Citywide Estimation of Travel Time Distributions with Bayesian Deep Graph Learning

James J.Q. Yu, *Senior Member, IEEE*

Abstract—Estimation of road link travel time serves a critical role in intelligent transportation operation and management. Due to the uncertainty nature contributed by the volatile traffic, travel time estimates are better described by probability distributions than deterministic models. Existing travel time distribution estimation approaches are mostly based on predefined probability distributions. Other approaches, while relaxing the constraint, fail to utilize the topological information and are data-inefficient. In this paper, we propose a novel Bayesian and geometric deep learning-based approach to estimate the travel time distributions of road links within citywide transportation networks based on vehicular GPS trajectories. Particularly, historical or real-time trajectories are first pre-processed to construct partial travel time maps, which are input into a tailor-made Bayesian graph autoencoder to reconstruct multiple complete travel time maps. We further adopt an auxiliary neural network to facilitate the parameter training of the proposed approach following adversarial training principles. To evaluate the proposed approach, we employ a real-world vehicular trajectory dataset in a series of comprehensive case studies. The empirical results indicate that the proposed approach outperforms the best-performing state-of-the-art baseline with an approximately 10% Kullback-Leibler divergence reduction.

Index Terms—Travel time distribution estimation, deep learning, Bayesian inference, graph convolution, adversarial training.

1 Introduction

Intelligent transportation systems (ITS) are among the essential constituting components of smart cities [1]. As a fundamental and strategic industry in the modern economy, intelligent transportation that aims to alleviate traffic congestion, reduce fossil fuel consumption, and improve transportation efficiency is calling for the attention of both the industry and the research community [2], [3]. Within ITS, travel time estimation plays a critical role in supporting high-level ITS services such as traffic monitoring and vehicle routing [4]. Better estimates improve the performance of these services, which in turn lead to a better experience of transportation users, including but not limited to vehicles and commuters [5]–[7].

Existing industrial solutions and research efforts on estimating travel time mostly aim to accurately provide the travel time of road links within a transportation network based on historical data [8]. Nonetheless, the exact link travel time is influenced by many factors that cannot be easily recorded, e.g., vehicle conditions and driving styles [9]. As a result, these uncertainties may hinder these deterministic travel time estimators from generating reliable estimates [4], [8]. In response to this issue, the community is embracing increasing research attention to the so-called travel time distribution estimation (TTDE), which presumes that the link travel time is a time-varying random variable driven by the heterogeneous and dynamic nature of the traffic [10], [11]. Pioneer TTDE work typically assumes that the travel time follows either Gaussian or log-normal distributions, and empirical results are developed to discuss their respective applicability [12], [13]. However, such parametric models, while easy for theoretical analyses, are not providing sufficient expressibility to represent the travel time dynamics [9], [14]. This concern leads to the current state-of-the-art research on TTDE without the Gaussianity assumption.

In general, there are two mainstream approaches to estimate travel time distributions without the prior assumption. Recently, Prokhorchuk *et al.* proposed to transform the non-Gaussian characteristics of travel time into Gaussians by using a Gaussian copula graphical lasso model in [4]. The main principle is to utilize the covariance matrices in the copula models to exploit the statistical correlation of travel times among adjacent road links. Thanks to its parametric nature, the proposed approach can estimate time distributions using low-resolution and sparse GPS trajectories without relying on previously common assumptions, e.g., Gaussianity and link independency. Subsequently, Duan *et al.* took a further step and devised a kernel density estimation-based model for TTDE [8]. By adopting this non-parametric probability density estimation method, the proposed approach is superior in capturing time-varying travel time dynamics with the change of road conditions.

However, there is a research gap in the TTDE problem. Existing TTDE approaches mainly use sparse vehicular traces [4] or limited traffic detectors [8] to estimate the distributions. On the one hand, these approaches can be applied to practical transportation networks with low-quantity and/or low-quality traffic data support, which expands the applicability. On the other hand, either data down-sampling has to be employed [4] or a high computational burden may be experienced [8] when a massive volume of historical vehicular traces are being processed, which is widely expected and highly plausible with contemporary crowdsourcing technologies. As a result, the data under-utilization issue arises. There exists related route travel time prediction approaches aiming to provide the time estimation of arbitrary routes between two locations in the urban area. These approaches, however, cannot fulfill the requirement of TTDE, as the latter is better utilized to generate optimal routes in real time while the former is more adopted in travel analysis and offline route inference [15]. The research gap advocates effort from the academia and industry to devise novel solutions to TTDE.

The prospects of a data-efficient TTDE approach make deep learning a compelling solution, which makes use of multiple layers of perceptrons to exploit latent characteristics from raw data [16]. To bridge the research gap, we propose a novel citywide TTDE approach based on recent developments of deep learning techniques in this work. We particularly consider the uncertainties that exist in the raw historical and real-time vehicular GPS trajectories and employ the principle of Bayesian deep learning [17] to learn from them. Furthermore, we adopt the graph convolution concept [18] for non-Euclidean space feature extraction, which better describes the transportation network than canonical tensor-based data analyses [7]. The proposed model is built using deep neural network building blocks from data mining and machine learning research to provide a novel solution to TTDE, as well as insights for future TTDE model development. The main efforts of this work are summarized below:

- We propose a novel TTDE approach based on deep learning techniques. The design principles of Bayesian and geometric deep learning are adopted to formulate the new deep neural network.
- We employ an adversarial training method to boost the model training performance of the proposed approach, whose training and online inference schemes are thereby devised.
- We validate the efficacy of the proposed approach on a real-world dataset with more than one billion GPS records. Empirical results demonstrate the significance and sensitivities of the proposed approach.

To our knowledge, this is one of the pioneering works on using deep learning to estimate travel time distributions of road links. The proposed approach is not limited to TTDE and can be further adapted to address a wide range of probabilistic estimation problems in ITS.

The rest of this paper is organized as follows. Section 2 gives a brief overview of the related work and formulates the TTDE problem. Section 3 elaborates on the proposed deep learning-based approach for TTDE with a detailed introduction to its training and inference. Section 4 presents case studies on a real-world dataset to demonstrate the efficacy of the approach. Finally, this work is concluded in Section 5.

2 Preliminaries

2.1 Related Work

Travel time estimation is a well-studied research topic in the literature [4] and is an application of spatio-temporal data mining [19]. Related research can be generally classified into two categories, i.e., deterministic travel time estimation which aims at estimating the mean travel time of road links, and TTDE that provides an estimated probability distribution of the travel time. With the emergence of the big data era, both of the problems now notably rely on the availability of historical travel time data [20]. A variety of data mining-based approaches are proposed to develop mean estimates, including but not limited to linear regression [21], support vector regressor [22], artificial neural network [23], hidden Markov models [24], etc. We refer readers to [4], [7] for a detailed overview of the deterministic travel time estimation literature.

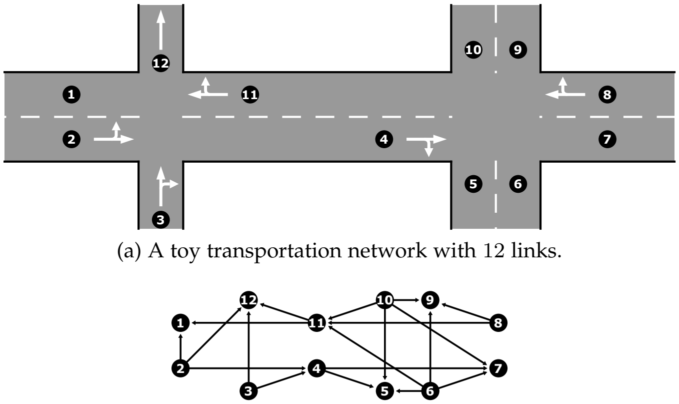
Together with the deterministic estimation problem, TTDE received a growing interest in the past few years [8]. A line of research was conducted with the assumption of Gaussian or log-normal travel time. To name a few, references [12] and [13] analyzed the applicability of Gaussian and log-normal distributions in modeling travel time considering traffic flow conditions and spatial dependencies. Reference [25] further investigated alternative probability distributions as the base parametric model. Reference [26] formulated the travel time using a gaussian mixture model. References [4], [27], [28] combined Gaussian copula with the graphical lasso, Bayesian inference, and section spillover analysis for TTDE, respectively. Reference [29] employed an moving average approach for estimating the deterministic travel time and a distribution estimation algorithm to provide the confidence interval of the previous estimation. On the other hand, attempts on relaxing the Gaussianity assumption lead to TTDE approaches based on a wide spectrum of learning techniques. For instance, reference [30] devised a Markov chain-based heuristic to compute travel time distributions for arterial road links. Reference [31] proposed an origin-destination pair-based Bayesian network for TTDE with the expectation-maximization algorithm to address path uncertainties. Reference [20] developed a deep learning-based generative adversarial network considering trip information maximization to estimate the distributions of both short and long trips. Reference [32] proposed a deep generative model to estimate the travel time distributions for routes comprising multiple road links, which better fits for post-travel analysis and route recovery than pre-travel planning [15]. Reference [33] adopted an encoder-decoder deep neural network to exploit the embeddings of road segments, traffic periodicities, and route trajectories for travel time estimation. The model aims at providing deterministic travel time estimations on routes than road links, which is still not an exact match for TTDE.

