CRATE: Privacy-preserving Travel Time Estimation

James Jianqiao Yu, *Senior Member, IEEE*

Abstract—Travel Time Estimation (TTE) stands as a cornerstone of efficient transportation systems. However, the critical imperative of privacy preservation within the TTE context remains notably underexplored. This gap underscores the pressing necessity for innovative solutions that prioritize the safeguarding of users' geo-privacy, particularly in light of the expanding prevalence of data-driven TTE algorithms. In this paper, a novel privacy-preserving TTE framework, CRATE, is proposed to ensure comprehensive privacy preservation for TTE without compromising service quality. CRATE achieves this objective by identifying random routes within a transportation network that yield identical travel times to the actual, privacy-rich route. This is accomplished through exploiting the embedding representations for road segments and routes, followed by the development of a highly efficient heuristic for random route generation. Furthermore, a travel time aggregation and calibration model is devised to enhance estimation accuracy while upholding user privacy. Case studies conducted on three real-world vehicular trajectory datasets demonstrate that CRATE attains comparable estimation accuracy to state-of-the-art non-privacy-preserving TTE algorithms while maintaining strict privacy protection. Additionally, CRATE's efficiency is showcased through deployment on both high- and low-end mobile handsets spanning the past decade.

Index Terms—Travel time estimation, privacy preservation, representation learning, mobility trajectory analysis, data mining.

1 Introduction

In the contemporary landscape of transportation, Travel Time Estimation (TTE) has emerged as a vital component underpinning the efficiency and efficacy of modern transport networks and intelligent transportation systems [1], [2]. TTE serves as one of the cornerstones in facilitating smooth and optimized travel experiences by providing real-time insights into the duration required to traverse along routes between specific locations [3], [4]. Whether it's for daily commutes, commercial logistics, or emergency response planning, accurate TTE is indispensable for effective route selection, congestion mitigation, and optimization of transportation resources [5].

The advent of mobile devices equipped with location tracking capabilities, combined with the ubiquity of navigation apps and GPS-enabled services, has resulted in an unprecedented abundance of trajectory data generated by individual users [6], [7]. In the past decade, the evolution of TTE algorithms has been significantly influenced by the proliferation of these crowdsourced trajectory data [8], [9]. These data-driven approaches encompass a spectrum of techniques, including machine learning and deep learning algorithms, statistical models, and tensor-based methodologies, which all exploit the spatiotemporal characteristics of trajectory data to infer travel times [10]. For instance, the industry-leading ride-hailing company DiDi Chuxing proposed a Wide-Deep-Recurrent (WDR) learning model designed to predict travel times based on floating-car data. Departing from conventional methods, the study formulates TTE as a spatial-temporal regression problem [1]. Through the integration of wide linear models, deep neural networks, and recurrent neural networks, the model achieves unparalleled accuracy in estimating travel times along predefined routes at specific departure times. This algorithm is not standalone, with a large array of novel algorithms in the past years emerging to push the limit of estimation accuracy [10].

However, amidst the progress in TTE methodologies and the reliance on crowdsourced trajectory data, a critical issue has remained largely unaddressed: the pervasive concerns surrounding user privacy [11]. While users willingly contribute their trajectory data to TTE service providers in exchange for the benefits of optimized route planning and real-time traffic updates, the implications of this data sharing on individual privacy are profound and multifaceted [12], [13]. At the heart of the privacy dilemma lies the inherent sensitivity of location-based data. Trajectory data, comprising information about users' starting points, destinations, intermediate waypoints, and timestamps, inherently encapsulates intimate details about individuals' movements and activities [14]. This wealth of information, when aggregated and analyzed, can unveil not only routine travel patterns but also sensitive aspects of users' lives, including their residential addresses, workplace locations, recreational activities, social interactions, and even medical appointments or other confidential engagements.

Further, the aggregation and analysis of trajectory data from multiple users raise concerns about group privacy and collective anonymity [13], [15], [16]. Even if individual trajectories are anonymized or pseudo-nonymized, the aggregation of data across a diverse user base can lead to the identification of unique travel patterns and behaviors, thereby compromising the privacy of entire user cohorts. This collective privacy risk is emphasized by the growing sophistication of data analytics techniques, which enable

James Jianqiao Yu is with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China, and with the Department of Computer Science, University of York, York YO10 5GH, United Kingdom (email: jqyu@ieee.org).

service providers to extract non-trivial insights from seemingly anonymized datasets [17].

