Learning Travel Time Distributions with Deep Generative Model

Xiucheng Li
Nanyang Technological University
xli055@e.ntu.edu.sg

Gao Cong
Nanyang Technological University
gaocong@ntu.edu.sg

Aixin Sun
Nanyang Technological University
axsun@ntu.edu.sg

Yun Cheng
ETH Zürich
chengyu@ethz.ch

ABSTRACT

Travel time estimation of a given route with respect to real-time traffic condition is extremely useful for many applications like route planning. We argue that it is even more useful to estimate the *travel time distribution*, from which we can derive the expected travel time as well as the uncertainty. In this paper, we develop a deep generative model – DeepGTT – to learn the travel time distribution for any route by conditioning on the real-time traffic. DeepGTT interprets the generation of travel time using a three-layer hierarchical probabilistic model. In the first layer, we present two techniques, amortization and spatial smoothness embeddings, to share statistical strength among different road segments; a convolutional neural net based representation learning component is also proposed to capture the dynamically changing real-time traffic condition. In the middle layer, a nonlinear factorization model is developed to generate auxiliary random variable *i.e.,* speed. The introduction of this middle layer separates the statical spatial features from the dynamically changing real-time traffic conditions, allowing us to incorporate the heterogeneous influencing factors into a single model. In the last layer, an attention mechanism based function is proposed to collectively generate the observed travel time. DeepGTT describes the generation process in a reasonable manner, and thus it not only produces more accurate results but also is more efficient. On a real-world large-scale data set, we show that DeepGTT produces substantially better results than state-of-the-art alternatives in two tasks: travel time estimation and route recovery from sparse trajectory data.

CCS CONCEPTS

• **Applied computing** → **Forecasting**; *Mathematics and statistics*;

KEYWORDS

Travel time distribution learning, Deep generative models, VAEs

ACM Reference Format:
Xiucheng Li, Gao Cong, Aixin Sun, and Yun Cheng. 2019. Learning Travel Time Distributions with Deep Generative Model. 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/10.1145/3308558.3313418

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
WWW '19, May 13–17, 2019, San Francisco, CA, USA
© 2019 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313418

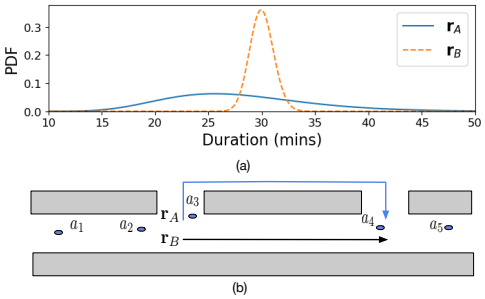


Figure 1: (a) The travel time distributions of two routes–$\mathbf{r}_A$ and $\mathbf{r}_B$, $\mathbb{E}[t_A] = 26$, $\mathbb{E}[t_B] = 30$. (b) Example of route recovery from sparse trajectory.

1 INTRODUCTION

Estimating travel time of a given route on a road network is an essential task for many applications, including route planning, taxi dispatching, and ride-sharing. Many online web apps (*e.g.,* Google Map and Uber) provide such service. Due to its great importance, significant research efforts have been made towards the accurate estimation of travel times [20, 38, 39, 41]. However, existing studies only focus on the estimation of *expected travel time*. We argue that it is much more helpful or even essential to estimate *travel time distribution* (the probability density function) of a given route, based on real-time traffic conditions. We illustrate this point through an example.

Motivating example. Mary has a time budget of 35 minutes to reach airport from home. She queries for the best route, and there are two candidates — route **A** and route **B**, denoted by $\mathbf{r}_A$ and $\mathbf{r}_B$, respectively. The travel time distributions of $\mathbf{r}_A$ and $\mathbf{r}_B$ are depicted in Figure 1a, with the expectation $\mathbb{E}[t_A] = 26$ (minutes), and $\mathbb{E}[t_B] = 30$ (minutes). If the system recommends route solely based on the expected travel time, then $\mathbf{r}_A$ is the best choice. However, as shown in the figure, $\mathbf{r}_A$ has a much larger variance. Therefore, Mary would have a high risk of missing her flight if taking $\mathbf{r}_A$. In contrast, the probability density function of t_B is much more concentrated, and it is much safer to take $\mathbf{r}_B$.

This example shows that it is more sensible to make decision based on the probability distribution of the travel time, instead of mere its expectation. In fact, estimating travel time distribution also addresses the issue of route recovery from the sparse trajectory [42]. Figure 1b shows two possible routes $\mathbf{r}_A, \mathbf{r}_B$ from a_3 to a_4 (due to the low sampling rate) in an observed trajectory $T_a = [a_1, \ldots, a_5]$.

As the sampling points usually carry timestamps, we could get the travel time cost from a_3 to a_4, denoted by t. The problem of inferring the most likely route can be expressed as

$$\underset{\mathbf{r}}{\text{argmax}}\ P(\mathbf{r}|t), \quad \mathbf{r} \in \{\mathbf{r}_A, \mathbf{r}_B\}.$$

Using the Bayes Rule, we have $P(\mathbf{r}|t) \propto P(t|\mathbf{r})P(\mathbf{r})$. That is, we need to access the value of travel time distribution $P(t|\mathbf{r})$ at any $t > 0$.

In this paper, we aim to learn the ***travel time distribution*** $P(t|\mathbf{r})$ for a given route $\mathbf{r}$, with the consideration of real-time traffic conditions. This problem is challenging due to a few reasons. First, data sparsity remains a key challenge. Even though we have access to a large number of trajectories, the trajectories demonstrate a skewed distribution in the space. Roads in central business district are frequently visited, while roads in rural or remote areas are rarely covered. It is reported that about 10% of paths cannot be estimated due to lack of neighbor trajectories [39]. Second, travel time heavily depends on real-time traffic conditions which are dynamic. Almost all existing methods [20, 25, 38, 39, 42] assume that traffic conditions in the same time slot (*e.g.,* 8:00am-9:00am on every Saturday) are temporally-invariant, when estimating travel times. The assumption does not hold because traffic conditions can be influenced by many random variables such as the daily activities of people, road works, and accidents. Considering real-time traffic conditions makes data sparsity issue even worse. Further, travel time is also influenced by spatial characteristics of a road *e.g.,* number of traffic lights, number of lanes, and speed limit. To the best of our knowledge, no existing study has incorporated these heterogeneous influencing factors (both spatial and temporal) into a single model for travel time estimation.

To address the aforementioned challenges simultaneously, we approach travel time distribution learning from deep generative perspective [23, 35]. Specifically, we propose a model named DeepGTT (Deep Generative Travel Time), a three-layer hierarchical probabilistic model [14]. Each layer in DeepGTT captures underlying dependencies among the relevant random variables in a principled way. In the first layer, we present two techniques, spatial smoothness embeddings and amortization [11], to share statistical strength among different road segments to address the data sparsity challenge. A representation ρ_i is learned for each road segment r_i. We further propose a convolutional neural network based representation learning component to address the second challenge. This learning component enables the encoding of dynamically changing real-time traffic conditions into a vector $\mathbf{c}$. In the middle layer, a nonlinear factorization model is developed to combine the road segment representation ρ_i and traffic condition $\mathbf{c}$ to generate the travel speed v_i for road segment r_i. This auxiliary random variable v_i allows us to separate the statical spatial road features from the dynamically changing traffic conditions, so as to address the third challenge. In the end layer, an attention mechanism [4] based function is proposed to generate the entire route travel time t by adaptively aggregating the road segment speed v_i. We derive a variational low bound [7] to train the entire model in an end-to-end fashion.

