Cross-Area Travel Time Uncertainty Estimation From Trajectory Data: A Federated Learning Approach

Yuanshao Zhu, *Member, IEEE*, Yongchao Ye, *Student Member, IEEE*, Yi Liu, *Graduate Student Member, IEEE*, and James J. Q. Yu, *Senior Member, IEEE*

***Abstract*— Along with urbanization and the deployment of GPS sensors in vehicles and mobile phones, massive amounts of trajectory data have been generated for city areas. The analysis of these data has substantially contributed to research and advancements of travel time estimation. However, existing work focuses on estimating travel time inside a particular area, and cross-area travel time estimation has privacy security challenges due to data exchange issues among areas. Meanwhile, the majority of methods estimate a deterministic travel time for a given trajectory, which does not account for complex traffic situations and user requirements. To address these problems, we propose a cross-area travel time uncertainty estimation algorithm for estimating the uncertainty of travel times while preserving privacy among different areas. Specifically, we design a comprehensive cross-area privacy-preserving solution that trains a tailor-made neural network travel time estimator in each area by local data, and incorporates federated learning for training. Furthermore, we employ Bayesian deep learning principles and adopt Monte-Carlo dropout to quantify the uncertainty associated with travel time. To evaluate the proposed approach, we conduct a series of comprehensive case studies with two real-world trajectory datasets. Extensive results demonstrate the superiority of the proposed approach compared to baselines in the context of the cross-area setting.**

***Index Terms*— Travel time estimation, federated learning, Bayesian deep learning, trajectory analytics, privacy-preserving.**

Manuscript received 30 March 2022; revised 13 June 2022; accepted 24 August 2022. This work was supported in part by the Stable Support Plan Program of Shenzhen Natural Science Fund under Grant 20200925155105002 and in part by the Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation under Grant 2020B121201001. The Associate Editor for this article was M. Mesbah. *(Yuanshao Zhu and Yongchao Ye contributed equally to this work.) (Corresponding author: James J. Q. Yu.)*

Yuanshao Zhu is with the Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: zhuys2019@mail.sustech.edu.cn).

Yongchao Ye and James J. Q. Yu are with the Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: 12032868@mail.sustech.edu.cn; yujq3@sustech.edu.cn).

Yi Liu was with the Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. He is now with the Department of Computer Science, City University of Hong Kong, Hong Kong, SAR, China (e-mail: 97liuyi@ieee.org).

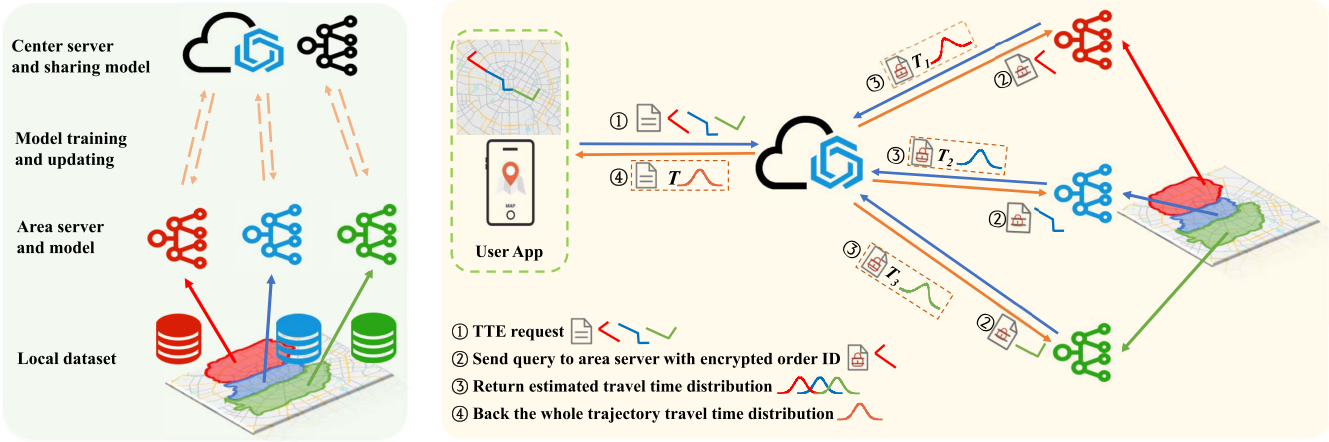
Digital Object Identifier 10.1109/TITS.2022.3203457

I. Introduction

With the popularity of GPS embedded devices and data collection technology, the massive trajectory data provides core support for the advance of intelligent transportation systems (ITS) [1]. The analytics of trajectory data has energized the ITS community, prompting a range of emerging applications, such as traffic prediction and travel time estimation (TTE) [2], [3]. Within ITS, accurate travel time estimation can help online taxi-hailing and navigation platforms (e.g., Didi chuxing, Google Maps) determine the optimal travel route and thus improve operational efficiency, which in turn can provide a favorable user experience [4], [5], [6]. Therefore, TTE techniques are critical for individuals and transportation service providers.

Over recent years, TTE has attracted widespread attention in the ITS community. In this context, the straightforward TTE approach calculates the average travel time from origin to destination with historical data, which has the advantage of easy implementation [7], [8], [9], [10]. However, these methods have a disadvantage in that they ignore external factors (departure time and weather conditions, for some examples) and spatio-temporal information about the road network that significantly impacts travel time, resulting in poor estimation accuracy [11]. Furthermore, the route planning objectives are not addressed by using only origin and destination, e.g., in the case of multiple routes connecting the same origin and destination.

In response to this issue, the research community is turning its attention to trajectory-based TTE studies, which aim to conduct spatio-temporal modeling of trajectory data to extract fine-grained features for estimating travel time [4], [12], [13]. Therefore, the majority of existing TTE work is designed to provide accurate and deterministic travel times for routes inside the transportation network. However, the deterministic trajectory travel time is influenced by a variety of factors that are difficult to represent uniformly, such as human behavior patterns and traffic conditions [6], [14]. As a result, deterministic travel time estimators ignore the traffic uncertainty and are insufficient to provide a reliable estimation. From the perspective of users, they may benefit from the reachable time confidence of their trips to help them plan their departure times and select routes. Uncertainty models that estimate auxiliary



(a) Training scheme for cross-area TTE system.

(b) Inference scheme for cross-area TTE system.

Fig. 1. Overview of the cross-area federated travel time uncertainty estimation algorithm. In the training stage (a), the area servers collect data from each area and then collaboratively train a travel time estimator through the central server based on its local data. In the test stage (b), the user application first divides a TTE query into corresponding segments according to areas and sends them to the central server. The central server sends segments to the corresponding area server, which returns a travel time distribution respectively. Finally, sums these distributions as the estimated value.

arrival probability can serve users better than deterministic TTE models [15].

Furthermore, existing industry solutions and research efforts for TTE mainly focus on estimating travel time based on trajectory data within a single administrative area (e.g., a central city). In fact, with the decentralization and area-based deployment of traffic management systems, a travel trajectory will likely cover multiple administrative areas within a city. Therefore, it is necessary to consider the cross-area issue when conducting TTE research. Fig. 1(a) presents an example of the cross-area TTE system, where different areas hold a local dataset. Intuitively, the center server can aggregate data from all areas and estimate travel time. However, strict privacy regulations (e.g., the General Data Protection Regulation [16]) prohibit service providers from exchanging data with other third-party entities to build predictive models involving massive amounts of data. For this reason, when performing cross-area TTE, we should legitimately rethink how to acquire data for travel time estimation modeling without comprising privacy.

To summarize, a satisfactory trajectory-based TTE system should be designed to meet the following challenges:

- **Spatio-temporal modeling:** As trajectory data contain abundant spatio-temporal information, a proper travel time estimator should be able to capture the spatio-temporal correlation of trajectory to extract fine-grained features, subsequently enabling accurate and efficient travel time estimation.
- **Uncertainty Estimation:** The travel time is affected by various factors that are difficult to record (e.g., traffic conditions and human behavior). Therefore, a reliable travel time estimator needs to provide the distribution and uncertainty of the travel time.
- **Cross-area Estimation:** Since trip trajectory is likely to traverse multiple areas and data cannot be shared owing to privacy concerns, a satisfactory TTE algorithm should take into account the problem of cross-area travel time estimation.