Despite the importance of TTE and the increasing emphasis on privacy preservation, the realm of privacy-preserving TTE (PPTTE) has garnered considerably less attention from the research community [11]. This relative neglect can be attributed, in part, to the inherent challenge of striking a delicate balance between the utility of TTE services and the imperative to safeguard user privacy. Literature [11] is among the first attempts towards PPTTE, in which the investigated region is geographically divided into multiple areas. Cross-area privacy-preservation is achieved through federated learning. Nonetheless, this approach neither protects data within areas nor is robust to coalitions. This aforementioned work, along with others, implies the current challenges of TTE:

- **Neglect of Privacy Preservation**: Privacy preservation is often sidelined in TTE research despite achieving exceptional performance, necessitating a reorientation towards balancing privacy concerns with performance metrics.
- **Complexity of Privacy-Preserving Approaches**: PPTTE methods based on federated learning face hurdles like heavy client computation, non-IID assumptions, and high communication overhead, impeding practical deployment and scalability.
- **Business Model Misalignment**: Local-side PPTTE approaches require server distribution of proprietary traffic data to clients, conflicting with established business models and data governance frameworks, necessitating solutions that align with stakeholder needs.

To bridge these research challenges, we propose **C**ontrastive **R**oute-**A**dvised **T**ravel Time **E**stimation (CRATE), a full privacy-preserving travel time estimation solution. The design of CRATE is based on an intuitive hypothesis that for any arbitrary route in a transportation network, there must be numerous randomly generated routes sharing the same travel time. CRATE discovers these random routes, called contrastive routes, by learning the road segment and route embeddings via trajectory-data-free self-supervised learning and an iterative contrastive route generation heuristic. Additionally, CRATE devises auxiliary deep models for travel time aggregation and calibration to achieve outstanding estimation accuracy, which is empirically demonstrated by real-world datasets from multiple sources.

The major contributions of this work are fourfold:

- We propose robust embedding models for road segments and routes in the context of TTE, which precisely preserve the semantic similarity of road segments and time similarity of routes.
- We devise a highly efficient contrastive route generation heuristic based on the aforementioned embeddings, thereby enabling CRATE to estimate travel times.
- We develop a lightweight deep learning model to aggregate and calibration the travel times of contrastive routes to achieve highly accurate estimations.
- We validate the efficacy of the proposed CRATE on three real-world datasets in different countries and by different data providers. Results indicate the significance of CRATE compared with state of the arts.

The remainder of this paper is organized as follows. Section 2 presents a brief literature review to the current landscape of TTE and PPTTE. Section 3 introduces preliminary definitions and the formulation of PPTTE. Section 4 overviews and elaborates the proposed CRATE algorithm. Section 5 demonstrates the configurations and results of a set of comprehensive case studies. Finally, this paper concludes in Section 6.

2 Previous Work

In this section, we review the state-of-the-art data-driven TTE techniques and current efforts to PPTTE. For a more comprehensive review of the general landscape of TTE, readers are referenced to [10].

2.1 Data-driven Travel Time Estimation

Contemporary data-driven TTE techniques can be generally grouped into two categories, namely, tensor-based and learning-based methods. Particularly, tensor-based methods are among promising methodologies for modeling variables in transportation systems due to their ability to represent multivariate relationships within high-dimensional arrays of numerical data, commonly named tensors. Leveraging tensors, researchers construct high-order tensors to address TTE challenges, as demonstrated by [18], which constructs a third-order tensor representing travel times of drivers on specific roads at distinct times. However, inherent data sparsity issues arise due to many road segments remaining untraversed during significant time slots. To mitigate this, tensor decomposition techniques are employed to reduce dimensionality, fill sparse data, and uncover implicit relationships, as exemplified by the context-aware tensor decomposition approach proposed by [18]. Similarly, [19] introduces Probabilistic Traffic Condition Clustering, utilizing tensor-based models alongside context-aware tensor decomposition to accurately estimate missing entries, while [20] incorporates congestion levels into their tensor model and proposes a coupled tensor decomposition algorithm enhanced with point-of-interest features for improved missing data recovery. Furthermore, tensor-based models are combined with learning-based approaches for feature extraction, as evidenced by [21], which integrates non-negative tensor decomposition with a CNN-RNN model to extract both long-term and short-term travel speed features within the travel speed features layer.