To the best of our knowledge, this is the first deep generative model developed for travel time distribution learning. A great advantage DeepGTT inherits from probabilistic methods [14], is that it is quite data-efficient [6, 12]; even only trained with a small fraction of data it could produce substantially better results than existing methods trained on much larger datasets. In summary, our contributions are as follows:

- For the first time, we develop a deep generative model, named DeepGTT, to learn travel time distributions. DeepGTT is designed to be data-efficient.The model utilizes spatial smoothness embeddings and amortization to model road segments, and a convolutional neural network based representation learning component to capture real-time traffic conditions.
- The carefully designed hierarchical architecture allows us to separate the dynamically changing traffic conditions from the static spatial features, and thus enables incorporating the heterogeneous influencing factors (both spatial and temporal) into a single model for travel time learning.
- We develop an attention mechanism based function to collectively aggregate road segment speeds to generate the observations. Then, a variational low bound is derived to enable the model to be trained in an end-to-end fashion. As DeepGTT optimizes the complete distribution rather than a single mean value, it could incorporate more variability for more accurate prediction.
- We conduct thorough experiments on a real world large traffic dataset. Experiments show that our model produces substantially better results than state-of-the-art methods in both travel time estimation and route recovery tasks.

The rest of the paper is organized as follows. The related work is discussed in Section 2. The definitions, problem statement and the overview of DeepGTT are presented in Section 3. Section 4 describes the details of our method. The experimental results are presented in Section 5. We conclude the paper and discuss the future work in Section 6.

2 RELATED WORK

We briefly review the related studies on *travel time estimation* and *deep generative models* in this section.

2.1 Travel time estimation

The road segment-based methods first estimate travel time on each individual road segment independently by using the speeds information collected by loop detectors. Then travel time on a given route is estimated by summing up the estimated times of the road segments traversed in the given route. Such methods fail to consider delays caused by traffic lights, and left/right turns, resulting in large estimation errors.

To address the weaknesses of road segment-based methods, route-based methods attempt to estimate the travel time of a route as a whole. There are two main threads for route-based methods, namely, nearest neighbors search [36, 39] and trajectory regression [20, 41, 48]. Nearest neighbors search estimates travel time of a route by averaging travel times of the historical trajectories that have the closed origin and destination with the query route. Trajectory regression methods, on the other hand, treat travel times of the road segments as unknown variables and solve a regression problem using the historical trajectories.

Several works also attempt to estimate the travel time distribution for a given route on road network. Hunter et al. [19] investigate travel time distribution of a path in arterial networks using Gaussian Markov Random Field. Wu et al. [42] propose a model named STRS, that seeks to recover the route from sparse trajectories. STRS comprises of a travel time distribution estimation component and a spatial transition inference component. Its travel time distribution estimation component first learns the mean value of travel time by trajectory regression, and then empirically study the relationship between the variance and mean value to derive the distribution. STRS has been shown to be the state-of-the-art route recovery algorithm. Raghu et al. [13] explore the prediction limits of different methods by estimating the entropy of the travel time distributions.

Almost all existing studies make the assumption that traffic conditions in the same time slot are temporally-invariant. In this paper, we relax this assumption by learning travel time distribution conditioned on the real-time traffic. Wang et al. [40] also attempt to estimate the travel time of a path based on the real-time traffic. They split the query path into multiple sub-paths and search them from the real-time trajectories. However this method seriously suffers from data sparsity issue, even tensor decomposition was used to mitigate it.

Very recently, two deep learning based travel time estimation models–DeepTTE [38] and WDR [41] are proposed. Our method differs from them in three aspects. First, both DeepTTE and WDR only learn to predict the expectation of travel time, whereas our model outputs a distribution. Second, DeepTTE and WDR both adopt the assumption that traffic conditions in a time slot are temporally-invariant (except that DeepTTE takes weather into consideration), while we learn the travel time distribution conditioned on the real-time traffic. Third, both of them employ an RNN to learn the travel time by brute force, which unavoidably leads to a data-hungry model. In contrast, our method is built on deep generative models which allow us to interpret the generation of travel time in a reasonable manner. As a result, our model not only produces more accurate results but also is data-efficient.

Li et al. [27] study the problem of estimating travel time of a *trip* defined by its origin and destination. We highlight that this problem actually is different from ours as we learn the travel time distribution for an entire *route*. Note that, a *trip* can be achieved through multiple routes. In many applications such as trip planning, ride-sharing, travel time estimation between two locations does not help much without knowing the routes to travel.

2.2 Deep Generative Models

Deep generative models or deep probabilistic methods are the hybrids of deep learning and Bayesian inference, which share the great advantages of two worlds. On the one hand, generative models [14] allow us to specify the relationships among random variables to incorporate our prior knowledge in a flexible way. On the other hand, the automatic-differentiation [5] permits us to perform inference on large-scale datasets.

Generative models have been widely adopted in text data mining [8], online recommendation [15, 16, 46], spatial data analytics [18, 45, 47] as well as air quality inference [9, 10]. They are usually trained with sampling-based methods [3, 32] which have the drawback of slow convergence, thus limits their usage to relative small datasets [17]. The emergence of VAE (Variational Auto-Encoders) [23, 35] makes training large-scale generative models possible by marrying Bayesian inference and the automatic-differentiation technique—the core ingredient in deep learning. Since then, deep generative models have been growing rapidly and achieved a range of state-of-the-art results in semi-supervised learning [22], multiple instances scene understanding [12], high quality images and audio synthesis [37, 43], large-scale online recommendation [28]. In contrast, deep generative models have been less explored in spatial-temporal data analysis field. To our knowledge, this is the first deep generative model developed for travel time distribution learning.

Table 1: Notation.

Symbol	Definition
T	Trajectory
$\mathbf{T}$	Trip
$\mathbf{r}$	Route
r_i	The i-th road segment in route $\mathbf{r}$
$\rho_i \in \mathbb{R}^{\lvert\rho_i\rvert}$	The learned representation of r_i
$v_i \in \mathbb{R}$	Travel speed of r_i
C	Real-time traffic indicator
$\mathbf{c} \in \mathbb{R}^{\lvert\mathbf{c}\rvert}$	The learned real-time traffic representation

3 PROBLEM DEFINITION AND DEEPGTT OVERVIEW

We start with definitions of *Road Network*, *Route*, *GPS Trajectory*, and *Trip*. We then formally define our research problem. For ease of reference, the notation used through the paper is given in Table 1.

Definition 3.1. **Road Network.** A road network is represented as a directed graph $G(V, E)$, in which V, E represents the vertices (crossroads) and edges (road segments) respectively.

Definition 3.2. **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 3.3. **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 represents the i-th GPS location and timestamp respectively.

Definition 3.4. **Trip.** A trip $\mathbf{T}$ is a travel along a route $\mathbf{r}$ in 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 $\mathbf{T}$.

In our study, all GPS trajectories are first mapped into the road network to get their underlying routes with a map matching algorithm [30].

The travel time distribution of a route $\mathbf{T}.\mathbf{r}$ highly depends on the real-time traffic condition and it is difficult to precisely define traffic condition. Intuitively, average speed of (sub-)trajectories could be considered as a measurement or sensing of the real-time traffic condition. Therefore, we propose to use all (sub-)trajectories

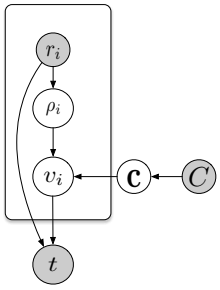


Figure 2: The graphical model of the generative process in which $\rho_i \in \mathbb{R}^{|\rho_i|}$, $v_i \in \mathbb{R}$, $\mathbf{c} \in \mathbb{R}^{|\mathbf{c}|}$ are latent variables and r_i, t are observed scalars.

collected during the time window $[\mathbf{T}.s - \Delta, \mathbf{T}.s)$ as the indicator of real-time traffic condition, denoted by C (or $\mathbf{T}.C$ for the route $\mathbf{T}.\mathbf{r}$). Here, Δ is a specified parameter *e.g.,* 30 minutes. We are now ready to formally define the research problem in this paper

Problem Statement. Given a road network $G(V, E)$ and a historical trajectory dataset $\mathcal{D} = \left\{T^{(m)}\right\}_{m=1}^{M}$, we aim to learn the travel time distribution $P(t|\mathbf{r}, C)$, for a given route $\mathbf{r}$ in the road network, conditioned on the real-time traffic condition indicator C.

