Spatial Transition Learning on Road Networks with Deep Probabilistic Models

Xiucheng Li[†], Gao Cong[†], Yun Cheng[‡]
[†] *Nanyang Technological Univeristy,* [‡] *ETH Zurich*
{xli055@e., gaocong@}ntu.edu.sg, chengyu@ethz.ch

***Abstract*—In this paper, we study the problem of predicting the most likely traveling route on the road network between two given locations by considering the real-time traffic. We present a deep probabilistic model–DeepST–which unifies three key explanatory factors, the past traveled route, the impact of destination and real-time traffic for the route decision. DeepST explains the generation of next route by conditioning on the representations of the three explanatory factors. To enable effectively sharing the statistical strength, we propose to learn representations of K-destination proxies with an adjoint generative model. To incorporate the impact of real-time traffic, we introduce a high dimensional latent variable as its representation whose posterior distribution can then be inferred from observations. An efficient inference method is developed within the Variational Auto-Encoders framework to scale DeepST to large-scale datasets. We conduct experiments on two real-world large-scale trajectory datasets to demonstrate the superiority of DeepST over the existing methods on two tasks: the most likely route prediction and route recovery from sparse trajectories. In particular, on one public large-scale trajectory dataset, DeepST surpasses the best competing method by almost 50% on the most likely route prediction task and up to 15% on the route recovery task in terms of accuracy.**

I. Introduction

In this paper, we study the problem, given the origin and destination, of predicting the most likely traveling route on a road network as well as outputing a probability value to indicate the likelihood of a route that being traveled. The problem finds applications in a variety of downstream tasks, such as taxi ridesharing schedule, route recovery from sparse trajectories [1], [2], and popular routes recommendation [3].

As a motivating example, in taxi dispatch system the origin and destination are usually given before the start of a trip, and thus predicting the most likely route could help us better arrange the taxi sharing by picking up the potential passengers that are waiting on or nearby the most likely traveled route. Furthermore, if we could score the likelihood of a route being traveled, we can also recover the underlying route from sparse trajectories. The route recovery problem [1], [2] arises in the real-world trajectories due to the low-sampling-rates or turning off of location-acquisition devices. Figure 1b shows two possible routes $\mathbf{r}_A$, $\mathbf{r}_B$ from a_3 to a_4 in an observed trajectory $T_a = [a_1, a_2, \dots, a_5]$. If we treat a_3, a_4 as the origin and destination respectively, we could score the likelihood of the two candidate routes to help infer the truly traveled route.

The problem addressed in this paper can be summarized as modeling the spatial transition patterns of real-world trips on the road network. Several proposals [4], [5] have revealed

TABLE I
Example of trips on road network.

Route	Destination	Frequency
$r_3 \to r_2 \to r_4$	C	400
$r_3 \to r_2 \to r_5 \to r_{10}$	C	100
$r_1 \to r_2 \to r_5 \to r_6$	A	100
$r_1 \to r_2 \to r_7 \to r_8$	B	100
$r_1 \to r_2 \to r_5 \to r_6 \to r_9$	B	100

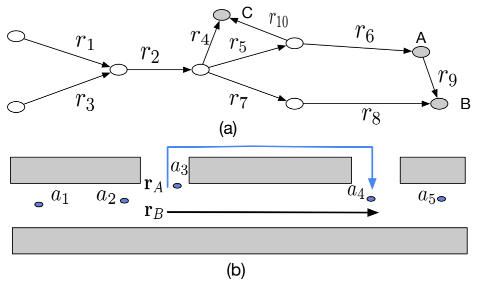


Fig. 1. (a) Road network. (b) Example of route recovery from sparse trajectory, in which the grey bars indicate building blocks and $\mathbf{r}_A$, $\mathbf{r}_B$ represent two possible routes.

that the transition patterns of vehicles are often highly skewed: some routes are more likely to be traveled than others. However, to reliably model such transition patterns, we argue that three key explanatory factors need to be carefully taken into consideration. We illustrate these key factors with an example; assuming we have observed 8 trips described in Table I, over a road network showed in Figure 1a.

First, the spatial transition patterns demonstrate strong sequential property. Consider that we try to predict the next transition of a vehicle driving on r_2. According to the historical trips, $\mathbb{P}(r_4|r_2) = 4/8$ is greater than $\mathbb{P}(r_5|r_2) = 3/8$ and $\mathbb{P}(r_7|r_2) = 1/8$. We have a high confidence to predict that the vehicle will transit to r_4. However, if we also know that the traveled road sequence is $r_1 \to r_2$, the prediction will then favor r_5 over r_4. Second, the trip destination has a global impact on the transition. Now consider that a vehicle is driving on r_5 and the trip destination is C. Based on the historical trips it will have a higher probability of transiting to r_6 than transiting to r_{10} since $\mathbb{P}(r_6|r_5) = 2/3 > \mathbb{P}(r_{10}|r_5) = 1/3$. But if we take into consideration the trip destination, we would predict the next road to be r_{10} as $\mathbb{P}(r_{10}|r_5, C) = 1$. Third, the route choices are also influenced by the real-time traffic.

The vehicle drivers tend to choose those less congested routes rather than the shortest one. Again, assuming that a vehicle is on r_2 and the trip destination is B. The historical trips show that it has equal probability of transiting to r_5 or r_7 as $\mathbb{P}(r_5|r_2, B) = \mathbb{P}(r_7|r_2, B) = 1/2$. However, if the traffic on r_7, r_8 is more congested than that on r_5, r_6, r_9, the driver is more likely to choose r_5 instead of r_7 even if the route $r_7 \rightarrow r_8$ is shorter.

It is challenging to unify all these factors into a single model. To capture the sequential property, the authors [5] propose to model the spatial transition patterns using the Hidden Markov Model (HMM) which requires explicit dependency assumptions to make inference tractable [6]. Wu *et al.* [7] consider the impact of destination on the route decision by treating each destination separately, thus failing to share statistical strength across trips with nearby destinations. In addition, they assume that accurate ending streets of the trips are available, and use the corresponding road segments to help make decision. However, in some applications we may only have the rough destination coordinates at hand *e.g.,* the driver may not end a trip on the exact street as the user requested in the taxi dispatch system. To incorporate the influence of traffic when learning the transition patterns, [2], [8] assume that traffic conditions in the same time slot (*e.g.,* 7:00am-8:00am every weekday) are temporally-invariant, which is not real-time traffic and may not hold. To our knowledge, no existing work on spatial transition learning is based on real-time traffic.

In this paper, we approach the spatial transition modeling problem from the generative perspective. We develop a novel deep probabilistic model–DeepST (Deep Probabilistic Spatial Transition), which unifies the aforementioned three key explanatory factors–sequential property, the impact of destination and real-time traffic into a single model, for learning the spatial transition patterns on the road network. DeepST explains the generation of a route by conditioning on the past traveled road sequence, destination and real-time traffic representations. The past traveled road sequence is squeezed by a Recurrent Neural Net (RNN) which does not make the explicit dependency assumption. To incorporate the impact of destination, we propose to learn K-destination proxies with an adjoint generative model instead of treating the destinations separately. The benefits of introducing the destination proxies are two-folds: 1) the proposed method will be robust to the inaccuracy of destinations; 2) it allows effectively sharing the statistical strength across trips. The destination proxies are learned jointly with the generation of routes which permits end-to-end training. To account for the influence of real-time traffic, we propose to learn the real-time traffic representation with a latent variable, whose posterior distribution can then be inferred from the observations. In the end, we develop an efficient inference method to learn the posterior distributions of the latent variables of DeepST based on observations; the inference method is developed within the Variational Auto-Encoders (VAEs) framework and is fully differentiable, enabling DeepST to scale to large-scale datasets. In summary, our contributions are as follows.

- For the first time, we develop a novel deep probabilistic model–DeepST– to learn the spatial transition patterns, which simultaneously takes into consideration the transition sequential property, the impact of destinations and real-time traffic.
- We propose a novel adjoint generative model to learn the K-destination proxies, which enables effectively sharing statistical strength across trips and the resulting model is robust to inaccurate destinations.
- We develop an efficient inference method that scales the model to large-scale datasets within the VAEs framework.
- We conduct experiments on two real-world datasets to demonstrate the superiority of DeepST over existing methods on two tasks: predicting the most likely routes and route recovery from sparse trajectories.

II. Related Work

We briefly review the related work on spatial transition modeling on the road network and deep probabilistic models.

A. Spatial transition modeling on road network

The spatial transition modeling on the road network in the literature arises in the following area: sparse trajectory similarity computation, route recovery from sparse trajectories, and future movement prediction.

Sparse trajectory similarity computation. The low-sampling-rated trajectories bring difficulty for the similarity computation, as an observed sparse trajectory could have multiple possible underlying routes. To handle this, Su *et al.* [5] proposed to learn the transition patterns among a fixed set of spatial objects from the historical trajectories by using the Hidden Markov Models; then sparse trajectories are calibrated to these spatial objects to compute similarities. Li *et al.* [9] addressed this problem with a seq2seq-based model, where they encoded the most probable route information into the trajectory representation.