To summarize, a number of approaches have been proposed to properly estimate travel time distributions. Nonetheless, less research concentrated on modeling and learning from the data uncertainties, which is significant and includes both epistemic (system) and aleatoric (measurement) ones in this context. Additionally, the spatial data correlation regulated by the transportation network is not taken into account, rendering possible performance degradation. To address these limitations, we propose a new deep learning-based TTDE approach in this work.

2.2 Travel Time Distribution Estimation

We aim to estimate the travel time distributions of road links in a citywide transportation network with sparse real-time GPS data. The transportation network is modeled as a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ is the set of links

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3117986, IEEE Transactions on Knowledge and Data Engineering



(a) A toy transportation network with 12 links.

(b) The corresponding directed graph.

Fig. 1. An example of the graph representation of transportation network.

(road segments) and $\mathcal{E}$ represents the connectivity of links in $\mathcal{V}$, i.e., $(v_i, v_j) \in \mathcal{E}$ if vehicles can directly traverse to link $v_j \in \mathcal{V}$ from $v_i \in \mathcal{V}$. Fig. 1 presents an illustrative example of this graph representation, which is different from the common one of road networks where intersections are represented by graph nodes, and is more expressive when exploiting link correlations [34]. Given ubiquitous and sparse real-time GPS trajectories, the objective of TTDE is to obtain the travel time distribution $\Pr_{v,t}(T)$ of all links $v \in \mathcal{V}$ during time slot t of length Δ. The path travel time distributions can be subsequently developed by accumulating those of the constituting links [4].

The primary challenge of TTDE trifold, each of which has garnered study interest from the data mining community but has not been adequately addressed collectively, particularly for TTDE. First, the aggregated epistemic and aleatoric uncertainty in GPS trajectories make any data mining strategy for discovering latent journey time distributions challenging. Generic data-driven approaches may be disturbed and instead learn the noise patterns, rendering inferior estimations. Developing a tailor-made estimator to fully utilize the massive historical data in the presence of uncertainties is a challenging task. Second, the travel time variations of connecting links are highly correlated so that the estimator needs to exploit how the link travel time depends on others [4], [7]. In a transportation network, such correlations are mainly captured by the adjacency matrix. Failing to use this topological information can potentially lead to degraded performance as notable computational effort is devoted to exploring the structural or spatial data correlation. Third, TTDE considers travel times as random variables and produces their estimates, unlike the well-studied deterministic travel time estimation that develops fixed-point estimations [8]. How to infer accurate and reliable probability distributions is still an open question to the data mining and representation learning communities. There are no de facto solutions for collaboratively addressing these challenges, and earlier efforts on other subjects cannot be easily transferred to TTDE. Taking these challenges into account, we offer a solution for TTDE by utilizing multiple building blocks and design concepts of deep learning.

3 Methodology

To overcome the previous challenges, in this work we propose a novel solution for estimating citywide travel time distributions of road links, named DeepTTDE. This solution is inspired by recent studies and advances in Bayesian deep learning, graph convolution, and adversarial generative models. It attempts to combine the merits of both to address all the previous challenges of TTDE. In this section, we first present an overview of the proposed DeepTTDE and then elaborate on its constituting components. Finally, we discuss the unique DeepTTDE training and inference schemes.

3.1 Overview

Fig. 2 presents the general framework of the proposed DeepTTDE. Three major modules cooperate to derive the travel time distributions, namely, map matching and travel time modeling, travel time distribution estimator, and adversarial training model. Given a collection of GPS trajectories, we first convert the raw geographical coordinates into paths on the graph $\mathcal{G}$, by which it is trivial to calculate the link travel time concerning each trajectory. Subsequently, multiple travel time maps are constructed using the trajectory link travel time data by a unique travel time modeling algorithm. Each of the travel time maps is an individual sample from the latent travel time distributions across the city, and we adopt a Bayesian graph autoencoder to resemble the distributions from these samples. Finally, an auxiliary adversarial training neural network is incorporated to assist the parameter training of the previous graph autoencoder, so that the accuracy of estimated distributions can be enhanced.

Before delving into the details of each module that makes up DeepTTDE, we first explain the rationale behind the techniques used, particularly the Bayesian graph autoencoder structure. All modules have been designed around the requirements of travel time distribution estimators. While deep learning-based techniques have demonstrated state-of-the-art performance in solving a vast number of problems, they generally suffer from a series of limitations including being weak in uncertainty learning, non-Euclidean representations, and being easily compromised by adversarial samples [17], [35]. One may find that the primary challenges of TTDE introduced in Section 2.2 notably overlap with the above limitations of general deep learning techniques.

DeepTTDE addresses the challenges and limitations by adopting the design principles of Bayesian deep learning, geometric deep learning, and adversarial machine learning. Bayesian deep learning is capable of accounting for both epistemic and aleatoric uncertainty, both of which are prevalent in traffic data [17], [36]. In particular, the model places a prior distribution over each of the neural network parameters and the posterior can be subsequently approximated via Bayesian inference. Therefore, model uncertainties are captured by the shape of parameter distributions and statistical uncertainties in the input data are reflected in the posterior. Second, geometric deep learning enables DeepTTDE to represent graph structures effectively during computation, alleviating the significant computational burden associated with deep learning models attempting to extract topological information from traffic data that does

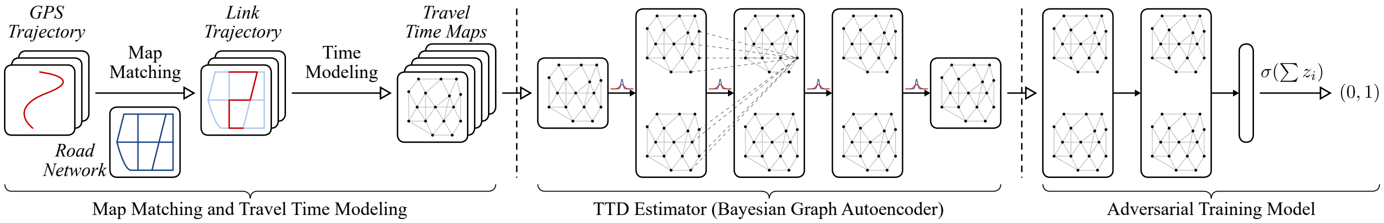


Fig. 2. Overview of DeepTTDE.

not explicitly demonstrate road connectivity [35]. Third, the Bayesian model is intrinsically probabilistic that enables general deep learning models to represent uncertainties and infer distributions, which perfectly meets the requirement of TTDE on estimating probability distributions [36]. With conditional probabilities serving as neural network parameters, Bayesian models estimate distributions by sampling these parameters instead of adding output noise or setting up multiple input scenarios.

3.2 Map Matching and Travel Time Modeling

In DeepTTDE, the main objective of map matching and travel time modeling is to provide adequate travel time maps for the latent information extraction by the subsequent travel time distribution estimator. As the name implies, it has two major steps. Map matching aims at projecting GPS trajectories onto a road network. As this problem has been studied extensively [37]–[39], we adopt an interactive voting-based algorithm implementing [37] to perform the map matching in DeepTTDE. This is also a widely adopted manner in related research, see [4], [7] for some examples.

After developing map-matched paths, travel time modeling attempts to compute the travel times for each link in the paths. This can be trivially achieved by assuming that vehicles follow a constant speed between consecutive trajectory points [4]. Specifically, for an arbitrary trajectory with k GPS locations, let the mapped position of the first trajectory point be p_1 and that of the last trajectory point be p_k. If consecutive positions p_i and p_{i+1} are both on the same link, the sampling time difference between them, denoted by $\Delta_{i\to i+1}$, is accumulated to the travel time of the respective link. Otherwise, the travel time of the "from" link is incremented by $\frac{d_{i\to i+1}}{d_{i\to i+1}+d_{i+1\to i}}\Delta_{i\to i+1}$ and the remaining time is added to that of the "to" link, where $d_{i\to i+1}$ and $d_{i+1\to i}$ are the distance from p_i and p_{i+1} to the intersection of the two links.

Consequently, multiple travel time maps can be constructed based on the calculated link travel times. For each travel time map, we first create a directed transportation graph according to Fig. 1 where each node embeds a road link. Each node v_i is initialized with a tuple $\mathbf{x}_i = \langle t_i, c_i, f_i^1, f_i^2, \cdots \rangle$, where t_i is the instantaneous travel time of v_i, c_i is the sampling time embedding[1], and f_i^j are static features of the respective road link, e.g., speed limit, lane count, link length, etc. Then, we randomly select a time interval of length $15\,\mathrm{min}$ and aggregate all trajectories that fall in the period. The trajectories are permuted and iteratively selected to be included in the travel time map: one is included if none of its constituting links are already covered by others. If a trajectory is included, the t_i and c_i values of all its constituting links are updated with the previously calculated respective travel time and sample time. This process repeats until a pre-defined portion ($70\,\%$ in the case studies to avoid the excessive computational burden, the value is taken from [4]) of all links are covered, or all trajectories have been checked. The produced travel time map is a real sample from the ground truth citywide travel time distributions. Notably, the map matching and the travel time modeling pre-processes are typical for trajectory-based traffic big data analysis. We open the possibility of exploring alternatives to traditional methods as a possible future research direction.

3.3 Travel Time Distribution Estimator

With the previous module that constructs ground truth travel time samples from the raw GPS trajectories, the objective of the subsequent travel time distribution estimator is to learn the travel time distribution dynamics with respect to the time. In DeepTTDE, a Bayesian graph autoencoder is proposed to accomplish the objective thanks to the merits as introduced in Section 3.1. To describe the architecture of the proposed deep learning model, we first briefly present its fundamental operation for non-Euclidean data processing, namely, graph convolution.