To jointly tackle the above challenges, we design a cross-area federated travel time uncertainty estimation algorithm for trajectory data, which can provide a scheme for training and testing travel time estimators while preserving privacy. In addition, the proposed scheme is capable of estimating the travel time distribution and uncertainty of a trajectory. The main contributions of this paper are summarized as follows:

- We propose a novel travel time estimation model, which uses manually extracted trajectory features and external attributes as inputs and employs a deep neural network to extract fine-grained features to estimate travel time.
- We adopt a Bayesian deep learning approach to improve the robustness of the proposed model, thereby providing uncertainty estimates of travel time.
- We design a well-established solution for a cross-area travel time uncertainty estimation algorithm, which enables training and testing of models while preserving privacy.
- We validate the effectiveness of the proposed scheme on two real-world datasets. The empirical results demonstrate the significance and scalability of the proposed approach.

The remainder of this paper is organized as follows. We first review the background of cross-area travel time uncertainty estimation issues in Sec. II. Then, we present the problem definitions and the cross-area training scheme in Sec. III. Next, we elaborate the proposed algorithm and trajectory analytics in Sec. IV. Subsequently, we conduct a series of case studies on two real-world trajectory datasets and analyze the simulation results in Sec. V Finally, we conclude this paper in Sec. VI.

II. Related Work

A. Trajectory-Based Travel Time Estimation

With the massive deployment of GPS sensors on vehicles, researchers used GPS trajectories and combined them

with advanced neural network techniques for TTE [17]. For example, Wang *et al.* proposed an error feedback recurrent convolutional neural network to estimate the travel time and velocity on sequential GPS points, then inferred the estimated time based on the velocity of each trajectory segment [18]. But the simple accumulation of travel times for each trajectory segment does not eventually produce highly accurate results. As a result, several studies improved this deficiency by constructing a spatio-temporal learning component to estimate the travel time of the complete trip [5], [19]. Regarding the challenge of predicting future trajectory information, a number of studies mapped trajectories to road networks and used a series of interconnected road segments traversed by trajectories for TTE [12], [20], [21], [22]. Wang *et al.* proposed a Wide-Deep-Recurrent learning model, which combined the advantages of wide linear models, deep neural networks, and recurrent neural networks. This scheme improved estimation accuracy by integrating external factors (weather, departure time, driver ID, etc.). Based on this concept, [21] improved the model inference speed for this scheme, and [20], [22] introduced auxiliary learning tasks to improve the model accuracy. Besides, there are a number of schemes that combined trajectory and graph neural networks to estimate travel times [11], [13], [23], which are not the main study focus of this paper and not be detailed here.

To summarize, a number of methods have been proposed for TTE. Nevertheless, few studies focused on modeling and data uncertainty learning, which is significant as traffic systems are complex and dramatically changing. In addition, the existing spatio-temporal modeling neural network models designed for solving TTE still have substantial potential for improvement. Therefore, we propose a novel neural network model for TTE that has superiority over existing methods.

B. Traffic Uncertainty Estimation

Due to the dynamic nature of human activities and traffic status, uncertainty estimation is also increasingly important in ITS. Markos *et al.* [24] proposed a Bayesian deep learning method for unsupervised GPS trajectory segmentation, which used the mean and variance of the prediction distribution to classify each input trajectory and estimate its prediction uncertainty. Reference [25] designed a Bayesian deep learning model for traffic speed prediction that can provide reliable results in transfer learning and learn favorable feature representations in scenarios with missing traffic data or insufficient data. These methods are representative of the literature on predicting traffic uncertainty through Bayesian deep learning. For specific TTE studies, Liu *et al.* [15] proposed a scheme for predicting travel time uncertainty based on a geographically indistinguishable approach, but modifying the raw trajectory data inevitably degrades the model performance. Furthermore, Yu *et al.* proposed a Bayesian and geometric deep learning method to estimate citywide travel time distribution [6]. The evaluation shows that the probabilistic distribution can describe the TTE better than the deterministic model.

Although the above methods have made long-term progress in traffic uncertainty estimation, they are still not considering cross-area TTE issues, while it has evoked substantial concerns. In this paper, we apply a Bayesian deep learning approach to provide travel time uncertainty estimates to improve the robustness of the proposed model.

C. Cross-Area Issues in Intelligent Transportation Systems

For contemporary ITS research, cross-area issues are inevitable due to the area-based deployment of management systems. However, since GPS point is closely related to personal location, the analytics of GPS trajectory data may reveal privacy information. On the other hand, with the enforcement of data privacy regulations [16], service providers or third-party organizations are strictly prohibited in exchanging users' private data. Therefore, a well-established cross-area travel time estimation method should not consider data sharing. Traditional solutions to address this problem typically use transfer learning, applying a model trained in one area to another [26], [27]. Nonetheless, these methods are unsuitable for areas with different traffic conditions and are not effective for areas with limited data. Fortunately, federated learning presents a new paradigm for cross-area collaborative training with privacy-preserving. For example, Liu *et al.* introduced federated learning to ITS, allowing it to predict traffic flow without sharing data [28]. Zhu *et al.* proposed supervised and semi-supervised federated learning schemes for travel mode identification, which enable accurate identification while preserving the trajectory data of each local device [29], [30]. With the rising of the internet of vehicles technology, the enhanced computing and communication capabilities make it possible to create collaborative federated learning applications. Ng *et al.* proposed to use unmanned aerial vehicles (UAVs) as communication relays between a central server and vehicles, and designed an auction-coalition formation framework for solving the allocation problem of UAVs [31]. Furthermore, Lim *et al.* considered the privacy concerns of UAVs when serving data for the Internet of Vehicles. They implemented a privacy-preserving collaborative machine learning task using a FL approach and multi-dimensional contract. In view of the heterogeneous sources of data, they used Gale-Shapley algorithm to assign areas to UAVs and achieved cross-areas federated learning training [32].

Inspired by the above work, this paper considers the problem of travel time uncertainty estimation in a cross-area context. We propose a method based on federated learning and Bayesian deep learning to accomplish the task and solve the above challenges, enabling cross-area travel time uncertainty estimation with privacy-preserving.

III. Preliminaries

In this section, we first formalize the cross-area travel time estimation problem and then present the training scheme of the proposed method.

A. Problem Formulation

Given a trajectory $\mathcal{P}$, it is denoted by a sequence of consecutively sampled GPS points, i.e., $\mathcal{P} = \{p_1,\ p_2,\ \ldots,\ p_n\}$.

Each GPS point $p_i = [lat_i,\ lng_i,\ t_i]$ contains the latitude, longitude, and timestamp, representing the device's (and user's) private location at time t_i. In practice, this trajectory may traverse multiple areas, and we can separate it into multiple sub-trajectories according to the boundaries of areas. Therefore, a private local dataset $\mathcal{D}_k = \left\{x^i,\ y^i\right\}_{i=1}^{n_k}$ is preserved for each area, where $x^i = \left\{\mathcal{P}_k^i,\ \text{Attr}^i\right\} \in \mathcal{D}_k$ is the input of the estimator. The Attr^i denotes additional attributes (e.g., departure time, day of week date, and corresponding order), and y^i is the corresponding travel time.

Based on the above description, for a given trajectory $\mathcal{P}$, it can be divided into multiple sub-trajectories $\mathcal{P}_k$ according to the area boundaries $\mathcal{P} = \mathcal{P}_1 \oplus \mathcal{P}_2 \oplus \ldots \oplus \mathcal{P}_k$. The goal of the cross-area TTE problem is to estimate the overall time T based on each sub-trajectory $\mathcal{P}_k$. This process is formulated as follows:

$$T = \sum_{i=1}^{k} f_k(x_k, w), \tag{1}$$

where $f_k(\cdot,\cdot)$ is the area travel time estimator, and w is the corresponding estimator parameter. Note that in our case studies, we project the trip trajectory onto the road network, then train and estimate based on it.

B. Cross-Area Training Scheme

In our cross-area setting, there is a set of area servers $\{S_1,\ \ldots,\ S_k\}$, and a central server S_c. With privacy and security concerns, local dataset $\mathcal{D}_k$ cannot be shared between different area servers. As a privacy-preserving paradigm, federated learning (FL) can be used to collaboratively train a shared deep learning model without compromising privacy between different areas [33]. Therefore, we incorporate the FL framework for assist training in this paper. In the cross-area scenario described in this paper, there are K area servers used to jointly train the global shared model $F(x; w)$, and a central server $\mathcal{S}_c$ is used for communication and parameter aggregation among the areas. Through this approach, it is possible to train cross-area models together without sharing their raw private data [34], [35]. We now introduce the FL-based cross-area training scheme organized into the following steps:

Phase 1 (Initialization) First, the central server initializes the global model $F(x; w)$ parameters w^0 and broadcasts them to all area servers. Then area servers initialize the local model according to the received parameters, i.e., $w_k^0 \leftarrow w^0$.