Route recovery from sparse trajectories. Route recovery attempts to infer the most likely route between two road segments that are not adjacent on the road network. Zheng *et al.* [4] modeled the spatial transition probability between adjacent road segments with one-order Markov model. Banerjee *et al.* [1] explored the problem with Gibbs sampling, in which the spatial transition probabilities were also modeled with the Markov model albeit high-order ones were employed. Wu *et al.* [2] proposed to use the inverse reinforcement learning to capture the spatial transition patterns. The problem we consider in this paper actually is more general than route recovery from sparse observations, in the sense that DeepST can be applied for route recovery while the vice verse may not hold, for two reasons: first, the route recovery attempts to infer the most likely routes for already observed trajectories, in which the travel time is available to help the inference, while in our problem setting we do not require the travel time to be known in advance; second, the route recovery problem assumes the accurate destination road segments are known, whereas DeepST is applicable even if only the rough destination coordinates are available.

Future movement prediction. Xue *et al.* [10] attempted to predict the destination of a trajectory based on its already observed partial sub-trajectory. They partitioned the space into cells and modeled the transition probability between adjacent cells with the first-order Markov Model. Zhao *et al.* [11] revisited the problem with RNN. Li *et al.* [8] developed a Bayesian model which is capable of predicting the future movement of a vehicle on the road network. The spatial transition was again modeled with the first-order Markov Model. As the Markov Model requires explicit dependency assumptions and struggles in accounting for the long range dependency, Wu *et al.* [7] explored to model the trajectories with RNN. They assumed that the exact destination road segments of trips are known and learned the representations of them to help route decision. In contrast, we only assume the rough destination addresses to be available and handle them with a novel adjoint generative model. Several proposals [12]–[14] consider the problem of the next location prediction, without the constraint of road networks, and thus they are not directly applicable to our scenario.

In addition, the existing methods either ignore the influence of traffic [1], [4], [5], [7], [10] or simply assume that the traffic conditions in the same periodic time slot (*e.g.,* 7-8am weekend) are temporally-invariant [2], [8]. To our knowledge, no existing work on spatial transition modeling is based on the real-time traffic.

B. Deep probabilistic models

The probabilistic models have been widely studied in the machine learning and data mining fields including, text analysis [15], [16], recommendation [17]–[19], and spatial data analytics [20]–[23]. As the probabilistic models were typically trained with sampling-based methods that suffer from slow convergence, which has limited their usage to relatively small datasets [24].

The progress of stochastic gradient variational inference [25], [26] which marries the Bayesian inference with deep learning, has successfully scaled probabilistic models to large-scale data sets. Such models are often referred to as deep probabilistic or generative models, and the key idea is to fit the posterior distributions with the neural nets by optimizing the Evidence Lower Bound (ELBO). Ever since, deep probabilistic models have been developing rapidly and achieved a wide range of state-of-art results in semi-supervised learning [27], image generation [28], scene understanding [29], realistic speech synthesis [30], and large-scale online recommendation [31]. However, they have been little explored in spatial-temporal data analytics. To our knowledge, [32], [33] are the only two proposals that attempt to explore the power of deep generative models for the spatial-temporal data mining problems. Li *et al.* [32] develop a deep generative model–DeepGTT, which aims to predict the travel time distribution of a given trip whereas DeepST seeks to predict the most likely route between the given origin and destination, they both rely on real-time traffic but have complete different generative processes and purposes. Liu *et al.* [33] propose a deep generative model GM-VSAE to detect the anomaly trajectories in an online manner. In this paper, we step towards this direction by presenting the first deep probabilistic model for spatial transition modeling.

TABLE II
NOTATION.

Symbol	Definition
T	Trajectory
$\mathbf{T}$	Trip
$\mathbf{r}$	Route
r_i	The i-th road segment in route $\mathbf{r}$
$\mathbf{x}$	Destination coordinate
$\boldsymbol{\pi}$	Destination allocation indicator
$\mathbf{c} \in \mathbb{R}^{\lvert\mathbf{c}\rvert}$	The real-time traffic representation
C	Real-time traffic indicator

III. Definitions and Preliminaries

We present the definitions and problem statement in Section III-A. The key ideas of Variational Inference (VI) and Variational Auto-Encoders (VAEs) are briefly reviewed in Section III-B.

A. Definitions and problem statement

For the convenience of reference, we list the notations used in the paper in Table II.

Definition 1. ***Road Network.*** *A road network is represented as a directed graph $G(V, E)$, in which V, E represent the vertices (crossroads) and edges (road segments) respectively.*

Definition 2. ***Route.*** *A route $\mathbf{r} = [r_i]_{i=1}^n$ is a sequence of adjacent road segments, where $r_i \in E$ represents the i-th road segment in the route.*

Definition 3. ***GPS Trajectory.*** *A GPS trajectory T is a sequence of sample points $\langle p_i, \tau_i \rangle_{i=1}^{|T|}$ from the underlying route of a moving object, where p_i, τ_i represent the i-th GPS location and timestamp, respectively.*

Definition 4. ***Trip.*** *A trip $\mathbf{T}$ is a travel along a route $\mathbf{r}$ on the road network starting at time s. We use $\mathbf{T}.\mathbf{r}$ and $\mathbf{T}.s$, respectively, to denote the traveled route and starting time of trip $\mathbf{T}$.*

Problem Statement. Given a road network $G(V, E)$ and a historical trajectory dataset $\mathcal{D} = \{T^m\}_{m=1}^M$, as well as the starting time, origin and destination of a trip $\mathbf{T}$, we aim to predict the most likely traveled route of the trip $\mathbf{T}$ as well as score the likelihood of any route being traveled by conditioning on the real-time traffic.

In this paper, we assume that the initial road segment $\mathbf{T}.r_1$ of the trip $\mathbf{T}$ is given; however, for the destination we only assume a rough coordinate $\mathbf{x}$ (denoted by $\mathbf{T}.\mathbf{x}$), *i.e.,* a pair of latitude and longitude, is available.

B. Preliminaries of VI and VAEs

The probabilistic methods or Bayesian generative methods provide us a principled way to explain the generation of observed data. They permit us to incorporate our prior knowledge regarding data into the model design. Specifically, they offer us two ways to achieve this 1) introducing appropriate latent variables $\mathbf{z}$ to serve as explanatory factors; 2) describing the generation of observed data by specifying proper generative process based on our domain knowledge. Generally, the probabilistic methods can be formulated as follows: we first draw a latent variable $\mathbf{z}$ from a prior distribution $p(\mathbf{z})$, and then relate $\mathbf{z}$ to observation $\mathbf{x}$ through a conditional distribution $p(\mathbf{x}|\mathbf{z})$; lastly, we intend to infer the posterior distribution $p(\mathbf{z}|\mathbf{x})$, which will be used in the prediction stage.

One of main challenges in adopting probabilistic methods lies on the posterior distribution $p(\mathbf{z}|\mathbf{x})$ inference. Using the Bayes rule, we have

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})}{p(\mathbf{x})},$$

and the difficulty here is the computation of marginal distribution $p(\mathbf{x}) = \int p(\mathbf{z}, \mathbf{x})d\mathbf{z}$ is intractable in high dimensional space. The traditional inference method Markov Chain Monte Carlo (MCMC) is usually only applicable to small datasets and simple models [34]. An alternative method, Variational Inference (VI), turns the posterior inference into an optimization problem. In comparison to MCMC, VI is much more efficient by taking advantage of numerical optimization algorithms.

In VI, we posit a family of densities $\mathcal{Q}$ and then search the optimal posterior approximation $q^*(\mathbf{z}) \in \mathcal{Q}$ that is closest to $p(\mathbf{z}|\mathbf{x})$ measured by KL divergence, *i.e.,*

$$q^*(\mathbf{z}) = \operatorname{argmin}_{q \in \mathcal{Q}} \mathrm{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})).$$

Extending the KL divergence term we get

$$\begin{aligned}\mathrm{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_{q(\mathbf{z})}\left[\log q(\mathbf{z})\right] - \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{q(\mathbf{z})}\left[\log q(\mathbf{z})\right] - \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{z}, \mathbf{x})\right] \\ &\quad + \log p(\mathbf{x}),\end{aligned}$$

from which we can conclude

$$\mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{z}, \mathbf{x})\right] - \mathbb{E}_{q(\mathbf{z})}\left[\log q(\mathbf{z})\right] \tag{1}$$

is a lower bound of the log-likelihood $\log p(\mathbf{x})$ since KL divergence is nonnegative. This lower bound is commonly known as Evidence Lower Bound (ELBO) in the VI literature, maximizing ELBO is equivalent to minimizing $\mathrm{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}))$ or maximizing the log-likelihood $\log p(\mathbf{x})$.