DeepGTT Overview. The high-level idea of our proposed solution DeepGTT is that we treat the travel time of a route $\mathbf{r}$ as a random variable t and explain the generation of t using a three-layer hierarchical probabilistic model. The graphical model of the generative process is depicted in Figure 2.

Intuitively, the travel speed of a road segment r_i depends on two important factors: 1) the statical spatial features such as the road types (highway, secondary way), the number of lanes, etc., and 2) the dynamic temporal feature, *i.e.,* the real-time traffic condition. In our model, we learn a road representation ρ_i for road segment r_i to capture its statical spatial features and use $\mathbf{c}$ to model the real-time traffic condition. The generative process of travel time t for route $\mathbf{r} = [r_i]_{i=1}^{n}$ is as follows:

- Draw the traffic condition representation $\mathbf{c} \sim P(\mathbf{c}|C)$.
- For the i-th road segment r_i in the route $\mathbf{r}$,
  - Draw the road segment representation $\rho_i \sim P(\rho_i|r_i)$.
  - Draw the road segment speed $v_i \sim P(v_i|\rho_i, \mathbf{c})$.
- Draw the travel time $t \sim P(t|\mathbf{r}, \mathbf{v})$.

In this model, $\mathbf{v} = [v_i]_{i=1}^{n}$ and $\mathbf{c} \in \mathbb{R}^{|\mathbf{c}|}$, $\rho_i \in \mathbb{R}^{|\rho_i|}$ are fixed-length vectors.

Specifically, in the generative process, we first draw the real-time traffic representation $\mathbf{c} \sim P(\mathbf{c}|C)$. For every road segment r_i in route $\mathbf{r}$ we draw its representation $\rho_i \sim P(\rho_i|r_i)$ which only depends on the statical spatial features of r_i. The introduction of the auxiliary random variable v_i in the middle layer allows us to model both r_i's statical spatial features and the dynamically changing traffic conditions. In other words, probability distribution $P(v_i|\rho_i, \mathbf{c})$ depends on both road representation ρ_i and traffic condition $\mathbf{c}$. In the end, we collectively aggregate the road segment speeds v_i to generate the travel time for the entire route $\mathbf{r}$ by drawing t from $P(t|\mathbf{r}, \mathbf{v})$.

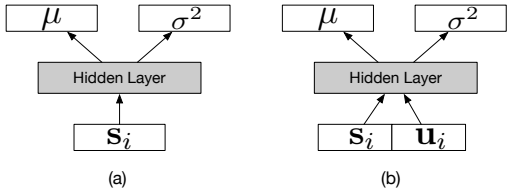


Figure 3: The parameterization of $P(\rho_i|r_i)$.

4 DEEPGTT MODELING

We next detail the modeling of road segment representation ρ_i (Section 4.1), traffic condition $\mathbf{c}$ (Section 4.2), speed (Section 4.3), and the aggregation (Section 4.4). Section 4.5 details the loss function and the learning algorithm. The prediction is discussed in Section 4.6.

4.1 Road Segment Representation ρ_i

As aforementioned, travel speed v_i is dominated by two main factors: the statical spatial features of road segment r_i and the dynamic temporal feature – real-time traffic condition. We model the impact of statical spatial features of r_i with distribution $P(\rho_i|r_i)$, where $\rho_i \in \mathbb{R}^{|\rho_i|}$ is the static road segment representation.

The high-level idea is to encode into ρ_i all spatial features that are indicative to travel speed via modeling $P(\rho_i|r_i)$ conditioned on such spatial features. In our model, we consider the following 5 types of statical spatial features:

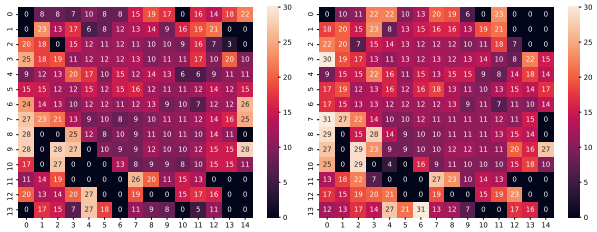
(1) Road types: *e.g.,* primary, secondary, tertiary, residential, etc.;
(2) Number of lanes: how many marked traffic lanes;
(3) Whether it is a one way or not;
(4) The road shape, *e.g.,* straight, curve;
(5) Spatial smoothness, *e.g.,* congestion propagation makes neighboring roads likely to have similar congestion level.

To overcome data sparsity issue, we present two techniques– amortization and spatial smoothness embeddings. These two techniques enable the sharing of statistical strength among different road segments and impose spatial smoothness on modeling $P(\rho_i|r_i)$.

Amortization. The main idea of amortization [11] is to use a function $f(\cdot)$ to map a given observation to a set of parameters that are shared across all data-points. This allows us to share statistical strength among different road segments, in the sense that the knowledge learned from frequently traveled road segments is shared to those rarely traveled. In our setting, the observation corresponds to the (1)-(3) types of categorical features.

For easy exposition, let us use $n^{(\mathrm{t})}$, $n^{(\mathrm{l})}$, $n^{(\mathrm{o})}$ to represent the number of possible values in the first 3 types of categorical features, respectively. Then we use $\phi^{(\mathrm{t})}$, $\phi^{(\mathrm{l})}$, $\phi^{(\mathrm{o})}$ to map a road segment to its corresponding feature value. For example, assuming there are 5 road types in total and r_1 belongs to the third road type, then we have $n^{(\mathrm{t})} = 5$ and $\phi^{(\mathrm{t})}(r_1) = 3$.

We introduce three spatial feature embedding matrices, $S^{(\mathrm{t})}$, $S^{(\mathrm{l})}$, and $S^{(\mathrm{o})}$, with size $n^{(\mathrm{t})} \times d^{(\mathrm{t})}$, $n^{(\mathrm{l})} \times d^{(\mathrm{l})}$, and $n^{(\mathrm{o})} \times d^{(\mathrm{o})}$ respectively, for the three types of spatial categorical features. Here, $d^{(\mathrm{t})}$, $d^{(\mathrm{l})}$, and $d^{(\mathrm{o})}$ are their embedding dimensions. Each row in $S^{(\cdot)}$ corresponds to an attribute embedding, *e.g.,* $S^{(\mathrm{t})}[1]$ (the first row of $S^{(\mathrm{t})}$) is the embedding of the first type of road and $S^{(\mathrm{t})}[\phi^{(\mathrm{t})}(r_i)]$ is the



(a) Real-time traffic condition C at 7am (b) Real-time traffic condition C at 11pm

Figure 4: The real-time traffic condition C at 7am and 11pm, where the cell value indicates the average speed in that local area.

embedding of r_i's road type. For road segment r_i we define $\mathbf{s}_i$ as the concatenation of its spatial feature embeddings, *i.e.,*

$$\mathbf{s}_i = \left[S^{(\mathrm{t})}[\phi^{(\mathrm{t})}(r_i)], S^{(\mathrm{l})}[\phi^{(\mathrm{l})}(r_i)], S^{(\mathrm{o})}[\phi^{(\mathrm{o})}(r_i)]\right] \in \mathbb{R}^{d^{(\mathrm{t})}+d^{(\mathrm{l})}+d^{(\mathrm{o})}}$$

In the spirit of amortization, we model the distribution $P(\rho_i|r_i)$ with a high dimensional Gaussian distribution whose parameters are the outputs of a determined function taking $\mathbf{s}_i$ as input

$$P(\rho_i|r_i) = \mathcal{N}\left(\mu(\mathbf{s}_i), \mathrm{diag}\left(\sigma^2(\mathbf{s}_i)\right)\right) \tag{1}$$

Here, $\mu(\mathbf{s}_i)$ and $\sigma^2(\mathbf{s}_i)$ are parameterized by two MLPs (Multi-Layer Perceptron) with shared input and hidden layers as shown in Figure 3a.