Phase 2 (Local training) For each round of training t, the central server randomly selects a fraction of all area servers to participate in the FL training task and sends global model parameters w^t to them. Each selected area server trains the local model $f_k(x_k;\ w_k^t)$ based on their dataset $\mathcal{D}_k$. The goal of local training is to minimize the following objective function:

$$\arg\min_{w_k} \mathcal{L}_k(w_k) = \frac{1}{n_k} \sum_{i}^{n_k} \ell(y_i,\ f_k(x_i;\ w_k)), \tag{2}$$

where $\ell(\cdot,\cdot)$ denotes a local loss function (e.g., $\ell(y,\ f_k(x; w)) = \frac{1}{2}(x^T w - y)$). Then, they send the updated model parameters w_k^{t+1} back to the central server.

Phase 3 (Aggregation) After receiving w_k^{t+1} from area servers, the central server updates the global parameters by using an aggregation algorithm such as Federated averaging (FedAvg [34]):

$$w^{t+1} = \frac{1}{|K_t|} \sum_{k \in K_t} w_k^{t+1}, \tag{3}$$

where $|K_t|$ denotes number of selected area servers. Specifically, the global minimizing objective is formulated as:

$$\arg\min_{w} \mathcal{L}(w) = \sum_{k=1}^{K} \frac{|K_t|}{|K|} \mathcal{L}_k(w). \tag{4}$$

Note that the aforementioned phases 2 and 3 repeat until the global model reaches convergence.

IV. Methodology

In this section, we introduce the proposed solution for cross-area travel time uncertainty estimation. We first describe trajectory data analytics techniques for manual feature extraction, including map matching and road network speed distribution recovery. Second, we elaborate on the architecture of the proposed travel time estimator. Then, we present the Bayesian deep learning method for estimating uncertainty by Monte-Carlo dropout [36]. Finally, we provide a detailed description of the cross-area travel time uncertainty estimation algorithm.

A. Trajectory Analytics

1) Map Matching for Trajectory Data: Before delving into the details of each module that constitutes the travel time estimator, we first explain the map matching technique used for processing the trajectories. We do not directly utilize the original trajectories for time estimation as in [4] and [5] for two main reasons: 1) In terms of origin and destination, projecting trajectories onto the road network instead of using original GPS points can obscure the user's precise location and alleviate concerns about privacy leakage. 2) For the practical deployment of TTE applications, it is difficult or infeasible to predict users' trajectories as they have different behavioral habits.

In real-world scenarios, the GPS points collected are not always accurate due to signal blockage or time delays. As depicted in the upper part of Fig. 2(a), points in red dashed circles are not on any road segment, which requires us to regularize the trajectory to the road network before processing it. Map matching algorithms that project trajectories points on the road network are well-studied [37], [38]. In this work, we used a map matching algorithm based on the hidden Markov model [37] that is commonly applied in related trajectory studies [39], [40]. For a GPS point p_i, each candidate road segment is represented as a hidden state h in the Markov chain with an observation probability $P(p_i|h)$ that depends on the distance between the trajectory point and the segment. Given a trajectory $\mathcal{P} = \{p_1,\ p_2,\ \ldots,\ p_n\}$, the

(a) (b)

Fig. 2. (a) Illustration of map matching. Upper: Original trajectory points; Lower: Map-matched trajectory points. (b) Heatmap of trajectory points in Chengdu (The deeper color indicates a more dense distribution of trajectories).

objective of map matching is to find a sequence of hidden states $\mathcal{S} = \{s_1,\ s_2,\ \ldots,\ s_n\}$ with maximum likelihood $V_{h,n}$,

$$\begin{aligned} \mathcal{S}_n &= \arg\max_{h\in\mathcal{H}_n} V_{h,n} \\ V_{h,i} &= P(p_i|h) \max_{s\in\mathcal{H}_{i-1}} \left\{ f(s,h) V_{s,i-1} \right\} \end{aligned} \tag{5}$$

where $\mathcal{H}_i$ is the set of hidden state at i and $f(\cdot,\cdot)$ is the transition probability between two hidden state.

With the regularized trajectories (see the lower part of Fig. 2(a)), a trajectory is represented by a sequence of road segments:

$$\mathcal{P} = \{p_1,\ p_2,\ \ldots,\ p_n\} \xrightarrow[\text{matching}]{\text{map}} \mathcal{R} = \{r_1,\ r_2, \ldots, r_m\}, \tag{6}$$

where $\mathcal{R}$ is the set of road segments traversed by the trajectory.

2) Recovery Speed Distribution: After obtaining a collection of map-matched road segments, a number of motion features can be exploited, among which real-time speed is the most important one. This can be easily calculated by the distance between consecutive GPS points and sampling interval. In particular, for two continuous sampling points p_i and p_j on the same road, the sampling time interval is denoted as Δt. The speed of these two point can be calculated by $\text{distance}(p_i, p_j)/\Delta t$. Based on this, the speed of a road segment can be approximated as the average speed of all neighboring sampling points.

However, GPS trajectories are unevenly distributed on the road network, i.e., concentrated in the city center and scattered at the edge areas. Fig. 2(b) presents a heat map with GPS trajectories in Chengdu projected onto the map, where the area in deeper color indicates a more dense distribution of GPS points. Similarly, there are more trajectories in the daytime and fewer trajectories in the nighttime. The uneven distribution of trajectories in temporal and spatial dimensions leads to massive missingness in the historical speed of each road segment. Therefore, we need to recover the speed distribution of the road network throughout. As this problem has been extensively studied in traffic data imputation methods [41], [42], we utilize Bayesian Gaussian CP tensor decomposition to recover speed distribution of the road network [41].

3) Static Features: In addition to the average speed of road segments, there are also a number of attributes that have a significant impact on travel time [11], [12]. In this study, we selected static attributes such as road sequence length, distance, week time, and departure time. We also extracted an important statistical feature from the road segments, i.e., "trip-avg". This feature represents the average travel time for each road, obtained from the distance of road divided by the speed recovered in Sec. IV-A2.

B. Travel Time Estimator

In this subsection, we introduce the architecture of the proposed travel time estimator. As shown in Fig. 3, the proposed model is firstly divided into two parts: namely, attribute learning module and temporal learning module. Following these two modules, we concatenate the latent representations to estimate travel time. We start by introducing the attribute learning module, which aims to learn static attributes mentioned in Sec. IV-A3. For better learning these features, we embed categorical features, i.e., weekID and timeID, into low-dimensional vectors $\mathbb{R}^3$ and $\mathbb{R}^8$, respectively. Specifically, categorical features are embedded by linear layers $y = Ax^T + b$, where x is the one-hot encoded vector and $A \in \mathbb{R}^{D\times V}$ is the weight matrix, V is the vocabulary size and D is the embedding dimension. Subsequently, z-score normalization is adopted on the trip distance, and the number of road segments traversed. The historical average travel time mentioned in Sec. IV-A3 is concatenated as well. After obtaining the attribute embedding, the embedded attribute features are concatenated and fed into three consecutive fully connect layers (FCLs), each of which is mathematically represented as follows:

$$y = \text{Act}(w \cdot x + b), \tag{7}$$

where w and b are trainable parameters, and $\text{Act}(\cdot)$ denotes the activation function, respectively.