In the past decades, the most widely adopted VI is mean-field VI due to its simplicity. Mean-field VI assumes that the approximated posterior distribution $q(\mathbf{z})$ admits a factorized form as $q(\mathbf{z}) = \prod_{j=1}^{|\mathbf{z}|} q(z_j)$, i.e., all z_j are mutually independent and each z_j is governed by its own factor distribution $q(z_j)$, whose parameters are referred to as variational parameters. Mean-field VI requires us to specify the parametric form for each factor distribution $q(z_j)$, and derive their parameter iterative equations by hand. The drawback is that it constrains us to build models within only a small fraction of probability distributions; otherwise, no parameter iterative equation exists. This implies the resulting models may lack the flexibility to explain the observed data. Moreover, the optimization of mean-field VI often relies on the Coordinate Ascent algorithm which also struggles when the datasets are very large [35].

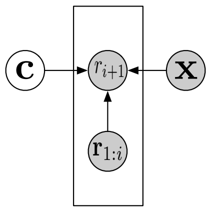


Fig. 2. The graphical model of the generative process in which $\mathbf{c} \in \mathbb{R}^{|\mathbf{c}|}$ is latent variable and represents the real-time traffic, $\mathbf{x}$ is the trip destination.

To address the drawbacks of mean-field VI, Variational Auto-Encoders (VAEs) [25], [26] combine the automatic-differentiation [36], the core ingredient of deep learning, with variational inference, which yields a flexible yet efficient inference framework for probabilistic generative methods. VAEs [25] replace $q(\mathbf{z})$ with $q(\mathbf{z}|\mathbf{x})$ and rewrite ELBO as

$$\mathrm{ELBO} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}(q(\mathbf{z}|\mathbf{x})|| \log p(\mathbf{z})), \tag{2}$$

and parameterize $q(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z})$ with neural networks, which are referred to as inference network and generative network respectively. More specifically, VAEs assume $q(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z})$ follow two parametric distributions, and fit the parameters of the two distributions with two neural networks. The inference network takes a datapoint $\mathbf{x}$ as input and produces a corresponding latent variable $\mathbf{z}$; while the generative network takes $\mathbf{z}$ as input and tries to decode $\mathbf{x}$. The ELBO serves as a loss function for both inference network and generative network, which is estimated with the Monte Carlo method by drawing L samples, $\mathbf{z}^{(l)}$, from $q(\mathbf{z}|\mathbf{x})$

$$\mathrm{ELBO} \approx \frac{1}{L}\sum_{l=1}^{L} \log p(\mathbf{x}|\mathbf{z}^{(l)}) - \mathrm{KL}(q(\mathbf{z}|\mathbf{x})|| \log p(\mathbf{z})), \tag{3}$$

where the KL term has analytic solution for the Gaussian prior and posterior. The parameters of the two networks are optimized by stochastic gradient descent algorithms which scale to large-scale datasets easily.

IV. The Proposed Method – DeepST

We first present the general idea of our proposed method and the generative process in Section IV-A. We then detail the representation learning of past traveled route and destination in Section IV-B and Section IV-C, respectively. The developed inference method is presented in Section IV-D, and the prediction is discussed in Section IV-E. We present the complexity analysis of DeepST in Section IV-F

A. The generative process

The high level idea of our proposed method DeepST is that we interpret the generation of a route by conditioning on the past traveled road segment sequence, the representations of destination and real-time traffic. In such a manner, DeepST simultaneously takes into consideration the aforementioned three key explanatory factors–the sequential property of transition, the global impact of destination and real-time traffic. The intuition is that once the generative model is trained on the observed trajectories, its parameters will be tuned to capture the spatial transition patterns hidden in the dataset, and thus could make prediction based on the learned patterns in the future. The graphical model is shown in Figure 2 and the generative process of one route is described as follows.

- Draw $\mathbf{c} \sim \mathrm{Normal}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$.
- For $i+1$-th road segment, $i \geq 1$
  - Draw $r_{i+1} \sim \mathbb{P}(r_{i+1}|\mathbf{r}_{1:i}, \mathbf{x}, \mathbf{c})$.
  - Draw ending indicator $s \sim \mathrm{Bernoulli}(f_s(r_{i+1}, \mathbf{x}))$.
  - If $s = 0$ then continue else end the generation.

We first draw a latent variable $\mathbf{c} \in \mathbb{R}^{|\mathbf{c}|}$ which represents the real-time traffic from an isotropic Gaussian prior distribution. Then the $i+1$-th road segment is drawn from $\mathbb{P}(r_{i+1}|\mathbf{r}_{1:i}, \mathbf{x}, \mathbf{c})$. Finally, we use a Bernoulli distribution whose parameter is the output of function f_s taking $r_{i+1}, \mathbf{x}$ as input, to decide whether to terminate the generation, and we use the Euclidean distance between the projection point of $\mathbf{x}$ on r_{i+1} and $\mathbf{x}$ to determine the termination of the generation, *i.e.,*

$$f_s(r_{i+1}, \mathbf{x}) = \frac{1}{1 + \|\mathbf{p}(\mathbf{x}, r_{i+1}) - \mathbf{x}\|_2},$$

where $\mathbf{p}(\mathbf{x}, r_{i+1})$ denotes the projection point of $\mathbf{x}$ on r_{i+1}, the Bernoulli distribution is chosen since s is a binary variable [37]. In this paper, we propose to model $\mathbb{P}(r_{i+1}|\mathbf{r}_{1:i}, \mathbf{x}, \mathbf{c})$ as a Categorical distribution (the categories are all road segments adjacent to r_i) whose parameter is the output of a function f taking the past traveled route sequence $\mathbf{r}_{1:i}$, destination $\mathbf{x}$ and real-time traffic $\mathbf{c}$ as input; f could be any differentiable function. In this paper, we use the additive function due to its simplicity,

$$\mathbb{P}(r_{i+1}|\mathbf{r}_{1:i}, \mathbf{x}, \mathbf{c}) = \mathrm{softmax}\left(\alpha^\top f_{\mathbf{r}}\left(\mathbf{r}_{1:i}\right) + \beta^\top f_{\mathbf{x}}(\mathbf{x}) + \gamma^\top \mathbf{c}\right)$$

in which $f_{\mathbf{r}}\left(\mathbf{r}_{1:i}\right) \in \mathbb{R}^{n_{\mathbf{r}}}, f_{\mathbf{x}}(\mathbf{x}) \in \mathbb{R}^{n_{\mathbf{x}}}$ are representations of past traveled route $\mathbf{r}_{1:i}$ and destination $\mathbf{x}$, respectively; $\alpha \in \mathbb{R}^{n_{\mathbf{r}} \times \mathcal{N}(r_i)}, \beta \in \mathbb{R}^{n_{\mathbf{x}} \times \mathcal{N}(r_i)}, \gamma \in \mathbb{R}^{|\mathbf{c}| \times \mathcal{N}(r_i)}$ are projection matrices that map the corresponding representations into the road segment space, and $\mathcal{N}(r_i)$ is the number of adjacent road segments of r_i, *e.g.,* $\mathcal{N}(r_2) = 3$ in Figure 1a. The projection matrices are shared across all road segments, and since different r_i may have different number of adjacent road segments, we substitute $\mathcal{N}(r_i)$ with $\max_{\forall r \in E} \mathcal{N}(r)$, *i.e.,* the maximum number of neighboring road segments on the road network. Note that since the model is data-driven, its parameters will be tuned to push the probability mass of $\mathbb{P}(r_{i+1}|\mathbf{r}_{1:i}, \mathbf{x}, \mathbf{c})$ only towards the road segments that are truly adjacent to r_i.

B. The representation of past traveled route

Most of existing studies [1], [5], [8], [10] model the transition relationship with Markov Model which requires to make explicit dependency assumption to make inference tractable [6]. In contrast, the Recurrent Neural Nets are capable of modeling the long range dependency by embedding a sequence of tokens into a vector. As presented in Section I, the transition patterns of vehicle on road network may demonstrate strong sequential property, it is desirable to capture such long range dependency when modeling the spatial transition, and thus we adopt the RNNs to squeeze the past traveled route into its representation. Specifically, we update the i-th hidden state as follows

$$\mathbf{h}_i = \begin{cases} \mathbf{0} & i = 1 \\ \mathrm{GRU}\left(\mathbf{h}_{i-1}, r_{i-1}\right) & i \geq 2 \end{cases}$$

where $\mathbf{h}_1$ is initialized as a zero vector and $\mathrm{GRU}(\cdot, \cdot)$ represents the Gated Recurrent Unit updating function [38]. Eventually, we choose i-th hidden state $\mathbf{h}_i$ as the representation of past traveled route, *i.e.,* we let $f_{\mathbf{r}}\left(\mathbf{r}_{1:i-1}\right) = \mathbf{h}_i$.