Graph convolution, most widely recognized in graph convolutional network [18], is a numerical operation that aggregates nodal information from neighboring nodes within a graph. When employed in neural networks, it aims at extracting local graphical features from the raw data. The operation inherits the concept of convolution filter from canonical convolutional neural networks (CNN) that handle data in Euclidean space. Such filters perform neighborhood mixing on the source data by defining a parametric uniform receptive field, resulting in local information sharing [16]. While receptive fields in CNN are typically rectangular, graph convolution employs the graph connectivity as the filter for neighborhood mixing to cope with the non-Euclidean structure of input graph data. Given the augmented adjacency matrix with a self-loop on each node of a graph, denoted by $\tilde{\mathbf{A}}$, graph convolution takes in features $\mathbf{x}$ to compute the output $\mathbf{y}$ as follows [18]:

$$\mathbf{y} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{x}\mathbf{w}), \qquad (1)$$

where $\tilde{\mathbf{D}} = \mathrm{diag}(\tilde{\mathbf{A}}\mathbf{1})$ is the diagonal degree matrix, $\mathbf{w}$ is the weight parameter of this graph convolution to be learned by back-propagation, and function σ is a nonlinearity—$\mathrm{ReLU}(x) = x^+ = \max(0, x)$ in this work. (1) can be

[1] We use sinusoidal encoding to embed the sampling time. Each date-time is represented by six values, i.e., sine and cosine of the time-of-day, time-of-week, and time-of-year, respectively.

interpreted as a first-order approximation to the localized spectral filter network, which itself is a local approximation to the spectral network that employs the graph Fourier transform to perform convolution in the frequency domain according to the convolution theorem [40], [41]. This connects the graph convolution with the Euclidean space convolution and the merits of the latter also apply to the former.

Note that the graph convolution defined by (1) is based on the premise that the graph Laplacian is symmetric, implying an undirected graph. This notion contradicts the illustration in Fig. 1 where traffic networks are directed. A typical way of resolving the issue is to relax the directed graph into an undirected one, thereby constructing a symmetric Laplacian matrix. Considering that transportation networks are modeled as graphs $\mathcal{G}$, we employ the symmetric matrix of the edge weight matrix as the adjacency $\mathbf{A} = \{a_{ij}\}$ when deriving $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, i.e., $a_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$ or $(v_j, v_i) \in \mathcal{E}$. In the transportation network context, this modification implies that while vehicles can only drive from link v_i to v_j if $(v_i, v_j) \in \mathcal{E}$, the travel time distribution of v_i is also correlated to that of v_j. Overall, this graph convolution operation reflects the physical relationships of the traffic in a transportation network. In practice, the travel times of adjacent links demonstrate a strong correlation. For example, if a link is congested, its merging traffic is also possibly jamming. This characteristic can be captured by the adjacency links in (1) so that neural networks know that the traffic state of one node (road link) can be propagated to its neighboring ones.

With the aforementioned graph convolution, one can already construct a deep neural network to capture the inter-link travel time correlation and perform travel time estimation. However, one significant challenge remains: no uncertainty estimates can be developed. Given a deterministic travel time map as constructed in Section 3.2, (1) can only calculate a deterministic output. This is resolved by Bayesian deep learning. From the probability theory point of view, a neural network with one or multiple layers of graph convolution is a probabilistic model $\Pr(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})$, where $\boldsymbol{\omega}$ is the collection of all weight parameters including $\mathbf{w}$ in (1) and other neural network weights. Without Bayesian inference, the training of fix-valued $\boldsymbol{\omega}$ follows the maximum likelihood estimation with the training set $\mathcal{D} = \{\mathbf{y}^{(i)}|\mathbf{x}^{(i)}\}_i$, i.e., $\boldsymbol{\omega}^* = \arg\max_{\boldsymbol{\omega}} \Pr(\mathcal{D}; \boldsymbol{\omega})$. If the regularization term is adopted to avoid overfitting, the optimal parameters follow the maximum a posteriori probability [42]:

$$\begin{aligned}\boldsymbol{\omega}^* &= \arg\max_{\boldsymbol{\omega}} \log \Pr(\mathcal{D}|\boldsymbol{\omega}) + \log \Pr(\boldsymbol{\omega}) \\ &= \arg\max_{\boldsymbol{\omega}} \log \Pr(\boldsymbol{\omega}|\mathcal{D}).\end{aligned} \tag{2}$$

If we consider the parameters to actually follow the posterior distributions embedded in the training set, the probabilistic model can exploit data uncertainties and estimate distributions with Bayesian inference [17], [43], [44].

Following this principle, the original graph convolution can be modified accordingly, denoted by $\mathbf{y} = f_{\boldsymbol{\omega}}(\mathbf{x})$, to cope with travel time uncertainties using zero-mean Gaussian as the prior distributions over the parameter space $\Pr(\boldsymbol{\omega})$ [45], [46]. Note that this does not impose a Gaussianity assumption on the travel time distributions, therefore overcomes the limit of existing research. According to Bayes' theorem, the posterior distribution can be obtained by

$$\Pr(\boldsymbol{\omega}|\mathcal{D}) = \frac{\Pr(\mathcal{D}|\boldsymbol{\omega}) \Pr(\boldsymbol{\omega})}{\Pr(\mathcal{D})} = \frac{\Pr(\boldsymbol{\omega}) \prod_i l(\mathbf{y}^{(i)}|f_{\boldsymbol{\omega}}(\mathbf{x}^{(i)}))}{\int \Pr(\boldsymbol{\omega}) \prod_i l(\mathbf{y}^{(i)}|f_{\boldsymbol{\omega}}(\mathbf{x}^{(i)}))\, \mathrm{d}\boldsymbol{\omega}}, \tag{3}$$

where $l(\cdot)$ is the likelihood. Nonetheless, the marginal probability $\Pr(\mathcal{D})$ cannot be estimated analytically. To overcome this problem, we employ variational inference to approximate the posterior through a variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{\omega})$ parameterized by $\boldsymbol{\phi}$ [47]. The optimal variational distribution that best approximates $\Pr(\boldsymbol{\omega}|\mathcal{D})$ can be found by minimizing the Kullback-Leibler (KL) divergence with respect to $\boldsymbol{\phi}$ as follows:

$$\mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\omega}) || \Pr(\boldsymbol{\omega}|\mathcal{D})) = \int q_{\boldsymbol{\phi}}(\boldsymbol{\omega}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{\omega})}{\Pr(\boldsymbol{\omega}|\mathcal{D})}\, \mathrm{d}\boldsymbol{\omega}. \tag{4}$$

With (3), (4) can be re-written as

$$\begin{aligned}\mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\omega}) || \Pr(\boldsymbol{\omega}|\mathcal{D})) &= \int q_{\boldsymbol{\phi}}(\boldsymbol{\omega}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{\omega})}{\Pr(\boldsymbol{\omega})}\, \mathrm{d}\boldsymbol{\omega} \\ &\quad - \sum_i \int q_{\boldsymbol{\phi}}(\boldsymbol{\omega}) \log l(\mathbf{y}^{(i)}|f_{\boldsymbol{\omega}}(\mathbf{x}^{(i)}))\, \mathrm{d}\boldsymbol{\omega} \\ &\quad + \log \int \Pr(\boldsymbol{\omega}) \prod_i l(\mathbf{y}^{(i)}|f_{\boldsymbol{\omega}}(\mathbf{x}^{(i)}))\, \mathrm{d}\boldsymbol{\omega}.\end{aligned} \tag{5}$$

To properly minimize this KL divergence, data subsampling [48] and reparameterization [47] are typically used [46]. The key idea is to reduce the total number of samples in calculations involving $\mathcal{D}$ to alleviate the computation burden. Furthermore, the neural network parameters are reparameterized by a deterministic differentiable transformation $\boldsymbol{\phi} = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon}$ where $\boldsymbol{\phi} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}\boldsymbol{\sigma}^\mathsf{T})$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ [47]. Subsequently, (5) can be optimized by stochastic optimizers. The obtained optimal $\boldsymbol{\mu}^*$ and $\boldsymbol{\sigma}^*$ reflect the optimum to the original $\mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\omega}) || \Pr(\boldsymbol{\omega}|\mathcal{D}))$ so that $q_{\boldsymbol{\phi}^*}(\boldsymbol{\omega})$ is a close approximation to $\Pr(\boldsymbol{\omega}|\mathcal{D})$. The estimated distribution can be approximated by

$$\Pr(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \int \Pr(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}) q_{\boldsymbol{\phi}^*}(\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega} = q_{\boldsymbol{\phi}^*}(\mathbf{y}|\mathbf{x}). \tag{6}$$

We refer readers to [47], [48] for related theoretical analyses.

To summarize, Bayesian inference substitutes each neural network parameter (e.g., $\mathbf{w}$ in (1)) with a parameterized Gaussian prior $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}\boldsymbol{\sigma}^\mathsf{T})$. After training the new parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ via KL divergence (5), we can sample K $\boldsymbol{\epsilon}$ values to draw fix-valued parameters from the prior. As each of the parameter samples can generate one deterministic inference, the K inferences naturally form the posterior distribution — travel time distribution in our context.