The second key component in the travel time estimator is a fundamental operation for fine-grained temporal feature extraction, i.e., the long short-term memory (LSTM) network. LSTM is a common neural network architecture proposed in [43], which is widely used in spatio-temporal data mining and time dependency acquisition tasks thanks to its design of state propagation over time. Specifically, LSTM contains four components: input gate i_t, output gate o_t, forget gate f_t, and intermediate cell state c_t. Given a sequence of road segments as input $\mathbf{x} = \{x_i\}_{1\le i\le N} \in \mathbb{R}^{F\times N}$ (F is the feature dimension for x_i), the output latent vector $\mathbf{h} = \{h_i\}_{1\le i\le N} \in \mathbb{R}^{R\times N}$ can be calculated by the following formulation:

$$i_t = \sigma\left(w_i \cdot \left[h_{t-1}, x_t\right] + b_i\right), \tag{8a}$$
$$f_t = \sigma\left(w_f \cdot \left[h_{t-1}, x_t\right] + b_f\right), \tag{8b}$$
$$o_t = \sigma\left(w_o \cdot \left[h_{t-1}, x_t\right] + b_o\right), \tag{8c}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh\left(w_c \cdot \left[h_{t-1}, x_t\right] + b_c\right), \tag{8d}$$
$$h_t = o_t \odot \tanh\left(c_t\right), \tag{8e}$$

where $\sigma(\cdot)$ is the sigmoid function, and the symbol $\odot$ represents Hadamard product, respectively. All $w \in \mathbb{R}^{R\times F}$ and $b \in \mathbb{R}^R$ denote the trainable weights and bias parameters in the network, respectively. In addition, we incorporate the embedded static attributes with road segments features to further help temporal learning. From the above formulation,

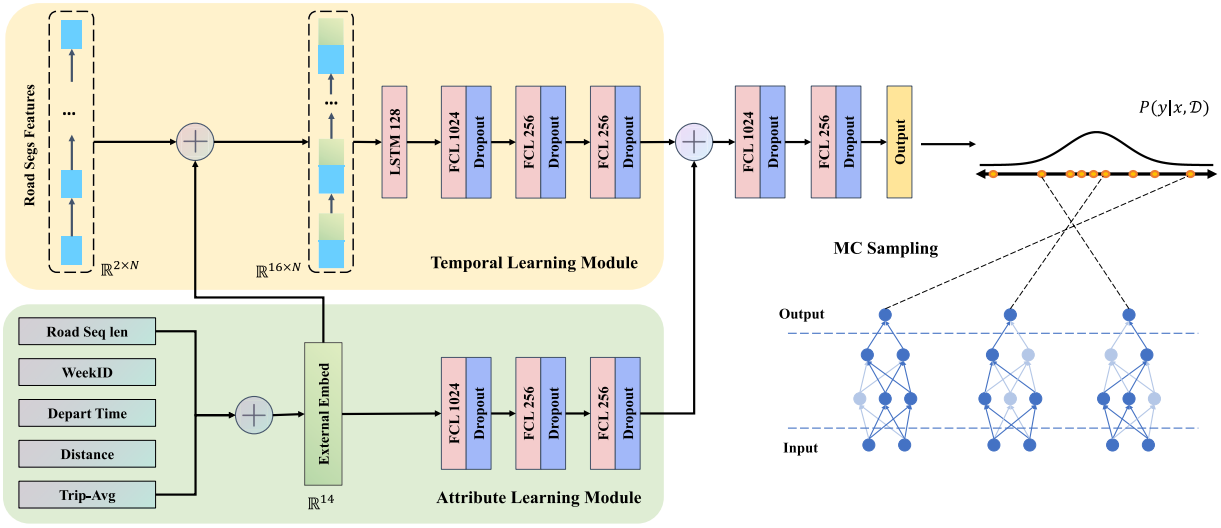


Fig. 3. The architecture of the proposed travel time estimator. The proposed model captures high-dimensional features using the attribute learning module (green part) and the temporal learning module (yellow part), respectively. The potential representations are then concatenated to estimate travel times. Finally, estimating the travel time distribution by multiple Monte Carlo sampling.

it can be observed that the calculation of the current time output h_t involves the integrated calculation of the previous output h_{t-1} and the cell state c_{t-1}, and the total amount of information passed from the previous to the current time is controlled by i_t, f_t, o_t. Through this information flow design, temporal correlation can be extracted from the input data, which is among the core factors for predicting travel times [5]. In this paper, we adopt a two-layer stacked LSTM structure where the hidden size of LSTM cells is set as 128. To extract fine-grained spatio-temporal information, we concatenate the road segment features and attribute embeddings as the input to the LSTM. Similar to the attribute learning module, we attach three FCLs after LSTM layers to extract hidden features.

Finally, the latent representations from the attribute learning module and temporal learning module are concatenated and subsequently fed into two FCLs for the estimation. To summarize, we can formulate the proposed travel time estimator as:

$$\begin{aligned} h_a &= \text{MLP}(\boldsymbol{E}_{\text{wid}} \oplus \boldsymbol{E}_{\text{dt}} \oplus \boldsymbol{E}_{\text{dis}} \oplus \boldsymbol{E}_{\text{len}} \oplus \boldsymbol{E}_{\text{avg}}), \\ h_r &= \text{MLP}(\text{LSTM}(x_{\text{road}} \oplus \boldsymbol{E}_{\text{attr}})), \\ \hat{y} &= \text{MLP}(h_a \oplus h_r), \end{aligned} \tag{9}$$

where $\boldsymbol{E}$ denotes the embedded attributes, and MLP represents a continuous sequence of FCL layers, respectively. h_a and h_r denote the high-dimensional features extracted by the attribute learning module and the temporal learning module, respectively. Except for the last output layer, we use the Rectified Linear Unit (ReLU), i.e., $f(x) = max(0, x)$ as the activation function and apply dropout to all FCLs.

C. Bayesian Uncertainty Estimation

With the previous neural network components, we can build a deep learning model to capture latent features and estimate a deterministic travel time. However, considering the shortcomings of the deterministic model we discussed in Sec. I, uncertainty estimation can better satisfy in a fluctuating traffic environment. Bayesian deep learning is a solution to this problem. In general, a neural network model can be considered as a conditional probability model $P(y \mid x, w)$, where w is the weights in the neural network (including the weights and biases we introduced in Eq. (8) and Eq. (7). For a given training dataset $\mathcal{D}$, the learning objective of a neural network can be regarded as a maximum likelihood estimation accordingly:

$$\text{w}^{\text{MLE}} = \arg\max_{w} \log P(\mathcal{D} \mid w). \tag{10}$$

From a Bayesian probability theory point of view, the Bayesian neural network assumes that the parameters all follow a posteriori probability distribution $P(w \mid \mathcal{D})$ over the dataset. Therefore, the objective of Bayesian deep learning is formulated as:

$$P(y \mid x, \mathcal{D}) = \int p(y \mid x, w)\, p(w \mid \mathcal{D}) \mathrm{d}w. \tag{11}$$

Nevertheless, this posterior probability is intractable according to Bayes theorem:

$$P(w \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid w) P(w)}{P(\mathcal{D})}. \tag{12}$$

In practical analysis, we often utilize variational inference methods [44] to approximate $P(w \mid \mathcal{D})$ by a distribution $q(w \mid \theta)$ parameterized by θ. The optimal variational distribution with parameter θ^* can be found by minimizing the Kullback-Leibler (KL) divergence as follows:

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \mathbf{KL}[q(w \mid \theta) \| P(w \mid \mathcal{D})] \\ &= \arg\min_{\theta} \int q(w \mid \theta) \log \frac{q(w \mid \theta)}{P(w) P(w \mid \mathcal{D})} \\ &= \arg\min_{\theta} \mathbf{KL}[q(w \mid \theta) \| P(w)] - \mathbb{E}_{q(w|\theta)}[\log P(w \mid \mathcal{D})]. \end{aligned} \tag{13}$$

Therefore, given the optimal distribution $q(w \mid \theta^*)$, the predictive distribution Eq. (11) is calculated approximately by:

$$P(y \mid x, \mathcal{D}) = \int p(y \mid x, w)\, q(w \mid \theta^*) \mathrm{d}w = q(y \mid x, \theta^*). \tag{14}$$

In summary, Bayesian deep learning replaces deterministic parameters in a neural network with a distribution of parameters $q(w \mid \theta)$. Since each parameter sample can generate a deterministic inference, we can obtain a posterior distribution by multiple inference, i.e., travel time uncertainty.

In this paper, we adopt Monte-Carlo dropout to obtain probabilistic estimation uncertainty, which is easy to implement and computationally efficient [36], [45]. The core idea is assume that the parameters of each layer in the neural network satisfy the **Bernoulli** (p) distribution and use the parameter p to control the probability of hidden neurons being dropped out. This is also a widely adopted manner in related research, see [24], [46] for some examples.