C. The representation of destination

The trip destinations have a global impact on the spatial transition, and thus it is critically important to properly learn the destination representations $f_{\mathbf{x}}(\mathbf{x}) \in \mathbb{R}^{n_{\mathbf{x}}}$ to help the route decision. Previous study [7] assumed that the destination road segments of the trips are available, and learned the representations for these road segments to guide the route decision. However, the exact destination road segment of a trip may not be available at the start of a trip. Furthermore, the method [7] learns representation of each road segment separately cannot effectively share the statistical strength across trips with the spatially close destinations. As an example, assuming we model the transition probabilities for the trips shown in Table I; if we treat the destinations separately as does the work [7], we will have

$$\mathbb{P}(r_{10}|r_5, C) = 1/3, \mathbb{P}(r_6|r_5, A) = 1/3, \mathbb{P}(r_6|r_5, B) = 1/3;$$

however the spatial proximity between A and B is small, and if we use one representation AB for all trips driving towards them, we will have

$$\mathbb{P}(r_{10}|r_5, C) = 1/3, \mathbb{P}(r_6|r_5, AB) = 2/3,$$

i.e., the transition patterns can be shared to mutually reinforce the transition probability between r_5 and r_6 across trips.

Intuitively, the trips whose destinations are spatially close should share similar destination representations. More importantly, the learned destination representations are also supposed to be able to effectively guide the spatial transition of their corresponding trips. Separating the destination representations learning and spatial transition learning into two stages prevents end-to-end training, which may lead to a suboptimal solution. Instead, it is desirable to jointly learn the destination representations and model the spatial transition such that the statistical strength could be effectively shared across trips. To summarize, when learning the destination representations

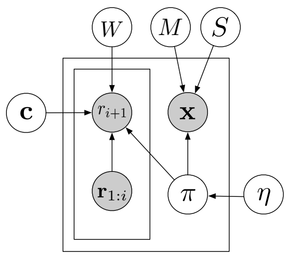


Fig. 3. The graphical model of the complete generative process in which $\mathbf{c} \in \mathbb{R}^{|\mathbf{c}|}$, $\boldsymbol{\pi} \in \mathbb{R}^K$ are latent variables, $M \in \mathbb{R}^{2\times K}$, $S \in \mathbb{R}_+^{2\times K}$, $W \in \mathbb{R}^{n_{\mathbf{x}}\times K}$ are parameters to be learned, and $\boldsymbol{\eta} \in \mathbb{R}^K$ is hyperparameter; $\boldsymbol{\pi}$ can be interpreted as the destination allocation indicator, the nonzero dimension of $\boldsymbol{\pi}$ indicates the proxy which the trip is allocated.

we should consider two points: 1) if the spatial proximity between the destinations of two trips is small, the learned destination representations should be similar; 2) the learned representations could effectively guide the spatial transition of their corresponding trips.

Towards this end, we propose to learn representations for the K-destination proxies that are shared by all trips, instead of learning each trip a separate destination representation. Specifically, we simultaneously learn representations of K-spatial locations, referring to as K-destination proxies, and adaptively allocate the trips into one of these destination proxies; the destination proxy representation will be used to guide the spatial transition of trips that are allocated to this proxy. We achieve this by developing an adjoint generative model which explains the generation of the destination coordinates with another latent variable $\boldsymbol{\pi}$. The adjoint generative process is as follows.

- Draw $\boldsymbol{\pi} \sim \text{Categorical}(\boldsymbol{\eta})$.
- Draw $\mathbf{x} \sim \text{Normal}(M\boldsymbol{\pi}, \text{diag}(S\boldsymbol{\pi}))$.

where $\boldsymbol{\eta} \in \mathbb{R}^K$ is a hyperparameter, $\boldsymbol{\pi} \in \mathbb{N}^K$ is a one-hot vector indicating which proxy the trip is allocated. Then we use $\boldsymbol{\pi}$ and M, S to generate the destination coordinate $\mathbf{x}$, where $M \in \mathbb{R}^{2\times K}$ is the mean value matrix and $S \in \mathbb{R}_+^{2\times K}$ is the variance matrix of the K-destination proxies; the operation $M\boldsymbol{\pi}$ (resp. $S\boldsymbol{\pi}$) selects one column of M (resp. S) which corresponds to the mean (resp. variance) of the allocated proxy. We interpret the observation $\mathbf{x}$ via conditioning on $M\boldsymbol{\pi}$ and $S\boldsymbol{\pi}$ with a Gaussian distribution which could tolerate the small deviation of $\mathbf{x}$ from the proxy mean value $M\boldsymbol{\pi}$, *i.e.,* adding some small noise to $\mathbf{x}$ does not change its proxy. Hence the resulting method will be robust to the inaccuracy of observations.

In such a fashion, we actually allocate each trip destination $\mathbf{x}$ to a proxy and the trips that are allocated to the same proxy will share the same proxy representation, so as to effectively share the statistical strength across these trips. As our eventual purpose is to obtain the proxy destination representation, we introduce an embedding matrix $W \in \mathbb{R}^{n_{\mathbf{x}}\times K}$ and let $f_{\mathbf{x}}(\mathbf{x}) = W\boldsymbol{\pi}$. We can see that the introduction of the latent variable $\boldsymbol{\pi}$ enables achieving the aforementioned two points simultaneously. On the one hand, $M\boldsymbol{\pi}$ and $S\boldsymbol{\pi}$ are supposed to be capable of explaining the observations $\mathbf{x}$, so as to satisfy the spatial constraint. On the other hand, $W\boldsymbol{\pi}$ should be able to yield a useful destination representation that can effectively guide the spatial transition of the trips, since this representation is learned by maximizing the probability of observed routes in $\mathbb{P}(r_{i+1}|\mathbf{r}_{1:i}, W\boldsymbol{\pi}, \mathbf{c})$. Consequently, the complete graphical model becomes the one shown in Figure 3. We can consider $\boldsymbol{\pi}$ as the destination allocation indicator, and the nonzero dimension of $\boldsymbol{\pi}$ indicates the proxy to which the trip is allocated.

D. Inference with VAEs

The ultimate generative process described in Figure 3 implies the following log-likelihood function for one trip,

$$\mathcal{L}(\Theta) = \log \sum_{\boldsymbol{\pi}} \int_{\mathbf{c}} \mathbb{P}(\mathbf{c}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)\mathbb{P}(\boldsymbol{\pi}|\boldsymbol{\eta}) \times \prod_{i=1}^{n-1} \mathbb{P}\left(r_{i+1}|\mathbf{r}_{1:i}, W\boldsymbol{\pi}, \mathbf{c}\right) d\mathbf{c} + \log \sum_{\boldsymbol{\pi}} \mathbb{P}(\boldsymbol{\pi}|\boldsymbol{\eta})\mathbb{P}(\mathbf{x}|\boldsymbol{\pi}, M, S) \tag{4}$$

where Θ is the collection of all parameters involved. The first term describes the log-likelihood of route generation, while the second term represents the log-likelihood of destination generation. We are required to infer the posterior distributions $P(\mathbf{c}|\mathbf{r}, \mathbf{x})$, $P(\boldsymbol{\pi}|\mathbf{r}, \mathbf{x})$, and fit the parameters Θ by maximizing the log-likelihood given the observations. The exact computation of the log-likelihood function is intractable as it requires us to integrate (sum) out two latent variables $\mathbf{c}$, $\boldsymbol{\pi}$ in the high-dimensional space. The sampling-based methods [39] suffer from the slow convergence and are also not suitable in our scenario.

To circumvent this, we develop an approximate inference method within the Variational Auto-Encoders (VAEs) framework [25], [26]. The general idea behind VAEs is to fit the intractable posterior distributions with appropriate inference neural nets; since the exact computation of log-likelihood is difficult, VAEs resort to maximize the evidence lower bound (ELBO) of the log-likelihood and propagate the gradients backwards to optimize the inference neural nets. To this end, we first derive the ELBO of $\mathcal{L}(\Theta)$ using the Jensen's Inequality [25] as follows,

$$\mathcal{L}(\Theta) \geq \mathbb{E}_{q(\mathbf{c},\boldsymbol{\pi}|\mathbf{r},\mathbf{x})}\left[\log \frac{\mathbb{P}(\mathbf{c})\mathbb{P}(\boldsymbol{\pi})\prod_{i=1}^{n-1}\mathbb{P}\left(r_{i+1}|\mathbf{r}_{1:i}, W\boldsymbol{\pi}, \mathbf{c}\right)}{q(\mathbf{c}, \boldsymbol{\pi}|\mathbf{r}, \mathbf{x})}\right] + \mathbb{E}_{q(\boldsymbol{\pi}|\mathbf{r},\mathbf{x})}\left[\log \frac{\mathbb{P}(\boldsymbol{\pi})\mathbb{P}(\mathbf{x}|\boldsymbol{\pi}, M, S)}{q(\boldsymbol{\pi}|\mathbf{r}, \mathbf{x})}\right] \tag{5}$$

in which $q(\mathbf{c}, \boldsymbol{\pi}|\mathbf{r}, \mathbf{x})$ and $q(\boldsymbol{\pi}|\mathbf{r}, \mathbf{x})$ are the approximated posterior distributions to be optimized. $q(\mathbf{c}, \boldsymbol{\pi}|\mathbf{r}, \mathbf{x})$ represents the joint posterior distribution of $\mathbf{c}, \boldsymbol{\pi}$ given the route $\mathbf{r}$ and destination $\mathbf{x}$. Since $\boldsymbol{\pi}$ denotes the proxy that $\mathbf{x}$ is allocated to, and this allocation is invariant to the traveled route $\mathbf{r}$, we factorize the joint distribution as

$$q(\mathbf{c}, \boldsymbol{\pi}|\mathbf{r}, \mathbf{x}) = q(\mathbf{c}|\mathbf{r})q(\boldsymbol{\pi}|\mathbf{x})$$