Leveraging the merits of deep learning, learning-based TTE methods offers the potential to further enhance predictive accuracy and feature extraction capabilities for TTE. To list a few of the classics, [22] utilized a CNN-based approach and introduced PathCNN, a unique architecture incorporating diverse pooling techniques to handle heterogeneous sub-path TTE images and regulate convolution for enhanced spatial feature capture. Notably, temporal dependencies were effectively captured using a one-dimensional CNN model, demonstrating novelty compared to prevalent RNN usage for temporal feature extraction. Further, [23] divided high-resolution maps into grids, integrating trajectory data for pattern recognition via deep CNNs on morphological layout images, facilitating effective representations for subsequent predictions. In contrast, RNNs are adept at modeling temporal data due to their memorization capability, with applications like bidirectional LSTM (BiLSTM) further enhancing TTE accuracy through backward information utilization and auxiliary supervision mechanisms, as illustrated in [24]. DeepTTE [3] directly estimates travel time for entire paths from raw GPS sequences, using sliding windows to transform paths into sequences of sampled points. Inspired by natural language processing, WDR [1] concurrently trains wide linear, deep neural, and recurrent networks. The algorithm treats routes as sentences and road segments as words, utilizing both wide and deep models to capture route statistics and recurrent models for detailed segment characteristics, enabling accurate predictions.

2.2 Privacy-preserving Travel Time Estimation

Despite the abundance of research in data-driven TTE, the exploration of PPTTE remains in its infancy, underscoring its critical importance in addressing privacy concerns amidst the proliferation of TTE solutions. Federated learning, characterized by a decentralized architecture facilitating client-side data retention and collective model training on the local side, offers a promising avenue for PPTTE. In [11], Zhu *et al.* developed a cross-area privacy-preserving solution integrating federated learning, enabling tailored travel time estimators for distinct areas while upholding inter-area privacy safeguards. Addressing limitations of traditional federated learning, [25] introduced GOF-TTE, featuring both a base global model and fine-tuned personalized models, enhancing prediction accuracy across diverse client scenarios while preserving privacy. Despite the promising numerical results shown in the respective literature, neither solutions achieve highest user privacy level as individual trajectories and routing requests are aggregated in middle-layer facilities (area server in [11], transfer environment in [25]) before computation. From end users' perspective, their privacy-rich trajectories still need to be uploaded to third-party devices for computation instead of being kept local, raising individual privacy leakage concerns. Continued research efforts are necessary to realize the vision of full PPTTE that safeguards user privacy without compromising predictive performance.

Besides federate learning, homomorphic encryption and differential privacy are among the common privacy preservation techniques. Homomorphic encryption allows computations on encrypted data without needing to decrypt it [26], preserving data privacy but requiring substantial computational resources, which leads to inefficiencies. Additionally, it is unclear how to leverage homomorphic encryption to encode continuous spatial locations in trajectories and develop corresponding server-side algorithms to interpret the ciphertext and estimate travel time without decryption. Differential privacy introduces statistical noise into datasets to prevent the identification of individual entries but requires aggregating and perturbing multiple users' trajectories [27], which raises similar privacy concerns as federated learning since users must upload their real trajectories to a middle-layer aggregator [11], [25]. The proposed CRATE takes a different privacy preservation route by using a contrastive route generation mechanism to create synthetic yet plausible routes that mimic the travel time of real routes without exposing the actual paths of users. This method ensures data privacy while maintaining high accuracy and efficiency in travel time predictions, making it well-suited for real-time TTE in transportation systems.

A related but distinct research topic is privacy-preserving traffic prediction, which aims at forecast future traffic conditions using recently aggregated data with privacy preservation considerations [28]. Traffic prediction and TTE serve different purposes and require distinct data handling methods. The former aggregates data like vehicle counts or speeds from sensors, which inherently anonymizes individual data, making privacy preservation simpler [29], [30]. In contrast, TTE focuses on individual trajectories with geo-spatial data, requiring sophisticated privacy-preserving methods to prevent route reconstruction. Consequently, privacy-preserving traffic prediction algorithms cannot be directly applied to travel time estimation without significant modifications, as they are designed for different types of input and output data and privacy requirements.

3 Preliminaries

In this section, we first define important concepts used throughout the sequel of this paper. Then we formulate the privacy-preserving TTE problem and analyze its challenges.