Specifically, we sample the binary variables of the hidden units in each network layer of the network (except the last layer). The variables of each hidden cell are taken to be 0 with probability p, i.e., for a given input, the cell is dropped. For example, if we apply M samples and collect the output of the network at test time, we can approximate the prediction uncertainty, i.e.:

$$\begin{aligned} \mathrm{Var}(y \mid x) &= \mathrm{Var}[\mathbb{E}(y \mid x, w)] + \mathbb{E}[\mathrm{Var}(y \mid x, w)] \\ &= \mathrm{Var}(f(x, w)) + \sigma^2 \\ &\approx \frac{1}{M} \sum_{i=1}^{M} \left(\hat{y}_{(i)} - \overline{\hat{y}}\right)^2 + \sigma^2, \end{aligned} \tag{15}$$

where $f(x, w)$ is the neural network model, $\hat{y}_{(i)}$ denotes the prediction of $i-$th sample, and $\overline{\hat{y}}$ is the mean of all sample prediction. σ^2 is the data uncertainty and refers to the inherent noise for the data generating process, which can be estimated on an independent validation dataset [47].

With the above theoretical basis of Bayesian neural network, we can construct a travel time uncertainty estimator by Monte-Carlo dropout method. As illustrated in Fig. 3, the proposed neural network model employs the dropout technique after each FCL (except for the last one). Accordingly, we apply Monte-Carlo sampling with $p = 0.1$ during the estimation.

D. Cross-Area Travel Time Estimation Scheme

In the previous subsection, we present a deep neural network-based travel time uncertainty estimator, which extracts spatio-temporal features from trajectories and incorporates a Bayesian deep learning approach to estimate uncertainty. Nevertheless, we still need to consider training and testing this estimator in the context of cross-area scenarios. This section will introduce the training and inferring processes of the proposed cross-area travel time estimation scheme separately. As described in Sec. I, in such a setting, we need to focus on the collaboration of different areas while protecting privacy. Therefore, we present the training scheme of the proposed method and the workflow for performing cross-area uncertainty estimation in the sequel.

Algorithm 1 Cross-Area Travel Time Uncertainty Estimation Algorithm

Input:
Order ID $\mathcal{O}$, travel trajectory $\mathcal{P}$.
Center server $\mathcal{S}_c$, Area server set $\{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$.
Output:
Estimated travel time distribution $\mathcal{T}$.
User:
Split the trajectory $\mathcal{P}$ into n ($n < m$) segments $\{P_1, \ldots, P_n\}$ based on the local map
Encrypt the order ID $\{O_1, \ldots, O_n\} \leftarrow$ **Encrypt**($\mathcal{O}$).
Send trajectory segments and encrypted order pairs $\{(P_1, O_1), \ldots, (P_i, O_i)\}$ to the central server.
Center Server:
Receives the segment and order pairs.
Send pairs to the corresponding area server $\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$.
Area Server:
for area server $\mathcal{S}_a \in \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$, **in parallel do**
  for Monte-Carlo sample $i = 1, \ldots, M$ **do**
    $t_i \leftarrow$ **AreaModel**(P_i, w)
    add t_i to the set of estimated values $\mathcal{T}_a$.
  end for
  send $\mathcal{T}_a = \{t_1, \ldots, t_M\}$ and encrypted order ID O_i to the center server.
end for
Center Server:
Receives all estimated $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ of areas and encrypted order ID.
Cross-merge and sum the estimates from different segment $\mathcal{T} \leftarrow$ **sum**($\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$)
Reconstruct the order ID $\mathcal{O} \leftarrow$ **Decrypt**($\{O_1, \ldots, O_n\}$)
Send estimated travel time distribution $\mathcal{T}$ back to the user according to the order ID $\mathcal{O}$

1) Model Training: The Fig. 1(a) shows the training scheme of the proposed algorithm, consisting of three main components: the central server, the area server, and the local dataset. The area servers utilize their respective local datasets for training, and the central server is responsible for parameters aggregation. For raw trajectories, the map matching method first projects the trajectories to a collection of road segments. Then, according to the area boundaries, the trajectories are partitioned into corresponding areas to arrange training datasets. The area-specific dataset is denoted by $\mathcal{D}_k = \left\{x^i, y^i\right\}_{i=1}^{n_k}$, where k denotes the area number, x^i represents the trajectory segments, and y^i is the ground truth trip time. Based on the local dataset of each area, we can follow the scheme introduced in Sec. III-B. In addition, due to the different spatio-temporal traffic characteristics among areas, after obtaining the global shared model by FL training, each area server still needs to carry out personalized training based on it. It can also be explained that the area server does

not share and update the model with other servers during the last round of federated training.

2) Model Inference: With the personalized estimating model for each area, we can use it for cross-area travel time uncertainty estimation. As summarized in the Algorithm 1, there are three crucial participants in this algorithm, namely, the user, the area server, and the central server. The user is the requester of the travel time prediction in the algorithm, determining the trip trajectory based on the local map and generating a collection of map-matched road segments. Subsequently, the user server divides the trajectory $\mathcal{P}$ into subsets $\{P_1, \ldots, P_n\}$ according to the boundaries of areas. Then encrypt the subsets order ID and sends these road segments and encrypted order pairs to central servers with corresponding order IDs. The central server is the coordinator in the algorithm who sends the road segments that need to be predicted to each area server. After receiving the estimation $\mathcal{T}_i$ of the road segment from each area server, the central server calculates the sum time $\mathcal{T} \leftarrow \textbf{sum}(\{\mathcal{T}_1, \ldots, \mathcal{T}_n\})$. Finally, the estimated time is returned to the corresponding user based on the decrypted order ID. Please note that in order to prevent each area server from colliding with each other to recover the trip information based on the order ID, the central server will send the order ID by encrypting it and then decrypting it after receiving the result (See the Fig. 1(b) for details). The area server is the executor in the algorithm who receives the prediction task and estimates it using the local model, then returns the travel time distribution $\mathcal{T}_i$ for this trip. With the above three roles presented in the Algorithm 1, we can estimate the travel time uncertainty for a cross-area trajectory while preserving privacy.

V. Case Studies

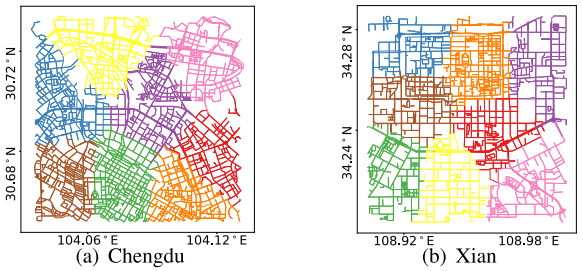
In this section, we evaluate the proposed algorithm by performing comprehensive case studies on two real-world trajectory datasets. We first introduce the datasets and simulation settings. Then, we compare the estimation accuracy with centralized and decentralized baselines and analyze the uncertainty of estimation. Finally, we investigate the scalability of the proposed model by inductive learning.

A. Data and Configuration

In this study, we conduct case studies on two real-world trajectory datasets, which are collected by taxi-hailing vehicles of two major cities in China, namely, Chengdu and Xi'an.[1] Specifically, both datasets are collected from Nov 1st, 2016 to Nov 30th, 2016, with a sampling interval of 2s to 4s. The statistics of both datasets are shown in Table I. The road network topology is obtained from OpenStreeMap.[2] Subsequently, GPS trajectories are mapped to road segments as discussed in Sec. IV-A. Since the proposed model is for cross-area travel time estimation, but the dataset does not have a specific area partition. We need manually partition the road network into different areas and arrange training datasets before starting the simulation. For a specific number of partition areas, the partition is performed based on the latitude and longitude of the road segment using K-means clustering. The partition will not change in the training and testing. Fig. 4 shows the partition of road network in Chengdu and Xi'an with the default setting (We divide the network into 8 areas by default and conduct simulations on different number of areas in Sec. V-C).

TABLE I
Statistics of Two Real-World Trajectory Datasets

Dataset	Trajetory Number	Average Time	Average Distance
Chengdu	3 493 918	11.42 min	7.42 km
Xian	2 180 348	12.58 min	5.73 km



(a) Chengdu (b) Xian

Fig. 4. The partition of road network. Road segments are clustered into different areas based on their latitude and longitude. Different colors indicate different areas.

For validation, we select trajectory recorded in the last week for testing, and the rest of the dataset is sequentially divided into training set and validation set with a ratio of 3 : 1. By default, the model is trained for a total of 30 rounds, with half of the clients randomly selected for each training round and parameter aggregation. For each training round, the selected client trains 10 epochs on its local dataset, and the batch size is set to 1024. The model parameters are updated by the Adam optimizer [48] with an initial learning rate of 0.001. All models are developed by PyTorch, and we perform all case studies are on a server with NVIDIA GeForce RTX 2080Ti GPUs. All experiments in this paper are simulated by serial training on a single server environment as described above. In the default case (8 area servers, half of the areas are selected for training in each round), the average training time per round is 17.4 min, i.e., the average training time per area is 4.4 min (disregarding the communication time between servers). Considering the real situation where the user device only needs to interact with the central server during the inference phase without additional inference computation. We can assume that a complete TTE request can be completed in an acceptable time.