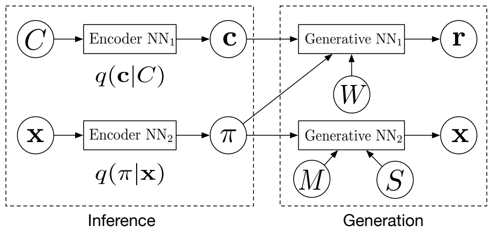


Fig. 4. The inference framework.

and for the same reason, $q(\boldsymbol{\pi}|\mathbf{x}) = q(\boldsymbol{\pi}|\mathbf{r}, \mathbf{x})$. But our scenario slightly differs from the typical VAEs here: because the data to be generated is observable in the typical usage of VAEs when performing inference; however, in our case we seek to predict the most likely route $\mathbf{r}$ which is not available at the time of prediction.

To sidestep this, we note that $q(\mathbf{c}|\mathbf{r})$ actually manages to infer the posterior distribution of traffic condition $\mathbf{c}$ from the transition patterns of $\mathbf{r}$. Intuitively, the real-time traffic at the start of trip $\mathbf{T}$ can also be measured by the average speed of (sub-)trajectories in the time window $[\mathbf{T}.s - \Delta, \mathbf{T}.s)$, Δ is a specified parameter *e.g.,* 20 minutes. We denote these (sub-)trajectories as C (or $\mathbf{T}.C$), and propose to extract the real-time traffic representation $\mathbf{c}$ from C. To this end, we partition the space into cells and calculate the average vehicle speed in a cell by using the real-time trajectories. However, the raw cell representation is sensitive to the spatial distribution of vehicles, *i.e.,* if there is no sensing vehicle passing the cell at the moment, the cell value will become zero; furthermore, the raw cell representation cannot reveal the similarity between two analogous traffic conditions, even if only one cell value changes, the representation will be considered to be different from the original one. Therefore, similar to [32], we use a Convolutional Neural Net (CNN) to extract the high level features from the raw cell representation. More formally, we have

$$\begin{aligned} \mathbf{f} &= \mathrm{CNN}(C) \\ q(\mathbf{c}|C) &= \mathrm{Normal}\left(\mu(\mathbf{f}), \mathrm{diag}\left(\sigma^2(\mathbf{f})\right)\right) \end{aligned} \tag{6}$$

where $\mu(\mathbf{f}), \sigma^2(\mathbf{f})$ are parameterized by two MLPs (Multiple Layer Perceptrons) with shared hidden layer.

Eventually, our inference framework is shown in Figure 4. The Encoder NN_1 is the stack of CNN+MLPs, which approximates the posterior distribution $q(\mathbf{c}|C)$; Encoder NN_2 approximates $q(\boldsymbol{\pi}|\mathbf{x})$ and is also parameterized by a MLP. The two encoders are referred to as inference nets, and the parameters set involved are denoted as Φ. For a given trip $\mathbf{T}$, we feed $\mathbf{T}.C$ and $\mathbf{T}.\mathbf{x}$ into the inference nets to draw the instances of random variable $\mathbf{c}, \boldsymbol{\pi}$; then the sampled instances are fed into the two generative processes represented by Generative NN_1 and Generative NN_2 in Figure 4, to compute the subsequent loss. To be specific, with the generated L samples $\{\boldsymbol{\pi}^{(l)}, \mathbf{c}^{(l)}\}_{l=1}^{L}$ we can estimate the ELBO using the

Algorithm 1: Learning Algorithm

```
   Input: Training dataset: $(\mathbf{r}_m, C_m, \mathbf{x}_m)_{m=1}^M$
   Output: Parameter sets: $\Theta$ and $\Phi$
1  while training is true do
2      $\mathcal{B} \leftarrow$ A random minibatch of data;
3      for $b = 1, \ldots, |\mathcal{B}|$ do
4          $\mathbf{c}^{(b)} \sim q(\mathbf{c}^{(b)}|C^{(b)})$;
5          $\boldsymbol{\pi}^{(b)} \sim q(\boldsymbol{\pi}^{(b)}|\mathbf{x}^{(b)})$;
6      ELBO $\leftarrow$ Calculating the ELBO using Equation 7;
7      for $\theta \in \Theta \cup \Phi$ do
8          $\mathbf{g}_\theta \leftarrow \nabla_\theta$ELBO;
9          $\theta \leftarrow \theta + \Gamma(\mathbf{g}_\theta)$;
10 return $\Theta$ and $\Phi$;
```

Monte Carlo method as

$$\begin{aligned} \mathrm{ELBO} \approx\ & \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{n-1}\log\mathbb{P}\left(r_{i+1}|\mathbf{r}_{1:i}, W\boldsymbol{\pi}^{(l)}, \mathbf{c}^{(l)}\right) \\ & + \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{n-1}\log\mathbb{P}(\mathbf{x}|\boldsymbol{\pi}^{(l)}, M, S) \\ & - \mathrm{KL}(q(\mathbf{c}|C)||\mathbb{P}(\mathbf{c})) - 2\mathrm{KL}(q(\boldsymbol{\pi}|\mathbf{x})||\mathbb{P}(\boldsymbol{\pi})) \end{aligned} \tag{7}$$

where $\mathrm{KL}(\cdot||\cdot)$ denotes the KL-divergence. Equation 7 involves evaluation of log-probability of two distributions, $\mathbb{P}\left(r_{i+1}|\mathbf{r}_{1:i}, \mathbf{x}, \mathbf{c}\right)$ and $\mathbb{P}(\mathbf{x}|\boldsymbol{\pi}, M, S)$, note that we ignore the superscript (l) for clarity. The first distribution characterizes the probability of next possible road r_{i+1} with a $\mathrm{softmax}$ distribution, and the log-probability can be calculated as follows,

$$\log\mathbb{P}\left(r_{i+1}|\mathbf{r}_{1:i}, \mathbf{x}, \mathbf{c}\right) = \log\frac{\exp(\alpha_j^\top\mathbf{h}_i + \beta_j^\top W\boldsymbol{\pi} + \gamma_j^\top\mathbf{c})}{\sum_{j'}\exp(\alpha_{j'}^\top\mathbf{h}_i + \beta_{j'}^\top W\boldsymbol{\pi} + \gamma_{j'}^\top\mathbf{c})}$$

where α_j is the j-th column of α. The second distribution is the Normal distribution with mean $M\boldsymbol{\pi}$ and variance $S\boldsymbol{\pi}$, whose log-probability calculation is straightforward. When the approximated posterior $q(\mathbf{c}|C)$ and prior $p(\mathbf{c})$ are both Gaussian distributions, the KL-divergence $\mathrm{KL}(q(\mathbf{c}|C)||p(\mathbf{c}))$ has closed form solution,

$$\mathrm{KL}(q(\mathbf{c}|C)||p(\mathbf{c})) = \frac{1}{2}\sum_{i=1}^{|\mathbf{c}|}(1 - \mu_i^2 - \sigma_i^2 + \log\sigma_i^2).$$

To enable the gradients flowing back into the inference nets, we need to reparameterize the random variables when sampling. For the Gaussian random variable $\mathbf{c}$, we can reparameterize it as

$$\mathbf{c} \sim \mathrm{Normal}\left(\mathbf{c}|\mu, \sigma^2\right) \Leftrightarrow \mathbf{c} = \mu + \sigma\epsilon, \quad \epsilon \sim \mathrm{Normal}(0, 1).$$

As $\boldsymbol{\pi}$ is a discrete random variable which does not admit the above reparameterization trick [25], [26], we instead resort to the Gumbel-Softmax relaxation [40], [41].