Spatial Smoothness Embeddings. Amortization allows us to share the statistical strength of the categorical features among different road segments. However, it does not consider the impact of road shape and spatial smoothness *i.e.,* the last 2 types of road features. To remedy this, for every r_i we add an additional embedding $\mathbf{u}_i \in \mathbb{R}^{|\mathbf{u}_i|}$. This embedding aims to capture the complex factors (*e.g.,* road shape) that are specific to the road segment. Moreover, to impose spatial smoothness, we run DeepWalk [31] on the road network $G(V, E)$ to initialize the embeddings $\mathbf{u}_i$. Then we concatenate $\mathbf{s}_i$ and $\mathbf{u}_i$ as the input of the MLPs to parameterize the probability distribution $P(\rho_i|r_i)$ as following (also shown in Figure 3b).

$$P(\rho_i|r_i) = \mathcal{N}\left(\mu(\mathbf{s}_i, \mathbf{u}_i), \mathrm{diag}\left(\sigma^2(\mathbf{s}_i, \mathbf{u}_i)\right)\right) \tag{2}$$

The embeddings $S^{(\mathrm{t})}$, $S^{(\mathrm{l})}$, $S^{(\mathrm{o})}$, $\mathbf{u}_i$ and parameters of the two MLPs will be learned from data.

4.2 Real-time Traffic Representation c

In the generative process, we assume there exists a distribution $P(\mathbf{c}|C)$ from which we could draw a random variable $\mathbf{c}$ that captures the real time traffic condition. To realize this, we need to first find a way to construct C which could roughly reveal the real time traffic condition. To this end, we partition the space into cells and estimate the average vehicle speed in a cell by using real time trajectories. Each cell is considered as a sensing of the local traffic congestion level in that area. Figure 4(a) and 4(b) show the constructed C at 7am and 11pm respectively on the same example day, and the cell size is 2km × 2km.

To model $P(\mathbf{c}|C)$, a straightforward approach is to reshape C into a vector to get $\mathbf{c}$ and then build the condition probability table with each entry corresponding to a possible $\mathbf{c}$ value. However, such a table representation has several drawbacks. First, the space cost is prohibitively expensive. Even if we restrict the average speed into an integer with k possible values, there will be $k^{|\mathbf{c}|}$ states for $\mathbf{c}$ in total. Second, the table representation is sensitive to the spatial distribution of vehicles at that time. If there is no sensing vehicle passing the cell at that time, the cell value becomes zero, *e.g.,* $C[0, 13]$, $C[0, 14]$ are 18, 22 at 7am and they become zero at 11pm as shown in Figure 4. This is unreasonable because 7am is typically a peak hour in a day. Third, the table representation could not reveal similarity between two similar states. Even it only one cell value changes in C, the resulting state $\mathbf{c}$ will be considered to be different from the original one.

To overcome the above shortcomings, we propose to use a CNN (Convolutional Neural Net) to extract the abstract features from the rough estimation speed matrix C. The intuition is that similar traffic conditions generate similar travel times for a given path, which in turn results in close loss. We can then propagate such signals through gradients back to the CNN whose parameters will be tuned to being robust to missing values and being capable of producing similar representations for analogous traffic conditions.

$$\begin{aligned} \mathbf{f} &= \mathrm{CNN}(C) \\ P(\mathbf{c}|C) &= \mathcal{N}\left(\mu(\mathbf{f}), \mathrm{diag}\left(\sigma^2(\mathbf{f})\right)\right) \end{aligned} \tag{3}$$

Equation 3 presents the modeling of $P(\mathbf{c}|C)$, where $\mu(\mathbf{f})$, $\sigma^2(\mathbf{f})$ are parameterized by two MLPs with shared hidden layer. The convolutional neural net comprises of three connected convolution blocks and an average pooling layer. Each convolution block consists of three layers: Conv2d → BatchNorm2d → LeakyReLU.

Note that, the choice of making $\mathbf{c}$ being a random variable instead of being a deterministic one allows us to incorporate more variability in modeling complex traffic conditions and to produce a more reliable model.

4.3 Road Segment Speed v_i

Once we get the static road speed representation ρ_i and the dynamic traffic representation $\mathbf{c}$ we could use them to generate road travel speed v_i for r_i. One straightforward method is the linear factorization models [29], *i.e.,* travel speed v_i is interpreted as the linear interaction between ρ_i and $\mathbf{c}$ as

$$v_i = \langle \rho_i, \mathbf{c} \rangle \tag{4}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product operation. However, this simple solution is limited by the linear assumption. We consider that travel speed on a road segment is very likely to be the result of nonlinear interaction between the static spatial features and the dynamic temporal traffic condition. Hence, we propose to model the generation of speed v_i in a nonlinear manner:

$$\begin{aligned} \mathbf{h}_i &= \mathrm{SELU}(W_1\rho_i + W_2\mathbf{c}) \\ \mu_i &= \langle \theta_\mu, \mathbf{h}_i \rangle, \ \sigma_i^2 = \langle \theta_\sigma, \mathbf{h}_i \rangle \\ P(v_i|\rho_i, \mathbf{c}) &= \mathcal{N}\left(\mu_i, \sigma_i^2\right) \end{aligned} \tag{5}$$

In the above equation, $\mathrm{SELU}(\cdot)$ is the self-normalizing nonlinear activation function [24] and W_1, W_2, θ_μ, θ_σ are parameters to be

learned and shared across all road segments. The Gaussian distribution is chosen due to the fact that travel speeds can be perfectly modeled by it [25]. The above generative process generalizes the classical linear factorization models. We actually can recover them by substituting the non-linear activation function SELU with the identity function.

4.4 Aggregating Road Segment Speeds

The Gaussian distribution of speed on road segment r_i naturally leads to Inverse-Gaussian distribution for travel time. The probability density function of Inverse-Gaussian distribution is given as follows:

$$p(t;\lambda,\mu) = \sqrt{\frac{\lambda}{2\pi t^3}} \exp\left\{\frac{-\lambda(t-\mu)^2}{2\mu^2 t}\right\} \tag{6}$$

where $\lambda, \mu > 0$ are parameters *i.e.,* mean μ and variance μ^3/λ. The Inverse-Gaussian distribution describes the first passage time of a Brownian random walk [1], and thus it is suitable for modeling travel time in our problem setting.

However, we cannot simply sum up the travel time distributions of r_i's to form the travel time distribution of the entire route $\mathbf{r}$, because Inverse-Gaussian distribution is not closed under addition. In other words, modeling $t = \sum_i \ell_i / v_i$ leads to an intractable model (here ℓ_i denotes the length of road segment r_i). Similar to [25], in our proposed solution, we consider the average speed of route $\mathbf{r}$, denoted by $v_\mathbf{r}$, as the weighted sum of v_i, *i.e.,*

$$v_\mathbf{r} = \sum_{i=1}^{n} w_i v_i, \quad w_i = \frac{\ell_i}{\sum_{j=1}^{n} \ell_j}.$$

Then we have

$$\mathbb{E}[v_\mathbf{r}] = \mathbb{E}\left[\sum_{i=1}^{n} w_i v_i\right] = \sum_{i=1}^{n} w_i \mu_i \tag{7}$$

$$\mathrm{Var}[v_\mathbf{r}] = \mathrm{Var}\left(\sum_{i=1}^{n} w_i v_i\right) = \sum_{i=1}^{n} w_i^2 \sigma_i^2 \tag{8}$$

Consequently, $P(t|\mathbf{r},\mathbf{v})$ will be a $\mathrm{IG}(\mu,\lambda)$ with

$$\mu = \frac{\sum_i^n \ell_i}{\mathbb{E}[v_\mathbf{r}]}, \quad \lambda = \frac{\mu^3}{(\sum_i^n \ell_i)^2 \mathrm{Var}[v_\mathbf{r}]}. \tag{9}$$

We scale route variance with the square of route length–$\sum_i^n \ell_i$ in Equation 9. This is to take into consideration travel time uncertainty is likely higher for longer route.