For performance evaluation, we employ three commonly used metrics, including Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE). These three metrics are widely used in contemporary TTE studies [5], [12], [13], [19], and can be

[1]These datasets can be downloaded (following approval of access request) at https://outreach.didichuxing.com/app-vue/

[2]https://www.openstreetmap.org/

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT APPROACHES. MAPE IS REPORTED IN PERCENTAGE (%)

Privacy	Approach	Chengdu			Xian		
		MAPE	RMSE	MAE	MAPE	RMSE	MAE
Centralized	HA	26.62	280.14	191.74	24.46	411.93	199.25
	TEMP	22.36	190.70	133.13	24.87	237.68	161.18
	XGBoost	21.26	174.32	126.30	17.02	169.71	115.48
	WDR	17.81	171.11	116.30	16.06	199.22	121.82
	Ours	**15.82**	**144.18**	**98.43**	**14.73**	**154.66**	**103.27**
Decentralized	WDR-FedProx	21.82	221.94	130.25	17.90	220.30	136.76
	WDR-NoFed	21.23	226.01	125.67	17.84	219.49	136.90
	WDR-FedAvg	20.72	215.44	124.66	17.10	213.82	130.64
	Ours-FedProx	16.85	168.57	107.03	15.23	181.30	109.19
	Ours-NoFed	17.02	168.78	108.22	15.77	182.68	112.42
	Ours-FedAvg	**16.29**	**160.03**	**101.93**	**15.20**	**177.14**	**108.32**

defined as follows:

$$\begin{aligned}
\mathrm{MAPE}(y,\ \hat{y}) &= \frac{1}{N}\sum_{i}^{N}\left|\frac{y^{(i)}-\hat{y}^{(i)}}{y^{(i)}}\right| \times 100\%,\\
\mathrm{RMSE}(y,\ \hat{y}) &= \sqrt{\frac{1}{N}\sum_{i}^{N}\left(y^{(i)}-\hat{y}^{(i)}\right)^{2}},\\
\mathrm{MAE}(y,\ \hat{y}) &= \frac{1}{N}\sum_{i}^{N}\left|y^{(i)}-\hat{y}^{(i)}\right|,
\end{aligned} \tag{16}$$

where $\hat{y}^{(i)}$ is the estimated travel time of trip i, $y^{(i)}$ denotes the ground truth, and N is the number of trips, respectively.

B. Baseline Methods

This section presents the baseline methods adopt in the case studies for comparison under the decentralized and centralized cases, respectively.

Decentralized: In this scenario, we follow the training and inference scheme in Sec. III-B and evaluate the performance of the cross-area algorithm.

- **NoFed:** We train the proposed model in each area independently by its local dataset without parameter aggregation, i.e., we do not use federated learning for model training.
- **FedAvg:** We train the model using the standard federated learning process and aggregate the model parameters by FedAvg.
- **FedProx:** Fedprox [49] is a modification of the FedAvg algorithm, who rewrites the loss function in Eq. (2) by adding the proximal term:

$$\arg\min_{w_k\in\mathbb{R}^d} h(w, w_k) = \mathcal{L}_k(w_k) + \frac{\mu}{2}\left\|w - w_k\right\|^2, \tag{17}$$

  The purpose of this design is to encourage local updates that are not too far from the initial global model and to reduce the impact of data heterogeneity distribution. In this paper, we use FedProx to test the performance of proposed method with different optimization algorithms.

Centralized: In this scenario, instead of partitioning the dataset by area, we treat the entire dataset as a single area for training and testing. By experimenting with this scenario, we can examine the capability of the proposed travel time estimator to model spatio-temporal information. Note that centralized approaches occupy unfair advantages on that they have only one area and do not require collaborative training of the model, violating the privacy preservation presumption.

- **Historical Average (HA):** HA estimates travel time by calculating the average speed of trips with the same departure time.
- **TEMP:** TEMP [8] is a route-free method that estimates travel time by querying and averaging the trips with neighboring origin and destination.
- **XGBoost:** XGBoost [50] is an established ensembling method. We aggregate the road segment sequences to keep each trip with the same feature dimension.
- **WDR:** WDR [12] is a widely deployed trajectory-based TTE method that combines wide, deep, and recurrent neural networks, achieving highly competitive prediction accuracy.
- **Ours:** We used the proposed travel time estimator introducted in Sec. IV-B, and we set the same training round in an identical environment as baseline methods for consistency evaluation.

C. Estimation Performance

Table II summarizes the three evaluation metrics of the proposed method and baseline approaches on the two datasets. From the simulation results, it can be observed that the proposed model outperforms all competing baselines in achieving the lowest MAPE, RMSE, and MAE on all datasets. For centralized scenarios, our approach (15.82%) significantly outperforms other methods, especially for WDR (17.81%), which is also based on deep neural networks. This result can also be verified in the decentralized scenario, where our method is able to achieve optimal results with the same aggregation algorithm. These results strongly suggest that our model has powerful spatio-temporal feature extraction and modeling capabilities in any scenario.

In addition, we can observe that using the FedAvg aggregation scheme is better than the one without aggregation. The reason is that the parameter aggregation can convey localization information of models between different areas,

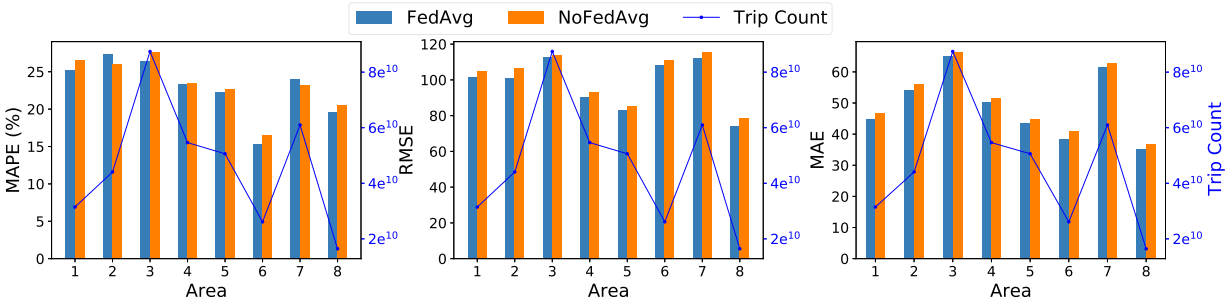


Fig. 5. Performance comparison of each area on Chengdu Dataset. The line graphs indicate the number of trajectories in different areas, and the bar graphs indicate the performance change of the model with/without the use of FedAvg algorithm.

TABLE III
COMPARISON OF ESTIMATION ACCURACY WITH DIFFERENT NUMBER OF AREAS ON CHENGDU DATASET

# Area	MAPE (%)	RMSE	MAE
1	15.81	144.16	98.40
8	16.29	160.03	101.93
16	16.75	163.96	104.89
32	16.77	162.83	105.10

allowing global models to be more general, together with personalized training can be better adapted to cross-area models. The advantage of FedAvg is also supported by the results in Fig. 5. In this case study, we compare the performance of each individual area model after using FedAvg for parameter aggregation. We can observe that parameter aggregation using FedAvg can improve the performance of almost all area models, thus improving the overall performance. In particular, for areas with small amounts of data (e.g., area 1 and area 6), parameter aggregation can significantly improve the performance of individual models. One may note that the FedProx algorithm does not work as well as FedAvg. The reason is that FedProx adds a proximal term (see Eq. (17)) that makes the local model not too far from the global model. Such a design slows down the convergence of the model, which makes it perform inferior to FedAvg.

Furthermore, we also examine the performance of the model for various sizes of manually defined areas, and the results are reported in Table III. We can clearly observe that the model performance decreases as the number of partition areas increases. This is explained by the fact that for the same size dataset, increasing the number of partition areas reduces the amount of local dataset accordingly, thereby affecting the model performance. On the other hand, the convergence speed of the global model can also be affected by too many decentralized models. Consequently, for the proposed cross-area TTE algorithm, smaller areas allow for better area-based traffic management as well as the protection of individual privacy. While too small areas will result in an increase in the number of local models, which will affect the performance.