The learning algorithm is presented in Algorithm 1. We highlight that it is unnecessary to compute C for each trip

Algorithm 2: Prediction Algorithm

```
Input: $(\Theta, \Phi, \mathbf{T}.r_1, \mathbf{T}.C, \mathbf{T}.\mathbf{x})$
Output: Generated route: $\mathbf{T}.\mathbf{r}$
1  $r_1 \leftarrow \mathbf{T}.r_1$, $C \leftarrow \mathbf{T}.C$, $\mathbf{x} \leftarrow \mathbf{T}.\mathbf{x}$;
2  Draw $\mathbf{c} \sim q(\mathbf{c}|C)$;
3  Draw $\boldsymbol{\pi} \sim q(\boldsymbol{\pi}|\mathbf{x})$;
4  for $i \geq 1$ do
5      Draw $r_{i+1} \sim \mathbb{P}(r_{i+1}|\mathbf{r}_{1:i}, W\boldsymbol{\pi}, \mathbf{c})$;
6      Draw ending indicator $s \sim$ Bernoulli$(f_s(r_{i+1}, \mathbf{x}))$;
7      if $s = 0$ then
8          continue;
9      else
10         break;
11 return $\mathbf{r}$;
```

in Line 4 of Algorithm 1, as we can discretize the temporal dimension into slots and let the trips whose start times fall into the same slot share one C. Since we take a mini-batch of data to compute the ELBO in Line 6 of Algorithm 1, the number of random variables L can be set to 1 in Equation 7 and we are still able to get low-variance gradient estimators in Line 8 of Algorithm 1. We update the parameters of the model with gradient descent in Line 9 of Algorithm 1. We also would like to point out that there might be multiple routes between a pair of origin and destination in the training dataset. In such a case the model will learn to assign these routes different likelihood scores since all of them are used in the training stage, and in the prediction stage only the one with the highest likelihood score will be returned.

E. Route prediction and likelihood score

Route prediction. In route prediction, given the trained network parameter sets Θ, Φ, the initial road $\mathbf{T}.r_1$, the set (sub-)trajectories $\mathbf{T}.C$ for referring real-time traffic, and destination $\mathbf{T}.\mathbf{x}$ of a trip $\mathbf{T}$, we run the complete generative process depicted in Figure 3 to generate the most probable route for the trip $\mathbf{T}$. The prediction algorithm is presented in Algorithm 2. Unlike the training stage, the prediction algorithm samples latent variables $\mathbf{c}$, $\boldsymbol{\pi}$ from the learned posterior distributions (Line $1-2$); we then use the sampled $\mathbf{c}$, $\boldsymbol{\pi}$ as well as the learned parameter W to generate the next road link r_{i+1} (Line 4), which is defined as

$$\mathbb{P}(r_{i+1}|\mathbf{r}_{1:i}, W\boldsymbol{\pi}, \mathbf{c}) = \mathrm{softmax}\left(\alpha^\top \mathbf{h}_i + \beta^\top W\boldsymbol{\pi} + \gamma^\top \mathbf{c}\right).$$

Route likelihood score. Scoring the likelihood of a given route $\mathbf{r}$ is similar to the route prediction except that the route is fixed. Once we draw $\mathbf{c}$ and $\boldsymbol{\pi}$ from the posterior distribution, the likelihood of $\mathbf{r}$ can be calculated as $\prod_{i=1}^{|\mathbf{r}|-1} \mathbb{P}(r_{i+1}|\mathbf{r}_{1:i}, W\boldsymbol{\pi}, \mathbf{c})$.

F. The complexity analysis

As our model is trained with SGD method, the convergence time is linearly to the number of trajectories in the training dataset. Given a trained model, the complexity of the route prediction and route likelihood score both are $O(|\mathbf{r}|)$. We empirically study the scalability of DeepST in Section V-D.

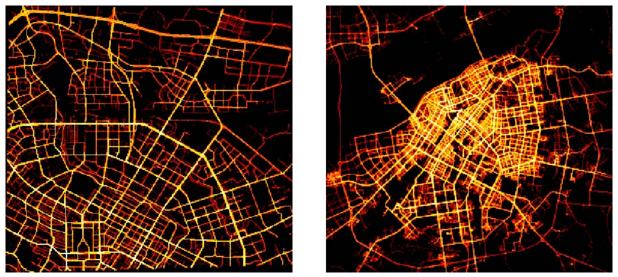

(a) Chengdu (b) Harbin

Fig. 5. The spatial distribution of the GPS points: (a) Chengdu with size 10×10 (km); (b) Harbin with size 28×30 (km).

TABLE III
DATASET STATISTICS.

City	**Chengdu**			**Harbin**		
Measures	min	max	mean	min	max	mean
Distance (km)	1.0	40	4.9	1.1	60	11.4
#road segments	5	85	14	5	102	24

V. EXPERIMENTS

We evaluate the effectiveness of DeepST on two real-world large-scale taxi trajectory datasets, against the state-of-art baseline methods in this section.

A. Experimental setup

Dataset description. We use two real-world large-scale trajectory datasets in our experiments.

- The first dataset is a **publicly available dataset** released by DiDi Company[1]. It was collected by 33,000 taxi cabs in a provincial capital city, Chengdu, in China. The sampling rate is around 3 seconds. We include its first 15 days' data, over 3 million trajectories, in our experiments.
- The second dataset contains over 2.9 million trajectories, which were collected by 13,000 taxi during one month in another provincial capital city, Harbin, in China. The sampling time interval is around 30 seconds per GPS point, and the accuracy of existing map-matching algorithm can be 99% at this sampling-rate [42].

Figure 5 shows the spatial distributions of the GPS points of the two datasets. The road network data is collected from the OpenStreetMap [43], and the Chengdu and Harbin datasets cover 3,185 and 12,497 road segments respectively. Figure 6 plots the distributions of travel distance and the number of road segments covered by the trips. Table III reports the basic statistics of the trips. The mean travel distance and number of road segments are 4.8 (km) and 14 respectively for the Chengdu dataset, and 11.4 (km) and 24 respectively for the Harbin dataset.

[1]https://outreach.didichuxing.com/research/opendata/en

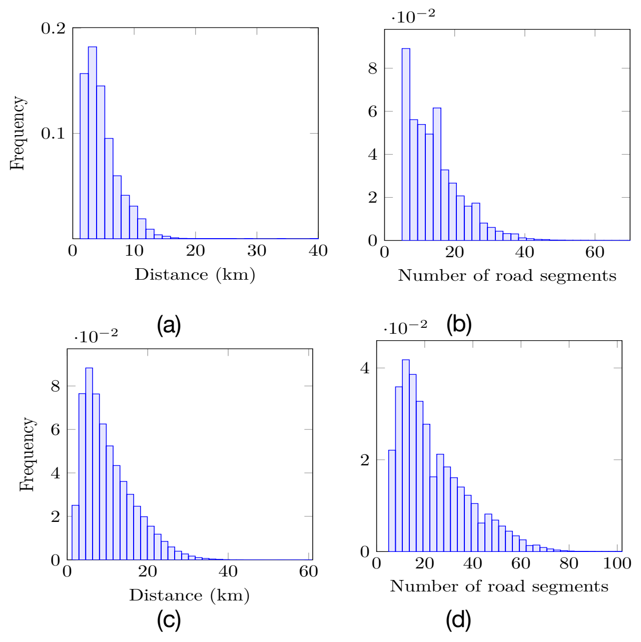


Fig. 6. The distributions of travel distance (km) and number of road segments: (a)-(b) for Chengdu; (c)-(d) for Harbin.

For the Chengdu dataset, we use its first 8 days' trajectories as training dataset, and the next 2 days' trajectories as validation, and the remaining ones are used for testing. For the Harbin dataset, we use the first 18 days' trajectories as training dataset, and the next 3 days' trajectories as validation, and the remaining ones are used for testing. As a result, the dataset size of training, validation, and testing is 1.6 (resp. 1.7), 0.4 (resp. 0.3), and 1.0 (resp. 0.9) million for Chengdu (resp. Harbin) dataset respectively.

Baseline Methods. DeepST is evaluated against baseline methods on two tasks: i) the most likely route prediction, and ii) route recovery from sparse trajectories. For the task of the most likely route prediction, we compare DeepST with DeepST-C, RNN, MMI (the first-order Markov Model), WSP (Weighted Shortest Path), and the state-of-art route decision model CSSRNN [7]. For the task of route recovery from sparse trajectories, DeepST is compared with STRS [2], which has been shown to be superior to other route recovery methods including HRIS [4], InferTra [1] and MPR [3].

- DeepST-C is a simplified version of DeepST without considering the impact of real-time traffic.
- RNN is the vanilla RNN that only takes the initial road segment as input. It is a further simplified DeepST by ignoring the impact of both the destination and real-time traffic.
- CSSRNN [7] is the state-of-art route decision model. It assumes the last road segments of the trips are known in advance and learns their representations to help model the spatial transition.
- MMI models the spatial transition by calculating the first-order conditional probability between adjacent road segments from the historical trips.
- WSP always returns the shortest path from the origin road segment to the destination road segment on the weighted road network. The edge weight equals to the mean travel time of corresponding road segment, and the mean travel time is estimated using the entire historical dataset.
- STRS [2] is the state-of-art route recovery method comprising of a travel time inference module and a spatial transition inference module. We substitute its spatial transition inference module with DeepST to demonstrate how DeepST can be used to enhance its performance.