Note that Equation 8 holds only when v_i's are mutually independent. This may not be true in reality as speeds of spatially contiguous road segments are highly correlated. On the other hand, this equation offers us the insight that $\mathrm{Var}[v_\mathbf{r}]$ can be represented as a weighted sum of $\mathrm{Var}[v_i]$. Next, we present a better way to assign weights, *i.e.,* an attention mechanism-based method to adaptively assign weights for road segments.

The general idea of attention mechanism [4] is to learn the weights for a collection of inputs under a query vector $\mathbf{q}$. It assigns a relatively larger weight for the input which is more important under query $\mathbf{q}$. In our case, the inputs are the hidden states $\mathbf{h}_i$ in Equation 5, and the query vector is the traffic representation $\mathbf{c}$ because the uncertainty of a road segment is largely determined by the traffic condition. More formally, we have

$$\begin{aligned} a_i &= \langle W\mathbf{c}, \mathbf{h}_i \rangle \\ \mathrm{Var}[v_\mathbf{r}] &= \sum_{i=1}^{n} \mathrm{softmax}(a_i)\sigma_i^2 \end{aligned} \tag{10}$$

where W is the parameter matrix to be learned.

Notably, as we collectively aggregate v_i's to generate t, we actually amortize the delays caused at the crossroads into their adjacent road segments. Also note that we directly use the parameters of $P(v_i|\rho_i,\mathbf{c})$ to form $P(t|\mathbf{r},\mathbf{v})$. As a result, v_i becomes the virtual node in the graphical model (see Figure 2).

4.5 Variational Loss and Learning Algorithm

The generative process implies the following log-likelihood function

$$\begin{aligned} \mathcal{L}(\Theta) = \log \int_\mathbf{c} \int_\rho \mathrm{d}\mathbf{c}\, \mathrm{d}\rho \prod_{m=1}^{M} P(\mathbf{c}_m|C_m) \\ \prod_{i=1}^{n_m} P(\rho_{mi}|r_{mi}) P(v_{mi}|\rho_{mi},\mathbf{c}_m) P(t_m|\mathbf{r}_m,\mathbf{v}_m) \end{aligned}$$

in which Θ denotes all the parameters involved. Unfortunately, the log-likelihood is intractable due to the appearance of the latent variable integrals. Further, the MCMC based methods [3, 32] are not suitable for our case due to their slow convergence speed.

To circumvent this, we make an important observation that travel time t is actually a kind of reflection of the real-time traffic condition indicator C. Therefore, we can view $P(\mathbf{c}|C)$ as the encoder and $P(t|\mathbf{r},\mathbf{c})$ as the decoder of the real-time traffic condition. In the light of variational auto-encoder [23], the log-likelihood for one route could then be approximated as (lower bound)

$$\begin{aligned} \mathcal{L}(\Theta) &\geq \mathbb{E}_{\mathbf{c},\rho}[\log P(t|\mathbf{r},\mathbf{c})] \\ &= \mathbb{E}_{\mathbf{c},\rho}\left[\sum_{i=1}^{n} \log P(\rho_i|r_i) + \log P(v_i|\rho_i,\mathbf{c}) + \log P(t|\mathbf{r},\mathbf{v})\right] \end{aligned}$$

where $\mathbf{c} \sim P(\mathbf{c}|C)$ and $\rho_i \sim P(\rho_i|r_i)$.

This loss can be interpreted as the expected negative reconstruction error. Indeed, if we put a weak prior distribution for $\mathbf{c}$ and ρ_i respectively, we recover the well-known variational loss [7] as

$$\begin{aligned} \mathcal{L}_v(\Theta) = \mathbb{E}_{\mathbf{c},\rho}[\log P(t|\mathbf{r},\mathbf{c})] - KL(P(\mathbf{c}|C)||P(\mathbf{c})) \\ - \sum_{i=1}^{n} KL(P(\rho_i|r_i)||P(\rho_i)) \end{aligned} \tag{11}$$

where $KL(\cdot||\cdot)$ indicates KL divergence. We now can approximate the variational loss using the Monte Carlo method and propagate the gradients backward using the reparameterization trick [23]. In our case, we reparameterize the Gaussian random variable $\mathbf{z}$ ($\mathbf{c}$ and ρ_i) as:

$$\mathbf{z} \sim \mathcal{N}\left(\mathbf{z}|\mu,\sigma^2\right) \Leftrightarrow \mathbf{z} = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(0,1)$$

and calculate the gradient w.r.t the parameters θ as

$$\nabla_\theta \mathbb{E}_\mathbf{z}[f(\mathbf{z})] = \nabla_\theta \mathbb{E}_\epsilon[f(\mu+\sigma\epsilon)] = \mathbb{E}_\epsilon[\nabla_\theta f(\mu+\sigma\epsilon)]$$

where $f(\mathbf{z})$ represents the log-likelihood function of the parameters. In this sense, our model becomes the Variational Auto-Encoder

Algorithm 1: Learning Algorithm

Input: Training dataset: $(\mathbf{r}_m, C_m, t_m)_{m=1}^{M}$
Output: Parameter set: Θ

1 **while** *training is true* **do**
2 $\quad\mathcal{B} \leftarrow$ A random minibatch of data;
3 $\quad\mathbf{c}_k \sim P(\mathbf{c}_k|C_k) \quad \forall C_k \in \mathcal{B}, k = 1, \ldots, |\mathcal{B}|$;
4 $\quad\rho_{ki} \sim P(\rho_{ki}|r_{ki}) \quad \forall \mathbf{r}_k \in \mathcal{B}, \forall r_{ki} \in \mathbf{r}_k, k = 1, \ldots, |\mathcal{B}|$;
5 $\quad\mathcal{L} \leftarrow$ Calculating the variational loss using Equation 11;
6 $\quad$**for** $\theta \in \Theta$ **do**
7 $\quad\quad\mathbf{g}_\theta \leftarrow \nabla_\theta \mathcal{L}_v(\Theta)$;
8 $\quad\quad\theta \leftarrow \theta + \Gamma(\mathbf{g}_\theta)$;
9 **return** Θ;

(VAE) which is fully differentiable. The learning algorithm is presented in Algorithm 1.

We highlight that it is unnecessary to compute the traffic condition indicator C for every trajectory in training dataset (Line 3 in Algorithm 1). Instead, we discretize the temporal dimension into slots and let the trajectories whose start time fall into the same slot share the same C (see Section 5.1).

4.6 Prediction

Given the learned parameters Θ, a query route $\mathbf{r}$ along with its real-time traffic condition indicator C, we run the forward computation (Lines 2–5 in Algorithm 1) to predict the travel time distribution $P(t|\mathbf{r}, C)$.

Specifically, we compute the mean value $\mu(\mathbf{s}_i, \mathbf{u}_i)$ (in Equation 2) for each road segment r_i and $\mu(\mathbf{f})$ (in Equation 3) for traffic condition indicator C. Different from the training stage where we perform sampling operation to draw instances of ρ_i and $\mathbf{c}$, we simply feed these mean values into the subsequent computation to generate v_i (Equation 5) in prediction. Once we obtain v_i for each r_i, we aggregate them to output the parameters μ, λ (Equation 9) of the travel time distribution $P(t|\mathbf{r}, C)$.

5 EXPERIMENTS

We evaluate the effectiveness and scalability of DeepGTT on a real-world taxi dataset, against the state-of-the-art baselines.