With the help of Bayesian graph convolution, we construct a Bayesian graph autoencoder for travel time distribution estimation. Fig. 2 presents the layered architecture of the proposed autoencoder that takes one travel time map as the input $\mathbf{x}$ and develops K outputs $\mathbf{y}_k$, each of which represents a possible complete citywide travel time map. At the beginning of this process, the original nodal features $\langle t_i, c_i, f_i^1, f_i^2, \cdots \rangle$ are standardized with z-score normalization, and the missing travel time entries are assigned with zeros. The resulting zero-padded graph feature matrix is input into the autoencoder, which adopts three consecutive Bayesian graph convolutions jointly defined by (1) and (6) for feature and uncertainty learning. In each of the convolutions, the number of nodal features is extended/squeezed to 128, 256, and 128, respectively. The last convolution operation is appended with a tanh-activated Bayesian graph convolution with the number of output features to be one in order to reconstruct the travel time map with missing values generated.

3.4 Adversarial Training Model

In the previous sub-section, we propose a Bayesian graph autoencoder to serve as the travel time distribution estimator. Accepting the travel time maps generated by the preceding map matching and travel time modeling module, the estimator can learn from the raw data and estimate citywide travel time distribution. Nonetheless, empirical results on training the autoencoder show that the optimization of its neural network parameters is subpar.

To accelerate the model training process, in DeepTTDE we employ an auxiliary adversarial training neural network (also known as discriminator [49]) to add a time map authenticity term to the KL divergence training objective. Adversarial training is motivated similarly to the inclusion of the discriminator into generative adversarial networks. While the prior Bayesian graph autoencoder is capable of predicting journey time distributions, it may require a significantly larger model capacity to make estimations and regularize the output based on latent transportation domain information that can aid the model's performance. Rather of that, we use an adversarial training neural network to determine if the generated travel time distributions agree with the extracted traffic dynamics from the ground truths. Case studies to be presented in Section 4.4 also reveal its efficacy empirically.

In particular, a discriminator neural network with three layers of graph convolution is established, each of which has 64, 16, and one output feature, respectively. The neural network accepts any reconstructed travel time map $\mathbf{y}_k$ derived by the preceding Bayesian graph autoencoder, and outputs a feature graph $\mathbf{z}_k$. Finally, the features are aggregated with summation and a sigmoid function is employed to map the aggregated feature to $(0, 1)$. This output reflects whether the discriminator considers the respective $\mathbf{y}_k$ "authentic" or not.

The objective of this discriminator is to help the estimator generate travel time maps indistinguishable from their real counterparts so that the estimator training is accelerated by adding this additional authenticity information to its fitness objective. The estimator and discriminator form a two-player minimax game with the following objective:

$$\begin{aligned}\min_{\boldsymbol\phi}\max_{\boldsymbol\Phi} V =& \mathbb{E}_{\Pr(\mathbf{x})}[\log(1-h_{\boldsymbol\Phi}(f_{\boldsymbol\omega}(\mathbf{x})))] \\ &+ \mathbb{E}_{\Pr(\mathbf{x})}[\log h_{\boldsymbol\Phi}(\mathbf{x})] + \mathrm{KL}(q_{\boldsymbol\phi}(\boldsymbol\omega)\|\Pr(\boldsymbol\omega|\mathcal{D})),\end{aligned} \tag{7}$$

where $h_{\boldsymbol\Phi}(\mathbf{z})$ describes the discriminator parameterized by $\boldsymbol\Phi$. At the beginning of training, a randomly initialized $\boldsymbol\phi$ makes the discriminator easily reject the generated travel time map $f_{\boldsymbol\omega}(\mathbf{x})$, rendering a large $\mathbb{E}_{\Pr(\mathbf{x})}[\log(1 - h_{\boldsymbol\Phi}(f_{\boldsymbol\omega}(\mathbf{x})))]$ value. The training process then tries to adjust $\boldsymbol\phi$ to generate realistic travel time maps to compromise the discriminator and resemble the real ones. At the same time, $\boldsymbol\Phi$ is also consistently optimized to improve the discriminator's capability of identifying generated maps. This process repeats until the estimator parameters are good enough to circumvent the discriminator by learning the latent travel time distribution in the raw data. Interested readers can refer to [49]–[51] for more detailed analyses of the adversarial training mechanism.

Algorithm 1: Data Augmentation for DeepTTDE

```
Data: ${\mathbf{x}^{(i)}}_{i=1}^{N}$, $M$
Result: $\mathcal{X} = \{\mathbf{x}^{(i),m}, \mathbf{x}^{(i)}\}_{i=1,m=1}^{N\times M}$
initialize the augmented travel time map set $\mathcal{X} = \emptyset$;
for $\{i, m\} \in \{1, 2, \cdots, N\} \times \{1, 2, \cdots, M\}$ do
    generate a random $\mathcal{V}^{(i)*,m}$ as a subset of $\mathcal{V}^{(i)+}$;
    $\mathbf{x}^{(i),m} \leftarrow \mathbf{x}^{(i)}$;
    for $j \in \mathcal{V}^{(i)*,m}$ do
        $\mathbf{x}_j^{(i),m} \leftarrow 0$;
    end
    append $\mathbf{x}^{(i),m}$ to $\mathcal{X}$;
end
```

3.5 DeepTTDE Training and Inference

In the previous sub-sections, we introduced the three major modules in DeepTTDE. Before adopting the model for online travel time distribution inference, one needs to first adjust the neural network parameter values to fully capture the latent raw data characteristics. While the Bayesian graph autoencoder can be trained with any stochastic gradient descent variant considering (1) and (5), it remains unclear how the input travel time maps can be used as the input and target output of the neural network. Therefore, we propose a training scheme tailored for DeepTTDE to account for the multi-sample nature of its travel time modeling and Bayesian deep learning.

Given a collection of raw GPS trajectories, the map matching and travel time modeling module is capable of constructing multiple travel time maps, each of which comprises the deterministic incomplete citywide travel time data of an arbitrary $15\,\mathrm{min}$ time interval (Section 3.2). For any travel time map $\mathbf{x}^{(i)}$ consisting of road link data $\mathbf{x}_j^{(i)}$ for $v_j \in \mathcal{V}$, we use sets $\mathcal{V}^{(i)+}$ and $\mathcal{V}^{(i)-}$ to aggregate the links that are covered and not covered by any raw trajectories in the travel time map, respectively. We augment the map by manually removing some of the available link data from $\mathcal{V}^{(i)+}$ and construct multiple new training maps. Algorithm 1 presents the pseudo-code for the augmentation process. In particular, M random subsets $\mathcal{V}^{(i)*,m} \subsetneq \mathcal{V}^{(i)+}$ are independently generated. Then we duplicate $\mathbf{x}^{(i)}$ to construct M identical maps $\mathbf{x}^{(i),m}$. Within each $\mathbf{x}^{(i),m}$, the available features of links in $\mathcal{V}^{(i)*,m}$ are removed. Consequently, $N\times M$ augmented travel time maps are developed based on N original maps. The objective of the subsequent travel time distribution estimator is to recover $\mathbf{x}^{(i)}$ based on the partial $\mathbf{x}^{(i),m}$. Aggregating both maps forms the training data set $\mathcal{X}$.

The neural network parameters in the Bayesian graph autoencoder and adversarial training model are adjusted iteratively. In each iteration, the autoencoder takes an $\mathbf{x}^{(i),m}$ and infers K different $\hat{\mathbf{x}}^{(i),m}$ to estimate $\mathbf{x}^{(i)}$, denoted by $\hat{\mathbf{x}}_k^{(i),m}$. Considering (7), the objective function of the autoencoder is designed as follows:

$$\min_{\boldsymbol{\phi}} \mathbb{E}_{\Pr(\mathbf{x})}[\log(1-h_{\boldsymbol{\Phi}}(\hat{\mathbf{x}}_k^{(i),m}))] + \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\omega})||\Pr(\boldsymbol{\omega}|\mathcal{D})), \tag{8}$$

where $\hat{\mathbf{x}}_k^{(i),m}$ is identical to the $\mathbf{y}^{(i)}$ when computing the KL divergence, i.e., (5). Subsequently, both $\mathbf{x}^{(i)}$ and $\hat{\mathbf{x}}_k^{(i),m}$ are input into the discriminator to evaluate the following objective function also derived from (7):

$$\min_{\boldsymbol{\Phi}} \mathbb{E}_{\Pr(\mathbf{x})}[\log(1-h_{\boldsymbol{\Phi}}(\hat{\mathbf{x}}_k^{(i),m}))] + \mathbb{E}_{\Pr(\mathbf{x})}[\log h_{\boldsymbol{\Phi}}(\mathbf{x}^{(i)})]. \tag{9}$$

For every 32 iterations, the computed loss values of both the autoencoder and the discriminator are utilized to adjust all neural network parameters using the Adam optimizer [52].

With trained DeepTTDE parameters, we can use the Bayesian graph autoencoder to infer the real-time citywide travel time distribution. Upon constructing a partial map with available information as introduced in Section 3.2, DeepTTDE uses the autoencoder to produce P complete travel time maps for prediction. Note that this hyperparameter is different from the previous K, which serves as the number of network parameter samples only during training. For each road link in the transportation network, we consider that each of the deterministic travel time inference in all of the P maps is a random sample from the travel time distribution, which shall approximate the ground truth given by the data. When evaluating the inferred distribution, we use the formulae of KL divergence and Hellinger distance for two discrete distributions $Q = \sum_r Q(r)$ and $R = \sum_r R(r)$:

$$\mathrm{KL}(Q||R) = \sum_r Q(r) \log \frac{Q(r)}{R(r)}, \tag{10a}$$

$$\mathrm{Hellinger}(Q,R) = \frac{1}{\sqrt{2}} \Big(\sum_r (Q(r)^{\frac{1}{2}} - R(r)^{\frac{1}{2}})^2 \Big)^{\frac{1}{2}}, \tag{10b}$$

where r is the index of histogram bins. Following the analysis in [4], we set the default number of histogram bins to 11 when calculating these two formulations.