Platform and Parameter Setting. The platform runs Ubuntu 16.04 OS, and the modelis implemented with PyTorch 1.0, and trained with one Tesla P100 GPU around 6 hours (resp. 10 hours) for the Chengdu (resp. Harbin) dataset. For the Harbin dataset, the number of destination proxies K is 1000; we partition the space into a 138×148 matrix with cell size 200m$\times$200m. For the Chengdu dataset, K is set to 500; the space is partitioned into a 87×98 matrix with cell size 100m$\times$100m. The CNN in Equation 6 comprises of three connected convolution blocks followed by an average pooling layer; each convolution block consists of three layers: $\text{Conv2d} \rightarrow \text{BatchNorm2d} \rightarrow \text{LeakyReLU}$. The dimension of real-time traffic representation $|\mathbf{c}|$ and the hidden size of all MLPs used are 256, $n_{\mathbf{r}}$, $n_{\mathbf{x}}$ are set as 128. We choose a three-layer stacking GRU with hidden size 256 for all RNNs used in the experiments. The time window size $\Delta = 30$ (minutes), and the temporal dimension is discretized into 20 (minutes) size slots, two trips share the same C if their start times fall into the same slot. The batch size $|\mathcal{B}|$ is 128. The model is trained with Adam [44] for 15 epochs, and early stopping is used on the validation dataset.

B. The most likely route prediction

We evaluate different methods on the most likely traveled route prediction task in terms of two measurements: recall@n and accuracy. 1) Denoting the ground truth route as $\mathbf{r}$, we first generate the most likely route $\hat{\mathbf{r}}$ using different methods. Since the generated route $\hat{\mathbf{r}}$ can be arbitrarily long, to give a fair comparison, we truncate $\hat{\mathbf{r}}$ by only preserving the first $|\mathbf{r}|$ road segments, denoting as $\hat{\mathbf{r}}_t$, and recall@n is defined as the ratio of the length of correctly predicted road segments over the length of ground truth $\mathbf{r}$, *i.e.,*

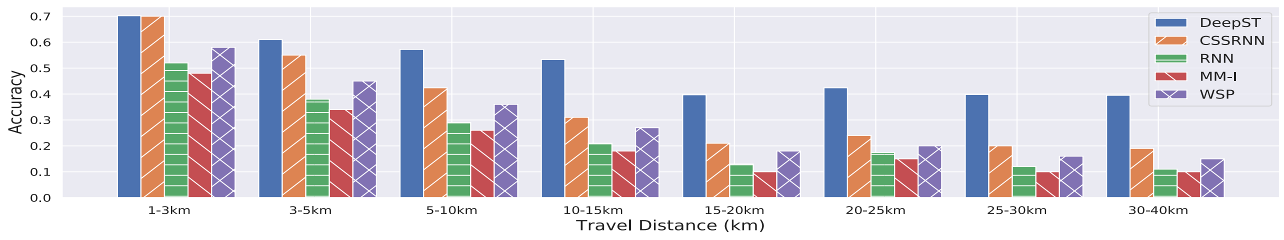
$$\text{recall@}n = \frac{|\mathbf{r} \cap \hat{\mathbf{r}}_t|}{|\mathbf{r}|}; \tag{8}$$

2) accuracy [2] is defined as the ratio of the length of correctly predicted road segments over the maximum value of the length of ground truth $\mathbf{r}$ and the length of predicted route $\hat{\mathbf{r}}$, *i.e.,*

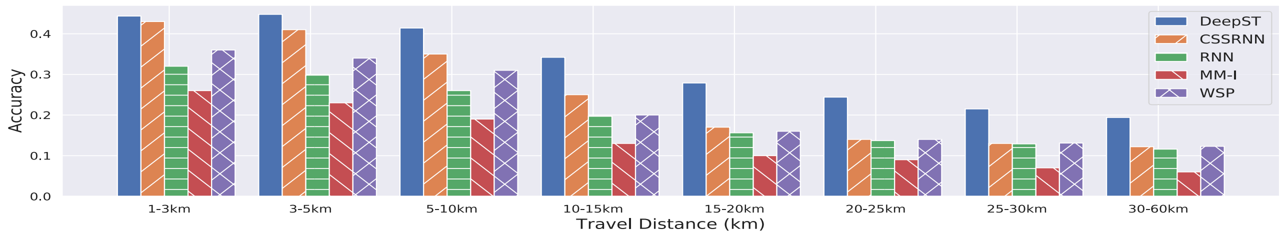
$$\text{accuracy} = \frac{|\mathbf{r} \cap \hat{\mathbf{r}}|}{\max(|\mathbf{r}|, |\hat{\mathbf{r}}|)}. \tag{9}$$

TABLE IV
OVERALL PERFORMANCE.

City	Chengdu						Harbin					
Method	DeepST	DeepST-C	CSSRNN	RNN	MMI	WSP	DeepST	DeepST-C	CSSRNN	RNN	MMI	WSP
recall@n	**0.637**	0.626	0.577	0.409	0.318	0.431	**0.397**	0.385	0.336	0.261	0.202	0.267
accuracy	**0.612**	0.601	0.556	0.389	0.281	0.431	**0.374**	0.366	0.313	0.172	0.154	0.267



(a) Chengdu



(b) Harbin

Fig. 7. The route prediction accuracy of different methods versus travel distance.

Overall Performance. Table IV shows the overall performance of different methods on the two datasets. RNN outperforms MMI, as it can capture the long-range dependency in the spatial transition patterns. Both RNN and MMI are worse than WSP in terms of recall@n and accuracy; because without considering the destinations, the former two methods will always make identical predictions for all trips that start from the same initial road segment. CSSRNN performs much better than RNN, MMI and WSP as it explicitly incorporates the influence of destinations by learning their distributed representations. This indicates that the destinations play a very important role in the route decision. DeepST surpasses CSSRNN by 10.4% (resp. 18.2%) in terms of recall@n and by 10.1% (resp. 19.5%) in terms of accuracy on the Chengdu (resp. Harbin) dataset, as it simultaneously models the spatial transition sequential property, the impact of destinations and real-time traffic in a principled way. All methods show better performance on the Chengdu dataset than on the Harbin dataset. This could be because the mean length of trips in the Harbin dataset is much longer and the road network topological structure of Harbin is more complex as shown in Figure 5, and thus the task on the Harbin dataset is more challenging.

Effectivenss of K-destination proxies. As shown in Table IV even without considering the real-time traffic, DeepST-C is able to surpass CSSRNN, which learns destination representations separately, by a fair margin. This verifies our conjecture that learning representations for the destinations separately cannot effectively share the statistical strength across trips and also demonstrates the effectiveness of our proposed K-destination proxies.

Impact of travel distance. We further study the impact of travel distance on the performance of different methods. To this end, we partition the trips in the test dataset by their length (km) into 8 buckets, $[1, 3)$,$[3, 5)$ $[5, 10)$, $[10, 15)$, $[15, 20)$, $[20, 25)$, $[25, 30)$, $[30, -)$ and calculate accuracy of different methods on the buckets. Figure 7 shows the accuracy of different methods over the travel distance. For the short trips (<3km), both DeepST and CSSRNN show much better performance than the other three methods. As the travel distance grows, not surprisingly, the performance of all methods drops. This is because as the travel distance increases, the number of possible routes between the origin and destination grows exponentially, and the task of predicting the most likely route from them becomes more difficult. Nevertheless, DeepST is able to surpass the baseline methods on all buckets. As the travel distance grows up to 10km, the performance gap between DeepST and baseline methods becomes more evident; in particular, DeepST surpasses the best baseline method by almost 50% in terms of accuracy on

TABLE V
ROUTE RECOVERY ACCURACY VERSUS SAMPLING RATE.

City	Chengdu									Harbin								
Rate (mins)	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
STRS	0.98	0.96	0.93	0.90	0.86	0.82	0.79	0.73	0.69	0.98	0.95	0.93	0.89	0.85	0.82	0.76	0.70	0.65
STRS+	0.99	0.98	0.97	0.95	0.93	0.91	0.88	0.83	0.81	0.98	0.96	0.95	0.92	0.89	0.86	0.82	0.77	0.75
δ (%)	1.02	2.08	4.30	5.56	8.14	11.0	11.4	13.7	15.9	0.00	1.05	2.15	3.37	4.71	4.88	7.89	10.0	15.4

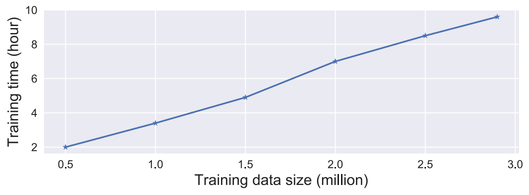


Fig. 8. Training time versus training data size on Harbin dataset.

TABLE VI
THE SENSITIVITY TO K ON HARBIN DATASET.

K	500	1000	1500	2000	2500	3000
recall@n	0.362	0.397	**0.398**	0.386	0.383	0.375
accuracy	0.339	**0.374**	0.373	0.368	0.362	0.353

the Chengdu dataset.