5.1 Experimental setup

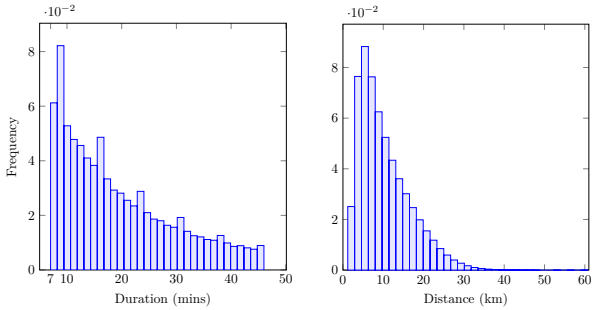
Dataset. The dataset was collected by 13,000 taxis during 28 days in a provincial capital city in China. It contains over 2.9 million trajectories. The sampling time interval between two consecutive points is around 30 seconds, and the map matching accuracy at this sampling rate can be as high as 99% [30]. We acquire the road network data from the Open Street Map [2]. There are 14,497 road segments in total and every road segment has the spatial features: the road type, the number of lanes, one way or not, and road length.

Figure 5 shows the spatial distribution of GPS points. As expected, routes at the central area of the city are frequently traveled whereas routes outside of the city are relatively less traveled by the vehicles. The average travel time is 19.4 minutes and average travel distance

Figure 5: The spatial distribution of the GPS points.

Table 2: Dataset statistics.

Measures	min	max	mean
Duration (mins)	7	46	19.4
Distance (km)	1.2	60	11.4



(a) Dist. of duration (minutes) (b) Dist. of travel distance (km)

Figure 6: The distributions of duration (minutes) and travel distance (km)

is 11.4 kilometers. Table 2 reports main statistics of the trips and Figure 6 plots distributions of trip duration and travel distance.

We use the first 18 days' trajectories as training dataset, the middle 3 days' trajectories as validation set. The remaining 7 days' trajectories are used for testing. The dataset size of training, validation, and testing is 1.7, 0.3, and 0.9 million respectively.

Baseline Methods. We evaluate DeepGTT on two tasks: (i) travel time estimation, and (ii) route recovery from sparse trajectories.

For travel time estimation, we compare DeepGTT with three baseline methods, namely, DeepTTE [38], WDR [41], and MURAT [27]. For route recovery from sparse trajectories, we demonstrate how we enhance the state-of-art route recovery method STRS [42] by simply changing its travel time distribution component to DeepGTT.

- DeepTTE and WDR are two leading travel time estimation models based on Deep Neural Nets. Both models treat road segments as tokens and compress a sequence of tokens (a route) to predict travel time by using RNN.

Table 3: Performance comparison of different methods.

Method	MURAT	WDR	DeepTTE	DeepGTT
RMSE (sec)	510.23	374.24	337.51	**193.04**
MAE (sec)	409.24	275.48	241.37	**141.75**

- MURAT is a multi-task representation learning based travel time estimation model. It only uses the origin-destination information.
- STRS is the state-of-the-art route recovery algorithm, which contains a travel time distribution learning component. The learning component first learns the mean value of travel time by trajectory regression, and then empirically study the relationship between the variance and mean value to derive the distribution.

Parameter settings. Our method[1] is implemented with PyTorch 0.4[2] and Julia 1.0[3], and trained with a Tesla K40 GPU. The platform runs on Ubuntu 14.04 OS with a Genuine Intel CPU.

The embedding sizes of $d^{(t)}$, $d^{(l)}$, $d^{(o)}$, $\mathbf{u}_i$ are set to 64, 32, 16, 200 respectively. The dimensions of the random variables are set as follows: $|\rho_i| = 256$, $|\mathbf{c}| = 400$, $|\mathbf{f}| = 600$, $|\mathbf{h}_i| = 500$. The space is partitioned into a 138×148 matrix C with cell size 200m $\times$ 200m, and $\Delta = 30$ minutes. We discretize the temporal dimension with a slot size 20 (minutes); two trips share the same traffic condition indicator C if their start time fall into the same slot. The batch size $|\mathcal{D}|$ in the training algorithm 1 is 150. The model is optimized by Amsgrad [21, 33] with an initial learning rate 0.001 for 10 epochs, and early stopping is used on validation dataset. We select the best hyperparameters for the baseline methods on valuation dataset.

5.2 Evaluation on travel time estimation

Overall performance. We evaluate all methods on this task by RMSE (Root Mean Square Error) and MAE (Mean Average Error),

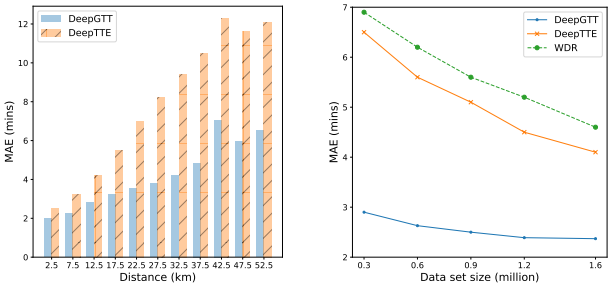
$$\text{RMSE}(\mathbf{t}, \hat{\mathbf{t}}) = \sqrt{\frac{1}{|\mathbf{t}|}\|\mathbf{t} - \hat{\mathbf{t}}\|_2^2}, \quad \text{MAE}(\mathbf{t}, \hat{\mathbf{t}}) = \frac{1}{|\mathbf{t}|}\|\mathbf{t} - \hat{\mathbf{t}}\|_1$$

where $\mathbf{t}$, $\hat{\mathbf{t}}$ denote ground truth, estimated value respectively.

Reported in Table 3, simple model MURAT leads to relatively large errors, and WDR surpasses it by a fair margin. Among all the baselines DeepTTE achieves the best results. Our model DeepGTT outperforms all the baselines by a large margin, for three reasons: 1) DeepGTT interprets the generation of travel time in a logical way instead of learning by brute force; and 2) DeepGTT makes prediction based on the learned real-time traffic representation, rather than assuming that traffic conditions in the same time slot are temporally-invariant; 3) DeepGTT optimizes the full distribution rather than a single mean value, and thus it could incorporate more variability for more accurate prediction.

Remarkably, DeepGTT reduces MAE to 141.75 seconds for the trips with 19.4 minutes average duration which may potentially facilitate many existing web services.

[1]The code is available at https://github.com/boathit/deepgtt
[2]https://pytorch.org
[3]https://julialang.org



(a) Travel distance (b) Training dataset size

Figure 7: (a) MAE changes against travel distance, and over the training dataset size, respectively.

Impact of travel distance. We now study the impact of travel distance on the performance of different models. To this end, we group the trips in test dataset into subgroups by their lengths (in 5KM step), $[0, 5), [5, 10), \ldots, [50, 55)$, and study the performance of different models on these subgroups. Figure 7(a) plots MAEs of DeepTTE and DeepGTT against travel distance (we omit other baselines due to their large errors). Not surprisingly, MAEs increase with the travel distance. Longer trips typically involve more road segments and have larger uncertainty. It is worth noting that the performance difference between DeepTTE and DeepGTT grows with the travel distance. This result suggests that DeepGTT is more trustful for long trip estimation.

Impact of training data size. One of appealing properties of DeepGTT inherited from probabilistic models is that it is quite data efficient [6, 12]. To demonstrate this property, we study the change of MAE with different number of training data points from 0.3 to 1.7 million, reported in Figure 7(b). Observe that even trained with only 0.3 million data points, DeepGTT surpasses baseline methods that are trained with the full training data by a notable margin. This can be explained by the fact that DeepGTT interprets the generation of travel time in a more plausible manner, and thus it could reveal the hidden dependencies among relevant variables and explore training data in a more efficient way. The data efficient property enables DeepGTT to be trained in a much faster speed as well, making it attractive to the online web services that require frequent updates.