3.1 Definitions

Definition 1 (Road Network)**.** In this work, the road network is represented by a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes (e.g., junctions and points of interest) in the network, and $\mathcal{E} \subseteq \{(u, v) \mid u, v \in \mathcal{V} \text{ and } u \neq v\}$ is the set of road segments connecting nodes in $\mathcal{V}$. To establish the connectivity between junctions in the road network, we derive a square adjacency matrix denoted as $\mathbf{A} = [a_{ij}] \in \mathbb{B}^{|\mathcal{V}|\times|\mathcal{V}|}$. Each element a_{ij} of the matrix is set to one if there exists a directed edge from node i to node j. We further employ two descriptor functions $\psi : \mathcal{V} \to \Psi$ and $\varphi : \mathcal{E} \to \Phi$ to map each node $v \in \mathcal{V}$ and edge $e \in \mathcal{E}$ to sets of pre-defined attributes denoted as Ψ and Φ, respectively, i.e., $\psi(v) \in \Psi$ and $\varphi(e) \in \Phi$. These attribute sets are designed to describe the explicit properties of the nodes and road segments within the road network.

Definition 2 (Route)**.** In this context, a route $\mathcal{R} = \{v_1, v_2, \dots, v_N\}$ is defined as a sequence of N traversed nodes $v_i \in \mathcal{V}$ in the road network. The departure time of the route is denoted as d_k, indicating the time at which the route was initiated. If the route is from the past data, it is associated with the corresponding total travel time denoted as t_k.

Definition 3 (Contrastive Route)**.** For privacy-preservation considerations, users' real routes $\mathcal{R}$ are not uploaded to the server. Instead, a privacy-preserving approach is employed where C contrastive routes $\widetilde{\mathcal{R}}_c$ are generated locally and uploaded for travel time estimation. Each contrastive route $\widetilde{\mathcal{R}}_c = \{v_{c,1}, v_{c,2}, \dots, v_{c,N_c}\}$ is composed of a sequence of N_c nodes selected from the road network to simulate a virtual traversal. The synthesis of contrastive routes aims to develop forged but realistic routes that have similar travel times to $\mathcal{R}$ while veiling users' location privacy.

Definition 4 (Road Segment and Route Representation Learning)**.** The goal of representation learning is to discover low-dimensional representations of road segments and routes to facilitate contrastive route generation. Given a road network $\mathcal{G}$, an embedding function $\omega : \mathcal{E} \rightarrow \mathbb{R}^D$ projects each road segment $e \in \mathcal{E}$ to a D-dimensional vector in the latent space $\mathbb{R}^D$, where $D \ll |\mathcal{E}|$. We further utilize another embedding function $\vartheta : \mathcal{R} \rightarrow \mathbb{R}^R$ to embed any route $\mathcal{R} \in \mathcal{O}$ or $\mathcal{R}_c \in \mathcal{O}$ into an R-dimensional embedding, where $\mathcal{O}$ is the set of all possible routes in $\mathcal{G}$ and $R \ll |\mathcal{O}|$. These two embedding functions are designed to capture informative and compact features of road segments and routes, respectively. The latter also enables effective generalization to new and unseen routes.

3.2 Problem Formulation and Analysis

The primary objective of this research is to provide users with travel time estimations that are both accurate and timely, all while maintaining their location privacy. To achieve this goal, we propose the CRATE framework and develop a set of encoding-decoding functions as follows:

$$\mathcal{L} = f^{\text{enc}}(\mathcal{R}), \tag{1a}$$

$$\mathcal{T} = f^{\text{est}}(\mathcal{L}), \tag{1b}$$

$$\tilde{t} = f^{\text{dec}}(\mathcal{T}), \tag{1c}$$

where $f^{\text{enc}} : \mathcal{R} \rightarrow \mathcal{L}$ and $f^{\text{dec}} : \mathcal{T} \rightarrow \mathbb{R}$ are the route encoding and travel time decoding functions, respectively, which are executed on the user's end. Additionally, the server performs the travel time estimation function $f^{\text{est}} : \mathcal{L} \rightarrow \mathcal{T}$ using obfuscated data $\mathcal{L}$. The resulting travel time estimation $\mathcal{T}$ is then communicated back to the user. Finally, $\tilde{t}$ represents the estimated travel time. By employing this design, the sensitive route information is encoded by f^{enc} before being uploaded as $\mathcal{L}$, ensuring that the uploaded data does not reveal the original route $\mathcal{R}$. This approach effectively safeguards user privacy while providing accurate travel time estimations.

