. Monitoring traffic and emissions by floating car data. (2004).

[9] Huiqi Hu, Guoliang Li, Zhifeng Bao, Yan Cui, and Jianhua Feng. 2016. Crowdsourcing-based real-time urban traffic speed estimation: From trends to speeds. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE, 883–894.

[10] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[11] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. 2012. Recent Development and Applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements* 5, 3&4 (December 2012), 128–138.

[12] Jaimyoung Kwon, Pravin Varaiya, and Alexander Skabardonis. 2003. Estimation of truck traffic volume from single loop detectors with lane-to-lane speed correlation. *Transportation Research Record: Journal of the Transportation Research Board* 1856 (2003), 106–117.

[13] Li Li, Yuebiao Li, and Zhiheng Li. 2013. Efficient missing data imputing for traffic flow by considering temporal and spatial dependence. *Transportation research part C: emerging technologies* 34 (2013), 108–120.

[14] Mu Li, Amr Ahmed, and Alexander J Smola. 2015. Inferring movement trajectories from GPS snippets. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 325–334.

[15] Weizi Li, Dong Nie, David Wilkie, and Ming C Lin. 2017. Citywide estimation of traffic dynamics via sparse gps traces. *IEEE Intelligent Transportation Systems Magazine* 9, 3 (2017), 100–113.

[16] Chuishi Meng, Xiuwen Yi, Lu Su, Jing Gao, and Yu Zheng. 2017. City-wide Traffic Volume Inference with Loop Detector Data and Taxi Trajectories. In *Proceedings of ACM International Conference on Advances in Geographical Information Systems*.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[21] OpenStreetMap contributors. 2017. OpenStreetMap. https://www.openstreetmap.org. (2017).

[22] Li Qu, Li Li, Yi Zhang, and Jianming Hu. 2009. PPCA-based missing data imputation for traffic flow volume: A systematical approach. *IEEE Transactions on intelligent transportation systems* 10, 3 (2009), 512–522.

[23] Li Qu, Yi Zhang, Jianming Hu, Liyan Jia, and Li Li. 2008. A BPCA based missing value imputing method for traffic flow volume data. In *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 985–990.

[24] Meng Qu, Jian Tang, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. 2017. An Attention-based Collaboration Framework for Multi-View Network Representation Learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1767–1776.

[25] Wenjie Ruan, Peipei Xu, Quan Z Sheng, Nickolas JG Falkner, Xue Li, and Wei Emma Zhang. 2017. 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, 607–622.

[26] Zhenyu Shan, Danna Zhao, and Yingjie Xia. 2013. 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, 118–123.

[27] Hongjian Wang and Zhenhui Li. 2017. Region Representation Learning via Mobility Flow. 10 (2017). https://doi.org/10.1145/3132847.3133006

[28] 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 Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 19.

[29] Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. MathDQN: Solving Arithmetic Word Problems via Deep Reinforcement Learning. (2018).

[30] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. 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, 25–34.

[31] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.

[32] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. 2018. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2496–2505.

[33] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. 2012. Constructing popular routes from uncertain trajectories. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 195–203.

[34] David Wilkie, Jason Sewall, and Ming Lin. 2013. Flow reconstruction for data-driven traffic animation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 89.

[35] Yuto Yamaguchi, Christos Faloutsos, and Hiroyuki Kitagawa. 2015. OMNI-Prop: Seamless Node Classification on Arbitrary Label Correlation.. In *AAAI*. 3122–3128.

[36] Ning Yang and Philip S Yu. 2016. Efficient Hidden Trajectory Reconstruction from Sparse Data. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 821–830.

[37] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. 2019. Revisiting Spatial-Temporal Similarity: A Deep Learning Framework for Traffic Prediction. *2019 AAAI Conference on Artificial Intelligence (AAAI'19)*.

[38] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* (2018).

[39] Xiuwen Yi, Yu Zheng, Junbo Zhang, and Tianrui Li. 2016. 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, 2704–2710.

[40] Xianyuan Zhan, Ruimin Li, and Satish V Ukkusuri. 2015. Lane-based real-time queue length estimation using license plate recognition data. *Transportation Research Part C: Emerging Technologies* 57 (2015), 85–102.

[41] Xianyuan Zhan, Yu Zheng, Xiuwen Yi, and Satish V Ukkusuri. 2017. Citywide Traffic Volume Estimation Using Trajectory Data. *IEEE Transactions on Knowledge and Data Engineering* 29, 2 (2017), 272–285.

[42] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 167–176.

[43] Jiangchuan Zheng and Lionel M Ni. 2013. Time-dependent trajectory regression on road networks via multi-task learning. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press, 1048–1055.

[44] Kai Zheng, Yu Zheng, Xing Xie, and Xiaofang Zhou. 2012. Reducing uncertainty of low-sampling-rate trajectories. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 1144–1155.

[45] Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in neural information processing systems*. 321–328.

[46] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 912–919.