4 Case Studies

In this section, we conduct a series of comprehensive case studies on a real-world dataset to evaluate the performance of DeepTTDE. Specifically, we first briefly introduce the dataset and simulation configurations employed. Then, the comparative study on the preciseness of estimated travel time distributions are investigated, and the impact of GPS trajectory sparsity is examined. Subsequently, an ablation test is carried out to validate the necessity of DeepTTDE constituting modules. Finally, a hyperparameter test demonstrates the impact of hyperparameter selection on model performance.

4.1 Data and Configuration

In this work, we employ the KDD CUP 2020 dataset[2] provided by DiDi Chuxing for thorough case studies. In particular, the dataset includes GPS trajectories of ride-sharing vehicles in the city Chengdu, China from Nov. 1st, 2016 to Nov. 30th, 2016. It provides approximately 1097 million GPS records with $2\,\mathrm{s}$ to $4\,\mathrm{s}$ sampling intervals from vehicles traversing the urban city. The transportation network topology is obtained from OpenStreetMap, and the map matching is conducted as introduced in Section 3.2.

To develop the ground truth traffic speed distributions, we first split the complete dataset into 2880 time periods, each of which contains all vehicular traces of $15\,\mathrm{min}$ in the day. In case that a trajectory spans over multiple periods, it is included in all of them. Subsequently, we adopt the standard travel time modeling technique introduced in Section 3.2 to develop the travel time of each trajectory with respect to every comprising road links. Since multiple trajectories may pass through the same road link within an arbitrary time period, all the respective travel time values are aggregated and collectively serve as the ground truth. Particular links that are not covered by any trajectories within the period is regarded as an element of $\mathcal{V}^{(i)-}$ and are not involved in the performance evaluation. We compare the TTDE results of the proposed DeepTTDE with these ground truth traffic speed distributions.

When employing the proposed approach for TTDE, real-time vehicular trajectories may only provide the instantaneous travel time of a small portion of all roads. To emulate such real-world cases, we use the methodology proposed in Section 3.2 to construct $N = 75$ base travel time maps for each of the time periods, rendering a total of 216 000 time maps. For each base travel time map, we randomly construct $M = 5$ maps by retaining the travel time data of $\beta\,\%$ links in $\mathcal{V}^{(i)+}$ according to Section 3.5. As a result, there are 1 080 000 travel time maps available for TTDE training and evaluation. For cross-validation, we randomly select 3600 time periods (540 000 samples) for training, 1800 (270 000 samples) for early-stopping validation, and the remaining 1800 (270 000 samples) for testing. The validation samples are evaluated after each neural network training epoch, and the whole training is terminated if the loss on this validation set does not decrease in the past three epochs. When evaluating the quality of estimated travel distributions, the KL divergence and Hellinger distance (Hlg. for short in the following tables) of all road links in $\mathcal{V}^{(i)+}$ are calculated and averaged with respect to the ground truth. Unless otherwise stated, $\beta = 30\%$, $K = 32$, and $P = 512$. All simulations are performed on computing servers with two Intel Xeon E5 CPUs and 128GB RAM. nVidia RTX 2080Ti GPUs are utilized for neural network computing acceleration.

4.2 Travel Time Distribution Estimation

We first study the deviation between estimated travel time distributions and the ground truth ones, which is among the most important metric in evaluating TTDE methods. In particular, the following methods serve as the baselines in this empirical study:

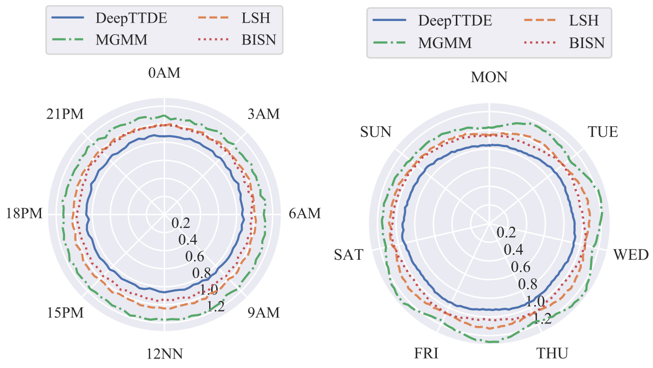
[2] Available at https://outreach.didichuxing.com/appEn-vue/KDD_CUP_2020.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3117986, IEEE Transactions on Knowledge and Data Engineering

TABLE 1
Performance of DeepTTDE and State-of-the-art TTDE Methods on KDD CUP 2020 Dataset

Time	Metric	DeepTTDE	Hist-1D	Hist-1W	Gaussian	Log-normal	LSH	MGMM	BISN
All	KL	0.86 ± 0.12	1.78 ± 0.30	1.63 ± 0.35	1.56 ± 0.21	1.28 ± 0.24	0.99 ± 0.15	1.08 ± 0.13	0.95 ± 0.14
	Hlg.	0.49 ± 0.07	1.01 ± 0.19	0.94 ± 0.14	0.94 ± 0.13	0.86 ± 0.16	0.59 ± 0.10	0.63 ± 0.10	0.55 ± 0.08
Congestion	KL	0.84 ± 0.12	1.73 ± 0.28	1.58 ± 0.36	1.75 ± 0.33	1.30 ± 0.25	1.04 ± 0.17	1.11 ± 0.15	0.95 ± 0.15
	Hlg.	0.48 ± 0.07	1.00 ± 0.19	0.94 ± 0.13	1.00 ± 0.12	0.86 ± 0.17	0.59 ± 0.10	0.68 ± 0.13	0.55 ± 0.09
Free Flow	KL	0.87 ± 0.13	1.82 ± 0.31	1.65 ± 0.33	1.47 ± 0.21	1.27 ± 0.22	0.97 ± 0.15	1.07 ± 0.12	0.96 ± 0.12
	Hlg.	0.49 ± 0.07	1.03 ± 0.18	0.94 ± 0.17	0.87 ± 0.15	0.86 ± 0.14	0.58 ± 0.10	0.59 ± 0.08	0.55 ± 0.08

- **Historical** asserts that link travel time distributions are static over time. The distribution of the same time period in the previous day (Hist-1D) or the previous week (Hist-1W) is used as the current travel time distribution.
- **Gaussian** [12] and **Log-normal** [13] assume that travel times follow either Gaussian or log-normal distribution. The two methods use historical data to fit the respective probabilistic model.
- **Latent-Segmentation, Hazard-Based** (LSH) [10] method leverages road condition-influencing factors to construct hazard-based models for TTDE. We use only the traffic speed to set up the hazard-based model due to the lack of meteorological data in GPS records.
- **Modified Gaussian Mixture Model** (MGMM) [26] employs a Gaussian mixture model for TTDE by considering signalized intersection delays.
- **Bayesian Inference of Sparse Networks** (BISN) [4] combines Gaussian copulas and Bayesian inference for TTDE. This approach is among the state-of-the-art.



(a) Daily KL divergence.

(b) Weekly KL divergence.

Fig. 3. KL divergence performance of DeepTTDE and best-performing baselines.

We notice a recent work [8] that leverages a kernel density estimator (KDE) to estimate travel time distributions. As this work focuses on using data from sparsely-deployed static traffic detectors for TTDE, we do not include the method as a baseline. According to the results, KDE develops results comparable to MGMM with approximately 0.05 KL divergence improvement [8, Table VI]. Therefore, comparing with MGMM can give a qualitative reference to its relative performance over the proposed DeepTTDE.

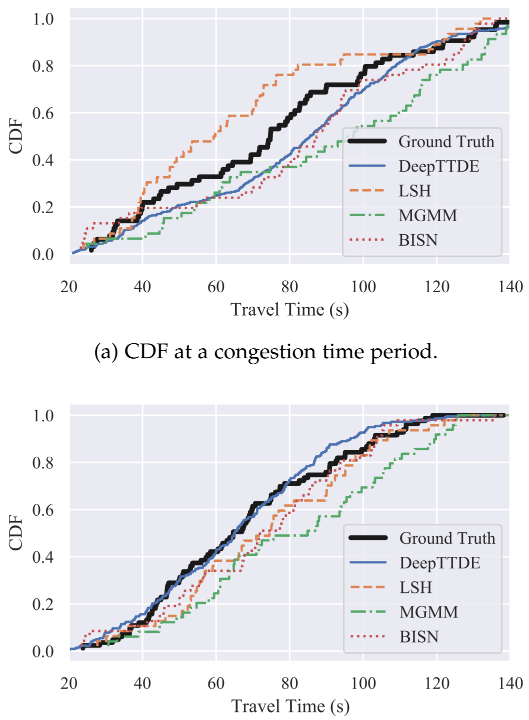
Table 1 presents the performance comparison of DeepTTDE and the above baseline methods on the KDD CUP 2020 dataset. In the table, we summarize the mean and standard deviation of KL divergence and Hellinger distance of all links in all testing time periods. Additionally, we further show the performance of compared methods on the congestion time periods (7:30am–9:30am and 5pm–7pm) and remaining free flow ones. From the comparison, we can observe that the proposed DeepTTDE consistently outperforms all baselines in all time periods. On average, DeepTTDE achieves a satisfactory average KL divergence at 0.86 on all time periods, while the best-performing baseline scores 0.95. The approximately 10% reduction can be credited to the unique Bayesian graph convolution operator devised in DeepTTDE which helps the model to better learn from the travel time uncertainties. This advocates the adoption of Bayesian and geometric deep learning principles in future data-driven TTDE related research. Furthermore, DeepTTDE is capable of generating precise travel time distributions in both congestion and free flow time periods with negligible performance deviation. While similar results can also be observed on BISN, other statistical approaches generally favor one type of time periods over the other, rendering less robust performance.