The realization of the CRATE framework faces several challenges that need to be addressed in order to provide accurate and timely TTE services to edge users. These challenges are as follows:

1) Well-designed and light-weighted neural network models are essential for effective and efficient representation learning on the edge.
2) Past route data with travel time annotations are scarce and biased on each user's end, hindering large-scale local model training.
3) Mobile TTE users are likely to concerns the energy and cellular data consumptions, preventing them from training deep models.
4) Due to privacy concerns, users refrain from sharing their real routes with others or the server.
5) Due to business concerns, the service provider does not distribute real-time traffic states.

In CRATE, we propose innovative solutions to address all of the aforementioned challenges. The contrastive route generation mechanism, coupled with the privacy-preserving TTE scheme, enables accurate, timely, and cost-effective TTE while ensuring user privacy.

3.3 Privacy Model

System Setting. The system consists of a set of mobile clients $\mathcal{C} = \{C_1, \ldots, C_n\}$ and a cloud server S. Each client C_i holds a real route $\mathcal{R}_i^* \in \mathcal{O}$ and, before calling the travel time estimation service, executes a randomized route generation algorithm $\mathsf{GenDummy}(\cdot)$ that outputs a multiset of $d \geq 2$ contrastive (dummy) routes $\mathbf{R}_i = \{\widetilde{\mathcal{R}}_c^{(i,1)}, \ldots, \widetilde{\mathcal{R}}_c^{(i,d)}\} \subseteq \mathcal{O}$. Only $\mathbf{R}_i$ is uploaded to S; the real route $\mathcal{R}_i^*$ never leaves the client. The subsequent protocol follows the encoding/estimation/decoding pipeline of Sec. 3.2 using the union of all uploaded dummies $\bigcup_{i \in \mathcal{C}} \mathbf{R}_i$.
Adversary Capabilities. We consider a static, probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ that may corrupt any subset of clients $\mathcal{C}$ and/or the server S before the protocol starts. For every corrupted party, $\mathcal{A}$ obtains its complete internal state (including real routes of corrupted clients) and may arbitrarily deviate from the prescribed protocol. Parties not controlled by $\mathcal{A}$ are termed honest. The adversary's view is denoted $\mathsf{View}_{\mathcal{A}}$ and consists of: (1) public system parameters, (2) the internal states of all corrupted parties, and (3) all messages exchanged during the protocol execution, including the uploaded dummy sets $\mathbf{R}_i$ of honest clients and the server's responses.
Privacy Goal. Intuitively, $\mathcal{A}$ must learn nothing about the real route $\mathcal{R}_i^*$ of any honest client C_i beyond what was already known a priori. We formalize this via a standard semantic-security definition.

Definition 5 (Route Privacy)**.** CRATE is said to provide (λ)-route privacy if for every PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that

$$\left(\mathsf{View}_{\mathcal{A}}^{\text{Real}}\right) \overset{c}{\approx} \left(\mathsf{View}_{\mathcal{S}}^{\text{Dummy}}\right),$$

where $\overset{c}{\approx}$ denotes computational indistinguishability with security parameter λ. The real view is generated from an execution in which each honest client runs $\mathsf{GenDummy}(\mathcal{R}_i^*)$ on its true route $\mathcal{R}_i^*$. The simulated view is produced by $\mathcal{S}$ given only the public parameters and the internal states of the corrupted parties but without any information about $\mathcal{R}_i^*$ for honest C_i.

Def. 5 guarantees that an adversary controlling the server and any coalition of colluding clients cannot infer any additional information about an honest user's real route from the dummies it observes, because the adversary's view could have been generated without access to that route. The model deliberately focuses only on confidentiality; it does not cover authenticity or verifiable integrity of the uploads — i.e., we accept that malicious clients may submit arbitrary routes as long as honest users' privacy is preserved.