5.3 Performance on route recovery

In this subsection, we demonstrate how DeepGTT could enhance the existing leading route recovery algorithm, in which we need to access $P(t|\mathbf{r})$ for any $t > 0$. As described in Section 1, route recovery from sparse trajectories is formulated as

$$\underset{\mathbf{r}}{\text{argmax}}\ P(t|\mathbf{r})P(\mathbf{r}), \quad \mathbf{r} \in \{\text{candiate routes}\}.$$

The first term $P(t|\mathbf{r})$ describes the probability of a route $\mathbf{r}$ for an observed travel time t, namely, the temporal component. The second term $P(\mathbf{r})$ describes the probability from the spatial transition perspective, namely, the spatial component. Actually, these are the

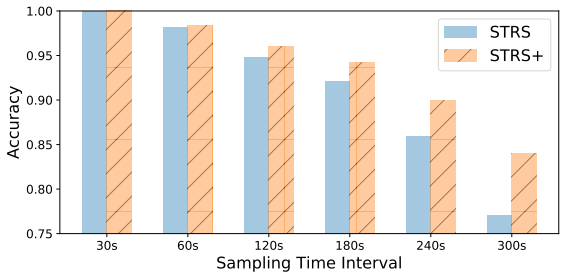


Figure 8: Accuracy comparison.

two main components in the state-of-the-art sparse route recovery algorithm STRS [42].

In this experiment, we replace STRS's temporal component with DeepGTT and refer to the new model as STRS+. The performance of route recovery is evaluated by route recovery accuracy [42]: the ratio of the length of correctly inferred road segments against the maximum value of the length of ground truth $\mathbf{r}_G$ and the length of inferred route $\mathbf{r}_I$.

$$\text{accuracy} = \frac{\text{length}(\mathbf{r}_G \cap \mathbf{r}_I)}{\max(\text{length}(\mathbf{r}_G), \text{length}(\mathbf{r}_I))}$$

We randomly select 10k trajectories from the test dataset and match these trajectories into the map. The matched results are used as ground truth. Then we down-sample these trajectories and recover the routes using STRS, and STRS+, respectively. Figure 8 reports the accuracy of STRS and STRS+, over different sampling time intervals (second). As sampling time interval increases, accuracy of both methods drop as expected. The reason is that larger sampling time interval leads to more possible candidate routes to be inferred between two sample points. Observe that accuracy of STRS+ is always higher than that of STRS. In particular, the difference between their accuracies becomes more evident when the sample time interval is larger. This result shows the superiority of STRS+.

5.4 The impact of real time traffic

In this paper, we relax the assumption that traffic conditions are temporally invariant for the same time slot. Instead, we propose to learn a real-time traffic representation $\mathbf{c}$, and by conditioning on which we generate the travel time. It is natural to ask the effectiveness of this real time traffic representation $\mathbf{c}$. We design two experiments to answer this question.

Experiment 1. We fill the traffic condition indicator C with a constant that is specific to the time slot. In this case, our model is forced to "assume" that traffic conditions are temporally invariant. The RMSE, MAE in travel time estimation increases from 193.04 to 272.45, and from 141.75 to 200.02, respectively. In other words, adopting the assumption that traffic conditions are temporally invariant in DeepGTT leads to a 41% performance drop in terms of MAE. This result implies that an appropriate modeling of $P(\mathbf{c}|C)$ is crucial for accurate travel time estimation.

Experiment 2. As a case study, we randomly select a route $\mathbf{r}_{102}$ with length 21.37km and evaluate the change of its travel time

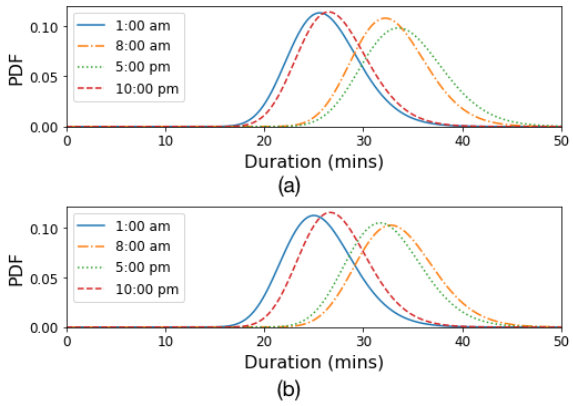


Figure 9: The travel time distributions of route $\mathbf{r}_{102}$ on Jan 03 (a) and Jan 17 (b), both of them are Saturday.

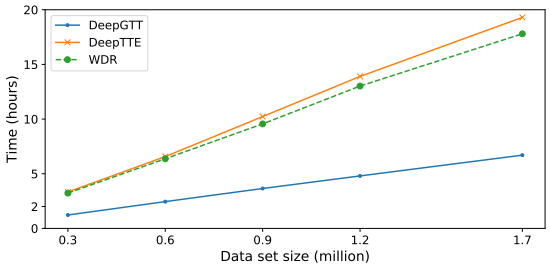


Figure 10: Time cost of all methods grows linearly with the training data size. However, DeepGTT has a much smaller constant term than its competitors.

distributions by taking different real-time traffic condition indicators C's. More specifically, we calculate C by using the (sub-) trajectories in different time slots and feed these C's into the model while keeping the route unchanged. Figure 9a and 9b plot the travel time distributions at different time slots on Jan 03, 2015 and Jan 17, 2015 respectively (both are Saturday). Two points are worth noting: I) The travel time distributions at 1:00am and 10:00pm exhibit smaller expectations and variances. This is reasonable because the traffic conditions at these time slots are less congested, compared with peak hours, *e.g.,* 8:00am and 5:00pm. II) Even both 8:00am and 5:00pm are peak hours, the traffic is more congested at 5:00pm on Jan 03, 2015 whereas it is more congested at 8:00am on Jan 17, 2015. Our observations suggest that traffic conditions in the same time slot are not invariant across the temporal dimension. The assumption made in previous studies does not hold [27, 38, 39, 41, 42].

5.5 Time Cost Study

The training time complexity of DeepGTT grows linearly with the number of trajectories. Figure 10 demonstrates that the training time of different methods vary with the data size. Even though the time cost of both DeepTTE and WDR grows linearly with the size of training data, they have larger constant terms compared with

DeepGTT. This is due to their employment of RNN which is more computationally expensive.

For prediction, DeepGTT can process over 0.9 million trajectories within 81 seconds with a mini-batch size 256. The fast running speed makes it ideal for online deployment.

6 CONCLUSION

For the first time, we develop a deep generative model DeepGTT for travel time distribution learning. We tackle the data sparsity challenge by presenting two techniques, amortization and spatial smoothness embedding, in the lower layer. We relax the assumption that traffic conditions are temporally-invariant by proposing a convolution neural net based real time traffic representation learning and a hierarchical structure. The introduction of an auxiliary random variable v_i in the middle layer enables separating the statical spatial features from the dynamically changing real-time traffic. As a result, we could incorporate these heterogeneous influencing factors into a single model. The further derived variational loss enables the model to be trained in an end-to-end fashion, and makes DeepGTT applicable to large-scale data sets. DeepGTT interprets the generation of travel time in a plausible manner rather than learning by brute force, thus it produces more accurate results and is data-efficient.

Evaluated on a real-world large-scale data set, we first demonstrate the superiority of DeepGTT over existing methods, including two recently proposed deep-neural-net-based approaches, in two tasks: travel time estimation and route recovery. Notably, even trained with a small fraction of data DeepGTT is able to produce substantially better results than the state-of-the-art baselines. We further design experiments to verify that an appropriate modeling of real-time traffic is crucial for accurate travel time distribution prediction, and the assumption made in previous studies that traffic conditions are temporally-invariant does not hold. Finally, we empirically compare the time cost of DeepGTT with the other two deep-neural-net-based approaches, which demonstrates that DeepGTT owns much faster training speed.