To better illustrate the model performance regarding time-of-day and time-of-week, Fig. 3 presents the KL divergence metric of DeepTTDE and three best-performing baseline methods with respect to time. The plot also suggests that the proposed DeepTTDE outperforms baselines at all investigated time periods. While not obvious, all TTDE methods demonstrate a periodic pattern for each day within a week, which accords with the ground truth where the congestion status of road links is also generally periodic. A deeper look into the raw results indicate a similar conclusion as previously given by [4], namely, Sunday generally has the largest estimation error for all four methods in Fig. 3. In the meantime, we did not observe the unusual KL divergence spike during the congestion hours. This discrepancy may be contributed by the different characteristics of the datasets employed in this work and [4]. Fig. 4 presents the empirical and estimated cumulative distribution functions (CDFs) for an arbitrary road link during a congested and a free flow time period. It can be observed that the CDF curve of DeepTTDE is generally the closest one to the empirical CDF.

Furthermore, the baseline approaches presented and compared in Table 1 are designed primarily for TTDE, while DeepTTDE adopts graph learning-based probability estimation techniques to handle this problem. We also compare the proposed approach with existing state-of-the-art probabilistic deep learning models and preliminary graph learning-

TABLE 2
Performance of DeepTTDE and Probabilistic Prediction Methods on KDD CUP 2020 Dataset

Time	Metric	DeepTTDE	DeepGTT	RAT-SPN	GCN-Gauss	GCN-Logn
All	KL	0.86 ± 0.12	1.32 ± 0.17	1.85 ± 0.26	1.62 ± 0.24	1.30 ± 0.23
	Hlg.	0.49 ± 0.07	0.88 ± 0.11	1.07 ± 0.15	0.91 ± 0.13	0.87 ± 0.15
Congestion	KL	0.84 ± 0.12	1.35 ± 0.17	1.96 ± 0.29	1.73 ± 0.27	1.29 ± 0.21
	Hlg.	0.48 ± 0.07	0.90 ± 0.10	1.11 ± 0.15	0.99 ± 0.14	0.85 ± 0.16
Free Flow	KL	0.87 ± 0.13	1.31 ± 0.18	1.79 ± 0.25	1.52 ± 0.23	1.30 ± 0.23
	Hlg.	0.49 ± 0.07	0.88 ± 0.11	1.03 ± 0.14	0.89 ± 0.13	0.88 ± 0.15



(a) CDF at a congestion time period.

(b) CDF at a free flow time period.

Fig. 4. KL divergence performance of DeepTTDE and best-performing baselines.

based distribution estimators to demonstrate the efficacy of DeepTTDE. Particularly, the following additional baselines with minuscule non-algorithmic changes are included in the comparison as summarized in Table 2:

- **Deep Generative Travel Time** (GTT) [32] adopts a three-layer deep generative model to learn the travel time distribution for trip routes. For a fair comparison, we consider each route comprises one road link, and the objective is to estimate its travel time distribution.
- **Random and Tensorized Sum-product Network** (RAT-SPN) [53] employs sum-product networks for deep learning by generating unspecialized random structures for the network and use deep learning optimizer to train a tensorized view of the network.
- **GCN-Gauss** employs a stack of three graph convolution layers [18] to estimate the mean and variance of road travel time distributions, considering them as Gaussian.
- **GCN-Logn** is similar to GCN-Gaussian with Log-normal distributions as the prior.

From the comparison, it is clear that none of the non-TTDE probabilistic prediction methods perform on par with or outperform DeepTTDE, indicating the necessity of developing tailor-made TTDE approaches. Albeit DeepGTT can produce reasonably steady estimations when compared to other baselines, the resulting accuracy is worse than both DeepTTDE and the better-performing Bayesian inference-based BISN. This finding can be attributed to the fact that DeepGTT was originally designed for route estimation rather than road link estimation, where the two problems may have significantly different properties. RAT-SPN also faces the same challenge, rendering it inferior than baselines in Table 1. Lastly, it seems that directly estimates the distribution parameters using graph learning with an either Gaussian or Log-normal prior does not cohere with the ground truth data distribution while the posterior estimation of DeepTTDE is relatively more intrinsic in this particular travel time estimation scenario.

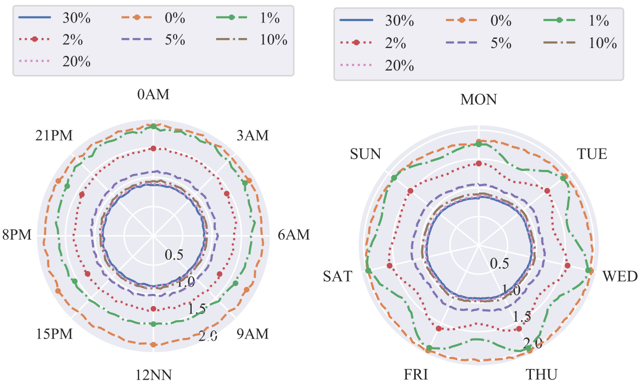
As travel time estimation is a real-time service of intelligent transportation systems, the computation time is of critical importance in determining whether a TTDE method can be applied online. Both the offline training and online inference of DeepTTDE require intensive algebraic calculations. With the computing platform adopted in the case study, one complete training of DeepTTDE takes approximately $26\,\mathrm{h}$ with the enormous $540\,000$ training and $270\,000$ validation sets. Considering that significant transportation network topology or traffic pattern changes are typically infrequent, DeepTTDE can be re-trained with the new data in a sufficiently short time. Furthermore, a well-trained DeepTTDE takes less than $3\,\mathrm{s}$ to generate 512 travel time maps for constructing travel time distributions, which is minuscule compared with the $15\,\mathrm{min}$ time period length. Therefore, we can conclude that DeepTTDE can be adopted in online TTDE services. Noted that the model training time heavily depends on the optimizer parameters. We employ the default configuration of Adam in this work, and training efficiency improvement is plausible with better parameter settings.

4.3 GPS Trajectory Sparsity

In the previous case study, we set $\beta = 30\%$ in both offline training and online inference. One may note that while it is possible to create travel time maps with 30% available data for time periods in the past, real-time GPS trajectories may not be so abundant that 30% of all road links are covered. In this test, we investigate the impact of GPS trajectory sparsity on the DeepTTDE performance. Specifically, the trained DeepTTDE with $\beta = 30\%$ is adopted as the esti-

TABLE 3
Performance of DeepTTDE Architectural Variants on KDD CUP 2020 Dataset

Time Periods	Metric	DeepTTDE	DeepTTDE-AEGC	DeepTTDE-ATGC	DeepTTDE-AT	DeepTTDE-GC-AT	BISN
All	KL	0.86 ± 0.12	1.17 ± 0.18	0.91 ± 0.13	1.25 ± 0.31	1.57 ± 0.33	0.95 ± 0.14
	Hlg.	0.49 ± 0.07	0.73 ± 0.14	0.52 ± 0.07	0.88 ± 0.22	0.94 ± 0.20	0.55 ± 0.08
Congestion	KL	0.84 ± 0.12	1.15 ± 0.17	0.91 ± 0.12	1.22 ± 0.32	1.49 ± 0.31	0.95 ± 0.15
	Hlg.	0.48 ± 0.07	0.72 ± 0.14	0.52 ± 0.07	0.86 ± 0.24	0.91 ± 0.20	0.55 ± 0.09
Free Flow	KL	0.87 ± 0.13	1.18 ± 0.18	0.91 ± 0.13	1.26 ± 0.31	1.59 ± 0.34	0.96 ± 0.12
	Hlg.	0.49 ± 0.07	0.73 ± 0.15	0.52 ± 0.07	0.89 ± 0.21	0.95 ± 0.19	0.55 ± 0.08
Relative Training Time		1×	1.64×	1.08×	0.89×	1.40×	N/A



(a) Daily KL divergence. (b) Weekly KL divergence.

Fig. 5. KL divergence performance of DeepTTDE at various GPS trajectory sparsities.

mator. We emulate practical scenarios where only 1%, 2%, 5%, 10%, or 20% of all road link data are available. These configurations can be achieved by using the same travel time modeling approach in Section 3.2 and stopping the trajectory-inclusion process at the above coverage percentage. For reference, the no-real-time-trajectory case ($\beta = 0\%$) is also emulated, where Hist-1D instead of DeepTTDE is adopted as a baseline approach.