4 Privacy-preserving Contrastive Route Advised Travel Time Estimation

In this section, we introduce the proposed CRATE algorithm for PPTTE. CRATE is essentially a server-client model,

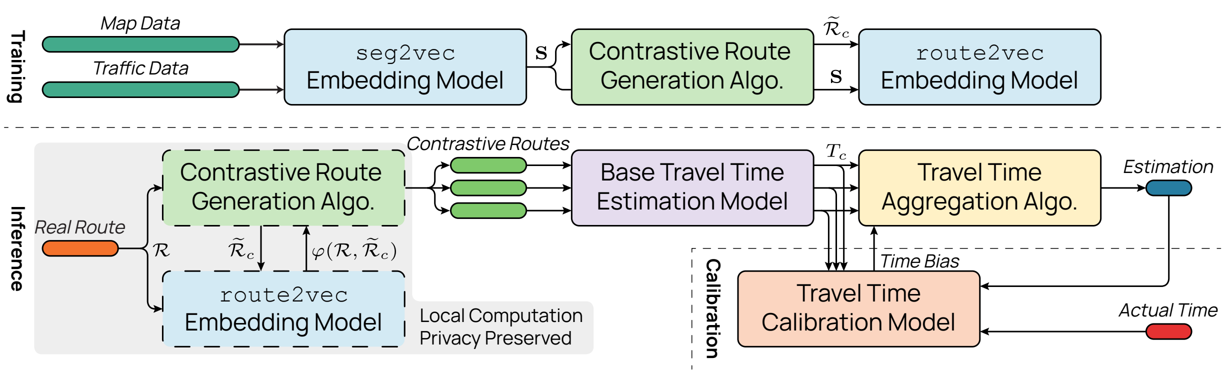
Training
Map Data
Traffic Data
seg2vec Embedding Model
S
Contrastive Route Generation Algo.
$\widetilde{\mathcal{R}}_c$
S
route2vec Embedding Model
Inference
Real Route
$\mathcal{R}$
Contrastive Route Generation Algo.
$\widetilde{\mathcal{R}}_c$
$\varphi(\mathcal{R}, \widetilde{\mathcal{R}}_c)$
route2vec Embedding Model
Local Computation Privacy Preserved
Contrastive Routes
Base Travel Time Estimation Model
T_c
Travel Time Aggregation Algo.
Estimation
Time Bias
Calibration
Travel Time Calibration Model
Actual Time

Fig. 1. Overview of CRATE. The framework comprises three execution phases. The execution of CRATE comprises three phases: training, inference, and calibration. The two blocks with dashed border in the inference phase are identical to corresponding ones in the training phase.

where the clients (TTE users) are responsible for the execution of f^{enc} and f^{dec} in (1) to generate random routes and the final travel time, respectively. The server is in charge of calculating the travel times of random routes (f^{est}), therefore exclude the involvement of users' privacy data. In the following, we first present an overview of CRATE and its training/inference phases. Then we elaborate the design of all constituting components of CRATE.

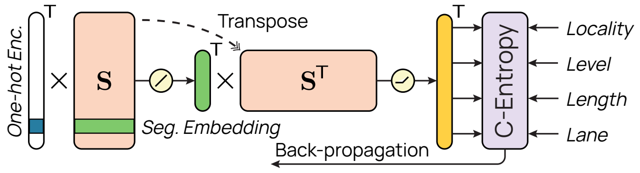

Fig. 2. Representation learning model of road segments, `seg2vec`.

4.1 Overview

Fig. 1 illustrates the overarching framework of the proposed CRATE approach for privacy-preserving travel time estimation. The execution of CRATE unfolds through three distinct phases, namely, an exclusive offline training phase conducted solely on the server, an online inference phase involving both the server and the clients, and an online calibration phase also involving both parties.

During the initial model training phase (top part of Fig. 1), the TTE server employs explicit road network data and synthesized routes to train a `seg2vec` model, which serves as a road segment embedding model. Subsequently, a contrastive route embedding (`route2vec`) model is constructed based on the trained `seg2vec` model. The trained `route2vec` model, along with a contrastive route generation algorithm, is disseminated to each user to facilitate online TTE.

Moving on to the second online inference phase (main lower part of Fig. 1), edge TTE users, acting as clients, employ the contrastive route generation algorithm and `route2vec` locally to generate contrastive routes derived from their real ones. These contrastive routes are then uploaded to the server, allowing for the estimation of their respective travel times using any available TTE models. Subsequently, the calculated times and relevant information are communicated back to the users, enabling the implementation of a contrastive route travel time aggregation scheme. This scheme ultimately produces the estimated travel time for the real routes.