D. Travel Time Uncertainty

As mentioned previously, besides deterministic estimation, the proposed method also provides the uncertainty

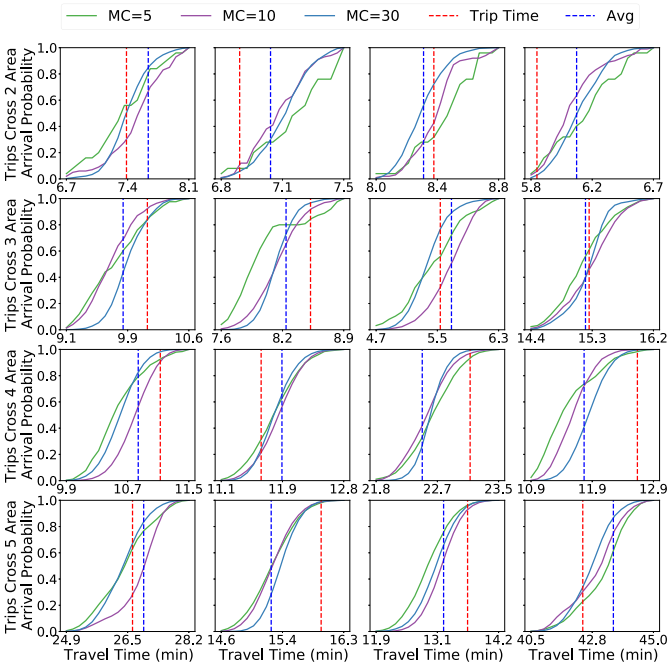


Fig. 6. Arrival probability of trip examples that traverse different number of areas. Although the predicted average time may be greater or less than the true time, the accuracy of the reachable time probability increases as the number of Monte Carlo samples increases. It is shown that performing multiple Monte Carlo sampling can estimate the uncertainty of TTE to the best extent.

quantification of travel time estimation. In this section, we utilize the Monte-Carlo sampling method in Sec. IV-C to estimate the predictive distribution of trips and analyze the arrival uncertainty. We randomly select trip examples that traverse a different number of areas and present the corresponding arrival probability in Fig. 6, where the red dashed vertical line indicates the ground truth trip time and the blue is the average estimated value. We can observe that although the overall MAPE is acceptable, the model's estimated values can be greater or less than the ground truth. In practical applications, such a deterministic value may confuse user's trip planning. Therefore, a more favorable approach is to predict the time distribution (i.e., uncertainty) of this trip and provide reachable time confidence. Fig. 6 also presents the cumulative density function (CDF) curves, which is the cumulative distribution of travel time estimations, indicating

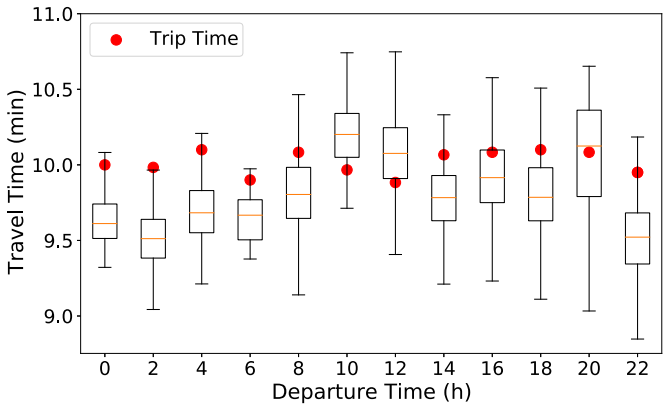


Fig. 7. Box plots of trips with different departure time on Chengdu dataset. Daytime (approximately 8:00 - 20:00) trips have a larger IQR (interquartile range) and maximum-minimum difference than nighttime trips, indicating more complex traffic activity and greater uncertainty in the estimates during the day.

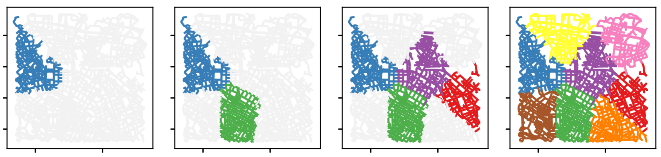


Fig. 8. Visualization of different number of visible areas in training. Areas in grey are unseen in the training stage, while other areas participate FL training.

TABLE IV
ESTIMATION ACCURACY WITH DIFFERENT NUMBER OF VISIBLE AREAS IN TRAINING ON CHENGDU DATASET

# Visible Areas	MAPE (%)	RMSE	MAE
1	21.48	208.43	136.43
2	21.50	207.39	138.39
4	18.76	184.41	119.42
8	16.29	160.03	101.93

how long a trip will take with a certain level of confidence. Taking the trip in the first figure as an example, we can choose different probability thresholds with different demands, e.g., the trip will arrive in 8minute with about 80% probability. Besides, we can observe that for short (covering fewer areas) trips, the uncertainty of the model changes more drastically, requiring multiple Monte-Carlo sampling. The reason is that short trips are more significantly influenced by road conditions and human factors, while long trips can take the changes into account when performing long-distance spatio-temporal modeling.

As the traffic state fluctuates during the day and night, we further divide the departure time into 2 h windows and investigate the estimation in different periods. For better visualization, we select trips that have similar travel time (about 10 min) and different departure time in Fig. 7. We can observe from the box plots that all the ground truth trip times are within the estimated distribution. Daytime (approximately 8:00–20:00) trips have larger interquartile ranges than nighttime trips, indicating greater uncertainty in the estimates due to instability associated with more traffic activity during the day. The model has a greater bias in the prediction of nighttime trips (The ground truth trip times are outside the interquartile range), probably due to less training data for nighttime trips.

E. Scalability by Inductive Learning

For a cross-area algorithm, in addition to accurate prediction and estimation with uncertainty, scalability to unseen areas also needs to be investigated. In this section, we examine this property by an inductive learning approach. Specifically, we assume that only some areas of the city are visible and can be used to train cross-area models. For those unseen areas, we directly deploy the trained model for testing. As illustrated in Fig. 8, from left to right, we set the number of visible areas to 1, 2, 4, 8 in training, respectively. Those visible areas are trained by FL training as mentioned in Sec. IV-D1, and unseen areas utilize the global shared model as the local model in the testing stage. Table IV summarizes the result with a different number of visible areas in training on Chengdu Dataset. The simulation results are consistent with the intuition that the accuracy of the prediction increases as the number of visible areas increases. It is worth noting that when only half of the city areas participate in the cross-area travel time prediction training, the MAPE is able to reach 18.76%, which is a mere drop of 2.47% compared to the scenario where all areas are engaged (16.29%). Consequently, the above results show that our algorithm can easily scale to unknown areas and achieve promising performance.

VI. CONCLUSION

In this paper, we propose a novel cross-area travel time uncertainty estimation algorithm. This method integrates Bayesian neural networks to extract fine-grained spatio-temporal features to predict the uncertainty of travel times with the ability of privacy preservation. Particularly, the trajectory data is firstly mapped to the road network and reconstructs the speed distribution of the road network. The proposed model uses map-matched road segment features together with other external features as inputs to a novel travel time estimator. To further estimate the travel time uncertainty, we adopt Monte-Carlo dropout methods to obtain the travel time distribution. Finally, federated learning is incorporated to protect data privacy across different areas. The proposed scheme is among the pioneering works in estimating the cross-area travel time uncertainty with federated learning technology. It addresses a few crucial challenges of TTE, namely, cross-area travel time uncertainty estimation.

To evaluate the performance of the proposed method, we conducted a series of comprehensive case studies on two real-world trajectory datasets, whose results demonstrate the superiority of the proposed method. In addition, we test the uncertainty estimation performance of the model to reveal the robustness of the algorithm for complex traffic scenarios and user requirements. Finally, we perform a hyperparameter test and a scalability test, revealing that the proposed method is capable of scaling to unknwon areas without data and multi-area deployment.

In the future, we will focus on area data sparsity issues, e.g., the distribution and the total amount of data are non-independently and identically distributed among different areas. In addition, we also plan to study the scalability of the algorithm to cope with the deployment of ultra-large scale urban scenarios.

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] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.

[4] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, Aug. 2014, pp. 25–34.

[5] W. Dong, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? Estimating travel time based on deep neural networks," in *Proc. 32nd AAAI Conf. Artif. Intell.* New Orleans, LA, USA: AAAI Press, 2018, pp. 2500–2507.