C. Evaluation on route recovery

DeepST is able to score the spatial transition likelihood of any given route, and thus can be used to boost existing route recovery algorithm. As mentioned in Section I, the route recovery algorithm attempts to infer the most likely route between two GPS points in a sparse trajectory. We use the state-of-art sparse route recovery algorithm STRS [2] to illustrate how DeepST can be used to enhance the existing route recovery algorithms. As the travel time t during the two points is often available, the problem is expressed as $\operatorname{argmax}_{\mathbf{r}} \mathbb{P}(\mathbf{r}|t)$ [2]. Using the Bayes rule we have,

$$\operatorname*{argmax}_{\mathbf{r}} \mathbb{P}(t|\mathbf{r})\mathbb{P}(\mathbf{r}), \quad \mathbf{r} \in \{\text{candiate routes}\}.$$

The first term $\mathbb{P}(t|\mathbf{r})$ measures the likelihood of a route $\mathbf{r}$ with an observed travel time t, namely, the temporal inference module; the second term $\mathbb{P}(\mathbf{r})$ scores the spatial transition likelihood of a route $\mathbf{r}$, namely, the spatial inference module. We substitute the spatial inference module $\mathbb{P}(\mathbf{r})$ of STRS with DeepST and the new model is referred to as STRS+. We compare the route recovery accuracy of STRS and STRS+ to see whether DeepST is able to enhance STRS.

To this end, we randomly select 10k trajectories from the test dataset and map these trajectories into the road network as the ground truth with the existing map-matching algorithm [42]. Then we downsample the trajectories with different sampling rates and infer the underlying route with STRS and STRS+. Table V presents the route recovery accuracy (as defined in Equation 9) of STRS and STRS+ over the sampling rate. The accuracy increase of STRS+ over STRS is denoted by δ. The superiority of STRS+ becomes obvious as sampling-rate increases and the trajectory becomes more sparse. Note that the need for route recovery is more critical when the trajectory is sparse. As the δ row shows, as the sampling rate grows up to 9 minutes STRS+ outperforms STRS by 15% in terms of accuracy.

D. Scalability

Figure 8 shows the scalability of DeepST on Harbin dataset. It can be seen easily that the training time grows linearly against the training dataset size (the same observation has been made on Chengdu dataset).

E. Parameter sensitivity study

In this paper we propose to learn K-destination proxies to effectively share statistical strength across trips, rather than treating each destination separately. As reported in Section V-B, the K-destination proxies have an important impact on the performance. Hence, we further analyze the impact of K on the performance in this experiment.

Table Table VI reports the recall@n and accuracy on the Harbin dataset as we vary k. The performance is significantly improved when K increases from 500 to 1000. This is because a small K could not provide sufficient number of proxies to guide the spatial transition of vehicles. Both recall@n and accuracy drop when increasing K to 2000 and beyond. This is because with a large K, no sufficient number of trips is allocated to a proxy, and thus different proxies cannot effectively share the desired statistical strength.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel solution to the problem of predicting the most likely traveling route on the road network between two given locations. For the first time, we unify three key explanatory factors–past traveled route, destination and real-time traffic–for the spatial transition modeling with a deep probabilistic model–DeepST. DeepST achieves this by explaining the generation of next route conditioned on the representations of the three explanatory factors in a principled way. The past traveled route is compressed with a RNN, and thus could account for the long range dependency. To enable effectively sharing statistical strength across trips, we propose an adjoint generative process to learn representations of K-destination proxies rather than learning the destination representations separately. The introduction of the latent variable $\mathbf{c}$ allows us to incorporate the impact of real-time traffic by inferring its posterior distribution. Lastly, an efficient inference algorithm is developed within the VAEs framework to scale DeepST to large-scale datasets. The experiments are conducted on two real-world large-scale trajectory datasets to demonstrate the superiority of DeepST over the competitors on two tasks: the most likely route prediction and route recovery from sparse trajectories. Remarkably, on the Chengdu trajectory dataset, DeepST surpasses the best competing method by almost 50% on the most likely route prediction task and up to 15% on the route recovery task in terms of accuracy.

We consider the following future directions. As the past traveled route $\mathbf{r}_{1:i}$ is generated by the model itself on the prediction stage, which may deviate from the ground truth and lead to accumulated errors, we plan to address it by exploring the techniques such as structure prediction. We also would like to explore the usage of DeepST in popular route discovery [3], anomaly trips detection [33], trajectory representation learning [9], [45], etc.

Acknowledgements

This research was conducted in collaboration with Singapore Telecommunications Limited and supported by the Singapore Government through the Industry Alignment Fund - Industry Collaboration Projects Grant. The authors would like to thank the anonymous reviewers who give thoughtful comments and helpful suggestions.

References

[1] P. Banerjee, S. Ranu, and S. Raghavan, "Inferring uncertain trajectories from partial observations," in *ICDM*, 2014.
[2] H. Wu, J. Mao, W. Sun, B. Zheng, H. Zhang, Z. Chen, and W. Wang, "Probabilistic robust route recovery with spatio-temporal dynamics," in *SIGKDD*, 2016.
[3] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *ICDE*, 2011.
[4] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *ICDE*, 2012.
[5] H. Su, K. Zheng, H. Wang, J. Huang, and X. Zhou, "Calibrating trajectory data for similarity-based analysis," in *SIGMOD*, 2013.
[6] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.
[7] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang, "Modeling trajectories with recurrent neural networks," in *IJCAI*, 2017.
[8] M. Li, A. Ahmed, and A. J. Smola, "Inferring movement trajectories from GPS snippets," in *WSDM*, 2015.
[9] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei, "Deep representation learning for trajectory similarity computation," in *ICDE*, 2018.
[10] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu, "Destination prediction by sub-trajectory synthesis and privacy protection against such prediction," in *ICDE*, 2013.
[11] J. Zhao, J. Xu, R. Zhou, P. Zhao, C. Liu, and F. Zhu, "On prediction of user destination by sub-trajectory understanding: A deep learning based approach," in *CIKM*, 2018.
[12] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *AAAI*, 2016.
[13] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, "Deepmove: Predicting human mobility with attentional recurrent networks," in *WWW*, 2018.
[14] Y. Chen, C. Long, G. Cong, and C. Li, "Context-aware deep model for joint mobility and time prediction," in *WSDM*, 2020.
[15] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, 2003.
[16] D. M. Blei and J. D. Lafferty, "Correlated topic models," in *NIPS*, 2005.
[17] P. K. Gopalan, L. Charlin, and D. Blei, "Content-based recommendations with poisson factorization," in *NIPS*, 2014.
[18] P. Gopalan, J. M. Hofman, and D. M. Blei, "Scalable recommendation with poisson factorization," *CORR*, 2013.
[19] Q. Yuan, G. Cong, and C. Lin, "COM: a generative model for group recommendation," in *KDD*, 2014.
[20] L. Hong, A. Ahmed, S. Gurumurthy, A. J. Smola, and K. Tsioutsiouliklis, "Discovering geographical topics in the twitter stream," in *WWW*, 2012.
[21] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. Magnenat-Thalmann, "Who, where, when and what: discover spatio-temporal topics for twitter users," in *KDD*, 2013.
[22] Y. Cheng, X. Li, Z. Li, S. Jiang, Y. Li, J. Jia, and X. Jiang, "Aircloud: a cloud-based air-quality monitoring system for everyone," in *SenSys 2014*, 2014.
[23] Y. Cheng, X. Li, Z. Li, S. Jiang, and X. Jiang, "Fine-grained air quality monitoring based on gaussian process regression," in *ICONIP*, 2014.
[24] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *JMLR*, 2013.
[25] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *ICLR*, 2013.
[26] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *ICML*, 2014.
[27] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *NIPS*, 2014.
[28] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "DRAW: A recurrent neural network for image generation," in *ICML*, 2015.
[29] S. M. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, K. Kavukcuoglu, and G. E. Hinton, "Attend, infer, repeat: Fast scene understanding with generative models," in *NIPS*, 2016.
[30] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, and O. Vinyals, "Parallel wavenet: Fast high-fidelity speech synthesis," in *ICML*, 2018.
[31] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *WWW*, 2018.
[32] X. Li, G. Cong, A. Sun, and Y. Cheng, "Learning travel time distributions with deep generative model," in *WWW*, 2019.
[33] Y. Liu, K. Zhao, G. Cong, and Z. Bao, "Online anomalous trajectory detection with deep generative sequence modeling," in *ICDE*, 2020.
[34] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei, "Automatic differentiation variational inference," *JMLR*, 2017.
[35] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *CoRR*, 2016.
[36] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *JMLR*, 2017.
[37] A. B. Dieng, C. Wang, J. Gao, and J. W. Paisley, "Topicrnn: A recurrent neural network with long-range semantic dependency," in *ICLR*, 2017.
[38] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, 2014.
[39] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to mcmc for machine learning," *Machine learning*, 2003.
[40] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *CoRR*, 2016.
[41] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *CoRR*, 2016.
[42] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *SIGSPATIAL*, 2009.
[43] [Online]. Available: https://www.openstreetmap.org
[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, 2014.
[45] D. Yao, G. Cong, C. Zhang, and J. Bi, "Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach," in *ICDE*, 2019.