In the future work, we would like to enhance the performance of DeepGTT by investigating more rich posterior distribution for the real time traffic representation with the normalizing flows [34]. In the present solution, DeepGTT assumes that the travel times of longer routes tend to show higher variance which may not hold for highways, and we will attempt to address this in the future work. Due to space limitation, we also omit the analysis of the impact of different explanatory factors' dimensionality and cell size, and we would like to give a thorough study in the future version. It is also desirable to explore the usage of DeepGTT in the problems such as ride-sharing, taxi-dispatching, travel time-related trajectory anomaly detection and time-aware trajectory representation learning [26, 44].

ACKNOWLEDGEMENTS

This research was conducted in collaboration with Singapore Telecommunications Limited and partially supported by the Singapore Government through the Industry Alignment Fund - Industry Collaboration Projects Grant. This work was also supported in part by Singapore MOE Tier-2 grant MOE2016-T2-1-137, MOE Tier-1 grant RG31/17, and NSFC under the grant 61772537. The authors would like to thank the anonymous reviewers who give the thoughtful comments and helpful suggestions.

REFERENCES

[1] [n. d.]. https://en.wikipedia.org/wiki/Inverse_Gaussian_distribution
[2] [n. d.]. https://www.openstreetmap.org
[3] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. 2003. An introduction to MCMC for machine learning. *Machine learning* 50, 1-2 (2003), 5–43.
[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014).
[5] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. 2017. Automatic Differentiation in Machine Learning: a Survey. *JMLR* 18 (2017), 153:1–153:43.
[6] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.
[7] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2016. Variational Inference: A Review for Statisticians. *CoRR* abs/1601.00670 (2016).
[8] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *JMLR* 3 (2003), 993–1022.
[9] Yun Cheng, Xiucheng Li, Zhijun Li, Shouxu Jiang, and Xiaofan Jiang. 2014. Fine-Grained Air Quality Monitoring Based on Gaussian Process Regression. In *ICONIP 2014*. 126–134.
[10] Yun Cheng, Xiucheng Li, Zhijun Li, Shouxu Jiang, Yilong Li, Ji Jia, and Xiaofan Jiang. 2014. AirCloud: a cloud-based air-quality monitoring system for everyone. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, SenSys '14, Memphis, Tennessee, USA, November 3-6, 2014*. 251–265.
[11] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. 1995. The helmholtz machine. *Neural computation* (1995), 889–904.
[12] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E. Hinton. 2016. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. In *NIPS 2016, December 5-10, 2016, Barcelona, Spain*. 3225–3233.
[13] Raghu K. Ganti, Mudhakar Srivatsa, and Tarek F. Abdelzaher. 2014. On Limits of Travel Time Predictions: Insights from a New York City Case Study. In *ICDCS 2014, Madrid, Spain, June 30 - July 3, 2014*. 166–175.
[14] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. 1995. *Bayesian data analysis*. Chapman and Hall/CRC.
[15] Prem Gopalan, Jake M Hofman, and David M Blei. 2013. Scalable recommendation with poisson factorization. *CORR* (2013).
[16] Prem K Gopalan, Laurent Charlin, and David Blei. 2014. Content-based recommendations with poisson factorization. In *NIPS 2014*. 3176–3184.
[17] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *JMLR* 14, 1 (2013), 1303–1347.
[18] Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J. Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the twitter stream. In *WWW 2012, Lyon, France, April 16-20, 2012*. 769–778.
[19] Timothy Hunter, Aude Hofleitner, Jack Reilly, Walid Krichene, Jerome Thai, Anastasios Kouvelas, Pieter Abbeel, and Alexandre M. Bayen. 2013. Arriving on time: estimating travel time distributions on large-scale road networks. *CoRR* abs/1302.6617 (2013).
[20] Tsuyoshi Idé and Masashi Sugiyama. 2011. Trajectory Regression on Road Networks. In *AAAI 2011, San Francisco, California, USA, August 7-11, 2011*.
[21] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *ICLR 2015* abs/1412.6980 (2014).
[22] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised Learning with Deep Generative Models. In *NIPS 2014, December 8-13 2014, Montreal, Quebec, Canada*. 3581–3589.
[23] Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *ICLR 2013* abs/1312.6114 (2013).
[24] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-Normalizing Neural Networks. In *NIPS 2017, 4-9 December 2017, Long Beach, CA, USA*. 972–981.
[25] Mu Li, Amr Ahmed, and Alexander J. Smola. 2015. Inferring Movement Trajectories from GPS Snippets. In *WSDM 2015, Shanghai, China, February 2-6, 2015*. 325–334.
[26] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. 2018. Deep Representation Learning for Trajectory Similarity Computation. In *ICDE 2018, Paris, France, April 16-19, 2018*. 617–628.
[27] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task Representation Learning for Travel Time Estimation. In *SIGKDD 2018, London, UK, August, 2018*.
[28] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *WWW 2018, Lyon, France, April 23-27, 2018*. 689–698.

[29] Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. Cambridge, MA.

[30] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *SIGSPATIAL 2009, November 4-6, 2009, Seattle, Washington, USA, Proceedings*. 336–343.

[31] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *SIGKDD 2014, New York, NY, USA - August 24 - 27, 2014*. 701–710.

[32] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Fast collapsed gibbs sampling for latent dirichlet allocation. In *SIGKDD 2008*. ACM, 569–577.

[33] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. *ICLR 2018*.

[34] Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational Inference with Normalizing Flows. In *ICML 2015, Lille, France, 6-11 July 2015*. 1530–1538.

[35] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *ICML 2014, Beijing, China, 21-26 June 2014*. 1278–1286.

[36] Dalia Tiesyte and Christian S. Jensen. 2008. Similarity-based prediction of travel times for vehicles traveling on known routes. In *SIGSPATIAL 2008, November 5-7, 2008, Irvine, California, USA, Proceedings*. 14.

[37] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural Discrete Representation Learning. In *NIPS 2017, 4-9 December 2017, Long Beach, CA, USA*. 6309–6318.

[38] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. In *AAAI 2018, New Orleans, Louisiana, USA, February 2-7, 2018*.

[39] Hongjian Wang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2016. A simple baseline for travel time estimation using large-scale trip data. In *SIGSPATIAL 2016, Burlingame, California, USA, October 31 - November 3, 2016*. 61:1–61:4.

[40] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *SIGKDD 2014, New York, NY, USA - August 24 - 27, 2014*. 25–34.

[41] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to Estimate the Travel Time. In *KDD 2018, London, UK, August 19-23, 2018*. 858–866.

[42] Hao Wu, Jiangyun Mao, Weiwei Sun, Baihua Zheng, Hanyuan Zhang, Ziyang Chen, and Wei Wang. 2016. Probabilistic Robust Route Recovery with Spatio-Temporal Dynamics. In *SIGKDD 2016, San Francisco, CA, USA, August 13-17, 2016*. 1915–1924.

[43] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2016. Attribute2image: Conditional image generation from visual attributes. In *ECCV 2016*. 776–791.

[44] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. Computing Trajectory Similarity in Linear Time: A Generic Seed-Guided Neural Metric Learning Approach. In *ICDE 2019, Macau, China, April 8-12, 2019*.

[45] Zhijun Yin, Liangliang Cao, Jiawei Han, Chengxiang Zhai, and Thomas S. Huang. 2011. Geographical topic discovery and comparison. In *WWW 2011, Hyderabad, India, March 28 - April 1, 2011*. 247–256.

[46] Quan Yuan, Gao Cong, and Chin-Yew Lin. 2014. COM: a generative model for group recommendation. In *KDD '14, New York, NY, USA - August 24 - 27, 2014*. 163–172.

[47] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-Thalmann. 2013. Who, where, when and what: discover spatio-temporal topics for twitter users. In *KDD 2013, Chicago, IL, USA, August 11-14, 2013*. 605–613.

[48] Jiangchuan Zheng and Lionel M. Ni. 2013. Time-Dependent Trajectory Regression on Road Networks via Multi-Task Learning. In *AAAI 2013, July 14-18, 2013, Bellevue, Washington, USA*.