Fig. 5 depicts the KL divergence of DeepTTDE given various GPS trajectory sparsities, where the weekly KL divergence is averaged over each hour for better illustration. It is also observed that the number of available trajectories does not have a notable influence on the model training and inference time. The simulation results accord with the intuition that more available real-time data generally leads to better-estimated travel time distributions. When β is smaller than 10%, increasing β significantly reduces the average KL divergence. Additionally, the performance of DeepTTDE is less stable with few data. For instance, the free flow off-peak performance is notably inferior to that during the congestion time periods. This is due to that congested time periods typically provide more GPS trajectories, rendering more data available given a fixed β percentage. The pattern is also clearly identifiable in the weekly chart, where the worst-case performance of 1% GPS sparsity is close to no-data scenario because the bare minimal available trajectories are not sufficient to drive DeepTTDE for proper estimations. However, none of the above issues exist when sufficient trajectories ($\beta \geq 10\%$) are provided. Considering that the current state-of-the-art TTDE methods require approximately 63% [8] to 70% [4] coverage of road links, DeepTTDE is capable of being applied to scenarios where much fewer real-time trajectories are available.

4.4 Ablation Test

In this work, we devise three TTDE modules to construct DeepTTDE. While the first map matching and travel time modeling module implements the standard data preprocessing for travel time estimation and is indispensable, the architecture of the Bayesian graph autoencoder and the inclusion of the adversarial training model is based on the intuitive insights into TTDE. In this subsection, we carry out an ablation test to validate their efficacy in the proposed DeepTTDE. In particular, we are interested in investigating the contribution of graph convolution and adversarial training towards the overall performance. Therefore, the following four variants of DeepTTDE are constructed:

- **DeepTTDE-AEGC**: All graph convolution operations in the Bayesian graph autoencoder is substituted by fully-connected neurons, i.e., (1) is replaced by $\mathbf{y} = \sigma(\mathbf{wx} + b)$ where b is a trainable bias parameter.
- **DeepTTDE-ATGC**: All graph convolution operations in the adversarial training model is substituted by fully-connected neurons. The last summation-sigmoid transformation is replaced by an additional layer of one fully-connected neuron activated by sigmoid.
- **DeepTTDE-AT**: The adversarial training model is removed from DeepTTDE. The training objective becomes the KL divergence defined in (5).
- **DeepTTDE-GC-AT**: All graph convolution operations in the model is substituted by fully-connected neurons. Additionally, the adversarial training model is removed.

All DeepTTDE variants are trained with the same dataset and configurations as introduced in Section 4.1.

Table 3 summarizes the simulation results of DeepTTDE variants with the KL divergence and Hellinger distance of the best-performing baseline, i.e., BISN, for reference. Additionally, the relative training time for DeepTTDE variants is presented considering the time required by DeepTTDE as the base value. The comparison indicates the necessity of graph convolution and adversarial training in DeepTTDE, in which adopting either technique can provide a notable performance metric improvement. This conclusion can be

TABLE 4
Hyperparameter Configurations

Label	Bayesian Graph Autoencoder	Adversarial Model
DeepTTDE	$128 + 256 + 128 + 1$	$64 + 16 + 1$
A	$128 + 1$	$64 + 16 + 1$
B	$128 + 256 + 128 + 1$	$16 + 1$
C	$128 + 128 + 256 + 128 + 128 + 1$	$64 + 16 + 1$
D	$128 + 256 + 128 + 1$	$64 + 64 + 16 + 1$
E	$64 + 128 + 64 + 1$	$32 + 16 + 1$
F	$256 + 256 + 256 + 1$	$128 + 16 + 1$

directly drawn by comparing DeepTTDE with DeepTTDE-AEGC and DeepTTDE-AT where approximately 36% and 45% average KL divergence degradations are witnessed by removing either technique, respectively. Additionally, the introduction of graph convolution can aim in the model's convergence by relieving the model of the burden of extracting graphical information from the training data due to the topology that has been incorporated into the model. Even though the training time for removing the adversarial model is lowered by 11%, the estimation accuracy has suffered as a result of the reduction. Therefore, the adversarial model is critical when it comes to both prediction and time performance. Finally, BISN can estimate better travel time distributions than both variants, rendering the two techniques essential in DeepTTDE.

4.5 Hyperparameter Test

In Section 3 we propose two neural networks in DeepTTDE, namely a Bayesian graph autoencoder and an adversarial training model. When designing a neural network, what are the optimal numbers of layers and neurons is a common question to be answered in order to achieve satisfactory model performance. While there are no generalized theoretical analyses that guide the selection of these hyperparameters, the trial-and-error approach is widely adopted to determine neural network architectures. In this subsection, we construct DeepTTDE variants according to the hyperparameter configurations recorded in Table 4 and examine their TTDE performance as the hyperparameter test. In this table, the values under column "Bayesian Graph Autoencoder" denote the number of graph convolution layers and their respective number of neurons (nodal features) in the distribution estimator of DeepTTDE, and the values under "Adversarial Model" describe those in the adversarial training model. For ease of discussion, each hyperparameter configuration is referred to as the label under the "Label" column.

Fig. 6 presents the performance comparison of the hyperparameter configurations. In this figure, the training time is presented as a relative value to that of DeepTTDE. From the comparison, we can come to a series of conclusions. First, reducing the model capacity by removing layers or neurons generally impairs the model performance. This can be derived by comparing DeepTTDE with models A, B, and E, which reduces the Bayesian graph autoencoder, the adversarial model, and both, respectively. The conclusion accords with the widely recognized principle of deep learning that adequately more layers and neurons in a neural network can improve its expressibility and therefore exploit more latent data characteristics for inference. Nonetheless, such improvement experiences marginal utility when the model capacity is sufficient for the designated task, and further expansion leads to drastically increased training time as a side effect of the expanding network parameter searching space. This is reflected in the comparison where models C and F use much more training time to achieve a comparable performance than the original DeepTTDE.

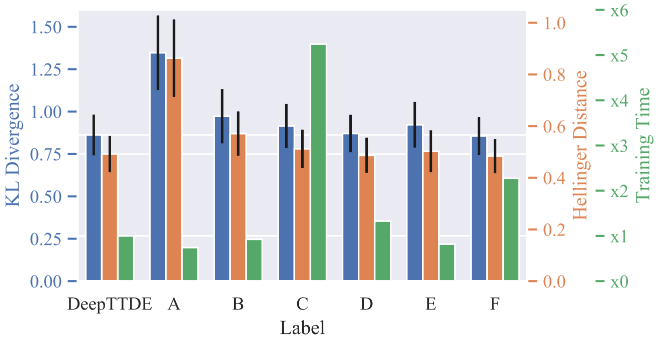


Fig. 6. Performance of DeepTTDE hyperparameter variants

TABLE 5
Performance of DeepTTDE and Baseline Methods on SIGSPATIAL 2021 GISCUP Dataset

Metric	DeepTTDE	BISN	DeepGTT
KL	0.95 ± 0.15	1.11 ± 0.17	1.28 ± 0.19
Hlg.	0.57 ± 0.09	0.66 ± 0.12	0.83 ± 0.13

Apart from the preceding architectural tests, another crucial hyper-parameter affecting model performance is the number of samples required to approximate the prior distribution in Bayesian inference, i.e., K. We also run offline tests on simulation settings with $K \in \{8, 16, 32, 64\}$ and observe that when K is decreased, model estimation accuracy degrades rapidly. For instance, the KL divergence of $K = 16$ reaches as high as 1.81 ± 0.38, nearly doubling that of the default $K = 32$. This is contributed by the calculation of loss defined in (8), which is partially grounded on the KL divergence calculation. A small K value renders the generated discrete distribution highly volatile and stochastic, which cannot resemble the posterior distribution and deteriorate the loss and subsequently the back-propagating gradient. On the other hand, increasing K to 64 does not impose statistically significant performance improvement but instead leads to prolonged training time. As each parameter sample of K requires one forward pass through the neural network, there is a saturation point of this hyper-parameter where introducing more samples cannot further improve the quality of loss. We figured that 32 is a good candidate for K, and adopt this value in all case studies.

4.6 Generality of DeepTTDE

KDD CUP 2020 dataset is adopted as the testing data in all previous case studies, which may not indicate the generality of DeepTTDE on other datasets. In this section, we further adopt the SIGSPATIAL 2021 GISCUP data[3] as an

[3] Available at https://www.biendata.xyz/competition/didi-eta/rules/

indication on whether the proposed model can be adopted in other cities and datasets. Specifically, the dataset includes approximately nine million trajectories in the city Shenzhen, China during August 2020, each of which includes the exact individual travel time of all traversing road links. The individual travel time values of each road comprising one or multiple links during the same time period are aggregated to formulate the ground truth travel time distribution, which are further utilized to construct base travel time maps according to Section 3.2. All simulation configurations are kept identical to Section 4.1, and the simulation results comparing DeepTTDE with BISN and DeepGTT are summarized in Table 5. The results indicate that DeepTTDE possess the generality of being applied to other cities and datasets while maintaining the advantage over the best-performing baselines powered by both Bayesian inference and deep learning techniques.

5 Conclusions

In this work, we propose a new travel time distribution estimation method based on Bayesian and geometric deep learning techniques. This method, named DeepTTDE, exploits the historical vehicular GPS trajectories to model citywide road link travel time distributions. Particularly, the historical data are first pre-processed to construct past travel time maps, each of which represents an instantaneous travel time status of the investigated transportation network. A novel Bayesian graph autoencoder neural network is subsequently used to learn from the historical data and explore the data uncertainties for estimating travel time distributions. Finally, an auxiliary neural network is incorporated to stabilize the training process of DeepTTDE. The proposed DeepTTDE is among the pioneering work on using deep learning techniques to estimate travel time distributions. It addresses a few critical challenges of TTDE, namely, data uncertainty and network topology learning, and probability distribution generation.