[6] J. J. Q. Yu, "Citywide estimation of travel time distributions with Bayesian deep graph learning," *IEEE Trans. Knowl. Data Eng.*, early access, Oct. 6, 2021, doi: 10.1109/TKDE.2021.3117986.

[7] S. Maiti, A. Pal, A. Pal, T. Chattopadhyay, and A. Mukherjee, "Historical data based real time prediction of vehicle arrival time," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Qingdao, China, Oct. 2014, pp. 1837–1842.

[8] H. Wang, X. Tang, Y.-H. Kuo, D. Kifer, and Z. Li, "A simple baseline for travel time estimation using large-scale trip data," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–22, Mar. 2019.

[9] W.-C. Lee, W. Si, L.-J. Chen, and M. C. Chen, "HTTP: A new framework for bus travel time prediction based on historical trajectories," in *Proc. 20th Int. Conf. Adv. Geograph. Inf. Syst. (SIGSPATIAL)*. Redondo Beach, CA, USA: Association for Computing Machinery, 2012, pp. 279–288.

[10] C. de Fabritiis, R. Ragona, and G. Valenti, "Traffic estimation and prediction based on real time floating car data," in *Proc. 11th Int. IEEE Conf. Intell. Transp. Syst.*, Beijing, China, Oct. 2008, pp. 197–203.

[11] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, "Multi-task representation learning for travel time estimation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. London, U.K.: Association for Computing Machinery, Jul. 2018, pp. 1695–1704.

[12] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. London, U.K.: Association for Computing Machinery, Jul. 2018, pp. 858–866.

[13] Y. Shen, C. Jin, J. Hua, and D. Huang, "TTPNet: A neural network for travel time prediction based on tensor decomposition and graph embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4514–4526, Sep. 2022.

[14] S. Susilawati, M. A. Taylor, and S. V. C. Somenahalli, "Distributions of travel time variability on urban roads," *J. Adv. Transp.*, vol. 47, no. 8, pp. 720–736, Dec. 2013.

[15] F. Liu, D. Wang, and Z.-Q. Xu, "Privacy-preserving travel time prediction with uncertainty using GPS trace data," *IEEE Trans. Mobile Comput.*, early access, Apr. 24, 2021, doi: 10.1109/TMC.2021.3074865.

[16] Council of European Union. (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council on the Protection of Natural Persons With Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation)*. European Commission. [Online]. Available: https://gdpr-info.eu/

[17] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, pp. 1–55, Sep. 2014.

[18] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, "Traffic speed prediction and congestion source exploration: A deep learning method," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*. Barcelona, Spanish, Dec. 2016, pp. 499–508.

[19] H. Zhang, H. Wu, W. Sun, and B. Zheng, "DeepTravel: A neural network based travel time estimation model with auxiliary supervision," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Stockholm, Sweden, Jul. 2018, pp. 3655–3661.

[20] Y. Sun *et al.*, "CoDriver ETA: Combine driver information in estimated time of arrival by driving style learning auxiliary task," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 5, pp. 1–12, Dec. 2020.

[21] K. Fu, F. Meng, J. Ye, and Z. Wang, *CompactETA: A Fast Inference System for Travel Time Prediction*. Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 3337–3345.

[22] Y. Sun, K. Fu, Z. Wang, C. Zhang, and J. Ye, "Road network metric learning for estimated time of arrival," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Milan, Italy, Jan. 2021, pp. 1820–1827.

[23] H. Hong *et al.*, "HETETA: Heterogeneous information network embedding for estimating time of arrival," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. Virtual Event, CA, USA: Association for Computing Machinery, Aug. 2020, pp. 2444–2454.

[24] C. Markos, J. J. Q. Yu, and R. Y. D. Xu, "Capturing uncertainty in unsupervised GPS trajectory segmentation using Bayesian deep learning," in *Proc. 35th AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 1, pp. 390–398.

[25] Y. Wu and J. J. Q. Yu, "A Bayesian learning network for traffic speed forecasting with uncertainty quantification," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–7.

[26] T. Mallick, P. Balaprakash, E. Rask, and J. Macfarlane, "Transfer learning with graph neural networks for short-term highway traffic forecasting," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 10367–10374.

[27] S. Di, H. Zhang, C.-G. Li, X. Mei, D. Prokhorov, and H. Ling, "Cross-domain traffic scene understanding: A dense correspondence-based transfer learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 745–757, Mar. 2018.

[28] Y. Liu, J. J. Q. Yu, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7751–7763, Aug. 2020.

[29] Y. Zhu, Y. Liu, J. J. Q. Yu, and X. Yuan, "Semi-supervised federated learning for travel mode identification from GPS trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 1–12, Aug. 2021.

[30] Y. Zhu, S. Zhang, Y. Liu, D. Niyato, and J. J. Q. Yu, "Robust federated learning approach for travel mode identification from non-IID GPS trajectories," in *Proc. IEEE 26th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2020, pp. 585–592.

[31] J. S. Ng *et al.*, "Joint auction-coalition formation framework for communication-efficient federated learning in UAV-enabled Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2326–2344, Apr. 2021.

[32] W. Y. B. Lim *et al.*, "Towards federated learning in UAV-enabled Internet of Vehicles: A multi-dimensional contract-matching approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5140–5154, Aug. 2021.

[33] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[34] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, vol. 54, A. Singh and J. Zhu, Eds., Apr. 2017, pp. 1273–1282.

[35] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 72–80, Feb. 2020.

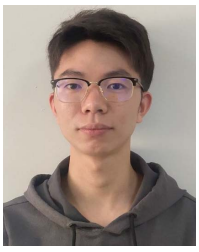
[36] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[37] C. Yang and G. Gidófalvi, "Fast map matching, an algorithm integrating hidden Markov model with precomputation," *Int. J. Geograph. Inf. Sci.*, vol. 32, no. 3, pp. 547–570, 2018.

[38] C. 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 *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst.*, Anchorage, AK, USA, Sep. 2012, pp. 776–781.

[39] X. Song, C. Zhang, and J. J. Q. Yu, "Learn travel time distribution with graph deep learning and generative adversarial network," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Indianapolis, IN, USA, Sep. 2021, pp. 1385–1390.

[40] 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, May 2020.
[41] X. Chen, Z. He, and L. Sun, "A Bayesian tensor decomposition approach for spatiotemporal traffic data imputation," *Transp. Res. C, Emerg. Technol.*, vol. 98, pp. 73–84, Jan. 2019.
[42] X. Chen and L. Sun, "Bayesian temporal factorization for multidimensional time series prediction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 4659–4673, Mar. 2021.
[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[44] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, Banff, AB, Canada, 2014, pp. 1–14.
[45] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.
[46] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.
[47] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu, and J. He, "Short-term load forecasting with deep residual networks," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3943–3952, Jul. 2019.
[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015, pp. 1–15.
[49] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.
[50] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794.



Yuanshao Zhu (Member, IEEE) received the B.Eng. degree in communication engineering from Shandong University, Weihai, China, in 2019, and the M.S. degree in electrical science and technology from the Southern University of Science and Technology, Shenzhen, China, in 2022. He is currently pursuing the Ph.D. degree with the Southern University of Science and Technology and the City University of Hong Kong joint program. His research interests include deep learning in smart city, intelligent transportation systems, and federated learning.

Yongchao Ye (Student Member, IEEE) received the B.Eng. degree in computer science and technology from Ningbo University, Ningbo, China, in 2020. He is currently pursuing the master's degree with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. His research interests include spatio-temporal data mining in smart city, intelligent transportation systems, and federated learning.

Yi Liu (Graduate Student Member, IEEE) received the B.Eng. degree from Heilongjiang University, China, in 2019. His research interests include security and privacy in smart city and edge computing, deep learning, intelligent transportation systems, and federated learning.

James J. Q. Yu (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees in electrical and electronic engineering from The University of Hong Kong, Pokfulam, Hong Kong, in 2011 and 2015, respectively. He is currently an Assistant Professor at the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China, and an Honorary Assistant Professor at the Department of Electrical and Electronic Engineering, The University of Hong Kong, where he was a Post-Doctoral Fellow, from 2015 to 2018. His general research interests are in smart city and urban computing, deep learning, intelligent transportation systems, and smart energy systems. His work is now mainly on forecasting and decision making of future transportation systems and basic artificial intelligence techniques for industrial applications. He was ranked World's Top 2% Scientists of 2019 and 2020 by Stanford University. He is an Editor of the *IET Smart Cities* journal.