To evaluate the performance of DeepTTDE, we conduct a series of comprehensive case studies on a large-scale real-world dataset of vehicular trajectories. The comparison with state-of-the-art baselines demonstrates the outstanding performance of DeepTTDE, which achieves approximately 10% KL divergence reduction with much less real-time data requirement. Additionally, we conduct a data sparsity test to reveal the sensitivity of DeepTTDE on the availability of real-time GPS trajectories, and an ablation test to investigate the necessity of the constituting components of DeepTTDE. Finally, a hyperparameter test proposes guidelines to determine the optimal neural network structure.

Future work can be divided into three parts. First, it is possible to extend the proposed method to address a wide range of data-driven intelligent transportation system problems, e.g., bus arrival time estimation and vehicular trajectory prediction. The merits of DeepTTDE can also address the challenges of these problems. Second, existing non-deep learning methods for TTDE may help design a hybrid estimator based on ensemble learning, which can be expected to inherit benefits from both parents. The existing undirect graph relaxation of transportation networks may also be replaced by directed graph convolution for better performance. Last, future research may uncover superior alternatives to DeepTTDE's building blocks, such as the pre-processes, which will help a wide range of traffic big data analysis applications in general.

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[2] F. Y. Wang, "Parallel control and management for intelligent transportation systems: concepts, architectures, and applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 630–638, Sep. 2010.

[3] A. Perallos, U. Hernandez-Jayo, I. J. G. Zuazola, and E. Onieva, *Intelligent transport systems: technologies and applications*. John Wiley & Sons, 2015.

[4] A. Prokhorchuk, J. Dauwels, and P. Jaillet, "Estimating travel time distributions by Bayesian network inference," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1867–1876, 2020.

[5] Y. Shen, C. Jin, and J. Hua, "TTPNet: A neural network for travel time prediction based on tensor decomposition and graph embedding," *IEEE Transactions on Knowledge and Data Engineering*, 2020, in press.

[6] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 7, pp. 1782–1795, 2015.

[7] K. Tang, S. Chen, and Z. Liu, "Citywide spatial-temporal travel time estimation using big and sparse trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 4023–4034, 2018.

[8] P. Duan, G. Mao, J. Kang, and B. Huang, "Estimation of link travel time distribution with limited traffic detectors," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3730–3743, 2020.

[9] S. Susilawati, M. A. P. Taylor, and S. V. C. Somenahalli, "Distributions of travel time variability on urban roads," *Journal of Advanced Transportation*, vol. 47, no. 8, pp. 720–736, 2013.

[10] E. K. M. Moylan and T. H. Rashidi, "Latent-segmentation, hazard-based models of travel time," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 8, pp. 2174–2180, 2017.

[11] R. Zhang, S. Newman, M. Ortolani, and S. Silvestri, "A network tomography approach for traffic monitoring in smart cities," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2268–2278, 2018.

[12] R. Li, G. Rose, and M. Sarvi, "Using automatic vehicle identification data to gain insight into travel time variability and its causes," *Transp. Res. Rec.*, vol. 1945, no. 1, pp. 24–32, 2006.

[13] W. Pu, "Analytic relationships between travel time reliability measures," *Transp. Res. Rec.*, vol. 2254, no. 1, pp. 122–130, 2011.

[14] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos, "Non-parametric estimation of route travel time distributions from low-frequency floating car data," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 343–362, 2015.

[15] D. Delling, A. Goldberg, M. Muller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. Werneck, "Route planning in transportation networks," Microsoft Research, Tech. Rep. MSR-TR-2014-4, January 2014.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[17] H. Wang and D.-Y. Yeung, "Towards bayesian deep learning: a framework and some existing methods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, pp. 3395–3408, 2016.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. International Conference on Learning Representations*, Toulon, France, Apr. 2017.

[19] S. Wang, J. Cao, and P. Yu, "Deep learning for spatio-temporal data mining: A survey," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.

[20] K. Zhang, N. Jia, L. Zheng, and Z. Liu, "A novel generative adversarial network for estimation of trip travel time distribution with trajectory data," *Transportation Research Part C: Emerging Technologies*, vol. 108, pp. 223–244, 2019.

[21] X. Zhang and J. A. Rice, "Short-term travel time prediction," *Transportation Research Part C: Emerging Technologies*, vol. 11, no. 3, pp. 187–210, 2003, traffic Detection and Estimation.

[22] C.-H. Wu, J.-M. Ho, and D. T. Lee, "Travel-time prediction with support vector regression," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 276–281, 2004.

[23] F. Zheng and H. van Zuylen, "Urban link travel time estimation based on sparse probe vehicle data," *Transportation Research Part C: Emerging Technologies*, vol. 31, pp. 145–157, 2013.

[24] B. Yang, C. Guo, and C. S. Jensen, "Travel cost inference from sparse, spatio temporally correlated time series using markov models," *Proc. VLDB Endow.*, vol. 6, no. 9, pp. 769–780, Jul. 2013.

[25] Y. Guessous, M. Aron, N. Bhouri, and S. Cohen, "Estimating travel time distribution under different traffic conditions," *Transportation Research Procedia*, vol. 3, pp. 339–348, 2014.

[26] Q. Yang, G. Wu, K. Boriboonsomsin, and M. Barth, "A novel arterial travel time distribution estimation model and its application to energy/emissions estimation," *Journal of Intelligent Transportation Systems*, vol. 22, no. 4, pp. 325–337, 2018.

[27] K. Wan and A. L. Kornhauser, "Link-data-based approximation of path travel time distribution with gaussian copula estimated through lasso," in *Proc. Transportation Research Board 89th Annual Meeting*, Washington DC, Jan. 2010.

[28] Y. Yu, M. Chen, H. Qi, and D. Wang, "Copula-based travel time distribution estimation considering channelization section spillover," *IEEE Access*, vol. 8, pp. 32 850–32 861, 2020.

[29] C. Shi, B. Y. Chen, and Q. Li, "Estimation of travel time distributions in urban road networks using low-frequency floating car data," *ISPRS International Journal of Geo-Information*, vol. 6, no. 8, 2017.

[30] M. Ramezani and N. Geroliminis, "On the estimation of arterial route travel time distribution with Markov chains," *Transportation Research Part B: Methodological*, vol. 46, no. 10, pp. 1576–1590, 2012.

[31] T. Hunter, R. Herring, P. Abbeel, and A. M. Bayen, "Path and travel time inference from GPS probe vehicle data," in *Proc. Neural Information Processing Systems foundation*, Vancouver, Canada, Dec. 2009.

[32] X. Li, G. Cong, A. Sun, and Y. Cheng, "Learning travel time distributions with deep generative model," in *Proc. The World Wide Web Conference*, 2019, pp. 1017–1027.

[33] H. Yuan, G. Li, Z. Bao, and L. Feng, "Effective travel time estimation: When historical trajectories over road networks matter," in *Proc. ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2135–2149.

[34] K. Guo, Y. Hu, Z. Qian, H. Liu, K. Zhang, Y. Sun, J. Gao, and B. Yin, "Optimized graph convolution recurrent neural network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2020.

[35] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–21, 2020.

[36] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proc. International Conference on Machine Learning*, 2015, p. 1613–1622.

[37] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G. Sun, "An interactive-voting based map matching algorithm," in *International Conference on Mobile Data Management*, Kansas City, MO, May 2010, pp. 43–52.

[38] R. Das and S. Winter, "Automated urban travel interpretation: a bottom-up approach for trajectory segmentation," *Sensors*, vol. 16, no. 11, p. 1962, Nov 2016.

[39] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet, "Online map-matching based on hidden Markov model for real-time traffic sensing applications," in *IEEE International Conference on Intelligent Transportation Systems*, Anchorage, AK, Sep. 2012, pp. 776–781.

[40] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Advances in Neural Information Processing Systems*, Barcelona, Spain, Dec. 2016.

[41] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. International Conference on Learning Representations*, Banff, Canada, Apr. 2014.

[42] R. Bassett and J. Deride, "Maximum a posteriori estimators as a limit of Bayes estimators," 2016, `arXiv:1611.05917 [math.ST]`.

[43] D. J. C. MacKay, "Bayesian methods for adaptive models," Ph.D. dissertation, California Institute of Technology, 1992.

[44] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, no. 4, pp. 1303–1347, 2013.

[45] M. Sun, T. Zhang, Y. Wang, G. Strbac, and C. Kang, "Using Bayesian deep learning to capture uncertainty for residential net load forecasting," *IEEE Trans. Power Syst.*, vol. 35, no. 1, pp. 188–201, 2020.

[46] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, Cambridge, 2016.

[47] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. International Conference on Learning Representations*, Banff, Canada, Apr. 2014.

[48] A. Graves, "Practical variational inference for neural networks," in *Proc. International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2011, p. 2348–2356.

[49] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[50] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: an overview," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018.

[51] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F. Y. Wang, "Generative adversarial networks: introduction and outlook," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[52] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proc. International Conference on Learning Representations*, San Diego, CA, Dec. 2015.

[53] R. Peharz, A. Vergari, K. Stelzner, A. Molina, X. Shao, M. Trapp, K. Kersting, and Z. Ghahramani, "Random sum-product networks: A simple and effective approach to probabilistic deep learning," in *Proc. Uncertainty in Artificial Intelligence Conference*, ser. Proceedings of Machine Learning Research, vol. 115, Jul. 2020, pp. 334–344.