The framework incorporates an additional third online calibration phase (lower right part of Fig. 1) aimed at refining the estimated travel times through feedback calibrations. Users may upload their estimated travel times as well as the corresponding actual travel times. While keeping the data related to the previous contrastive routes, the server leverages this information to train a travel time calibration model, which generates time offsets. This model is subsequently utilized to produce offsets for new contrastive routes, enhancing the accuracy of the local travel time aggregation scheme in estimating travel times.

CRATE resolves the challenges mentioned in Section 1 and Section 3.2 by exclusively conducting the intensive model training process on the server, eliminating the need for users to undergo local model training as required in Federated Learning-based approaches. Additionally, throughout all three phases of CRATE, users are not obliged to upload their present or previous personal routes, and the server is not required to distribute road network data. As a result, concerns regarding privacy and the potential exposure of proprietary data are effectively addressed. Indeed, CRATE is an incremental unsupervised learning framework that does not rely on historical route and travel time information for initialization, although the inclusion of such data can significantly enhance the initial estimation performance. The accuracy of the estimations is ensured through the employment of the calibration model, which leverages real-time travel time (instead of route) data provided by users. This online learning design significantly enhances the overall performance of CRATE while simultaneously safeguarding user privacy.

From the modular design standpoint, CRATE comprises five key components, namely, the `seg2vec` model, the contrastive route generation algorithm, the `route2vec` model, the travel time estimation and aggregation scheme, and the travel time calibration model. The subsequent subsections provide detailed introductions and discussions of these components.

4.2 Representation Learning of Road Segments

In this section, we introduce the first component of CRATE, seg2vec, which is utilized to acquire embeddings of road segments. The primary goal of this representation learning endeavor is to significantly condense the dimensionality of the road segment domain, from $|\mathcal{E}|$ to D, while simultaneously preserving the similarity (and disparities) among segments. Given that CRATE's central principle revolves around the development of contrastive routes akin to the actual ones, the similarity of road segments is defined by their adjacency and homogeneity relationships. seg2vec is designed to generate embeddings in such a manner that similar road segments find themselves proximate within the embedding space $\mathbb{R}^D$.

Fig. 2 presents the representation learning model of road segments with seg2vec. As shown in the figure, seg2vec functions as a multi-task classifier aiming at developing the similarity score of pairs of road segments. This approach is rooted in and significantly expands upon the Skip-gram technique utilized in natural language processing [31]. At the core of this model lies the trainable embedding matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{E}| \times D}$. Given a one-hot vector $\mathbf{u} \in \mathbb{R}^{|\mathcal{E}|}$ representing the segment $u \in \mathcal{E}$, seg2vec employs the only non-zero row within $\mathbf{u}^\mathsf{T}\mathbf{S}$ as the embedding for u, denoted as $\mathbf{e}_u \in \mathbb{R}^D$. The embedding matrix $\mathbf{S}$ is trained on the similarity of road segments. Particularly, we compute $\mathbf{s}_u = \{s_{u,v}\} = \mathbf{e}_u^\mathsf{T}\mathbf{S}^\mathsf{T} \in \mathbb{R}^{|\mathcal{E}|}$ to generate a vector of road segment similarity scores, where each entry $s_{u,v}$ refers to the similarity between segments u and v.

Considering the proximity and homogeneity of road segments, the target similarity score $\hat{s}_{u,v} = \langle \hat{s}^{\text{loc}}_{u,v}, \hat{s}^{\text{lvl}}_{u,v}, \hat{s}^{\text{len}}_{u,v}, \hat{s}^{\text{lne}}_{u,v} \rangle$ is defined by four facets:

1) Locality $\hat{s}^{\text{loc}}_{u,v} \in \mathbb{B}$ is set to 1 if u and v are within L hops in the road network.
2) Road level $\hat{s}^{\text{lvl}}_{u,v} \in \mathbb{B}$ is set to 1 if u and v are of the same road level (e.g., motorway, truck, primary, secondary, tertiary, and residential as classified by OpenStreetMap).
3) Segment length $\hat{s}^{\text{len}}_{u,v} \in (0, 1]$ is the ratio of the lengths of u and v.
4) Road lane $\hat{s}^{\text{lne}}_{u,v} \in \mathbb{B}$ is set to 1 if u and v has the same number of lanes.

With the quantified similarity scores, the embeddings $\mathbf{S}$ can be trained through back-propagation and the following binary cross-entropy loss:

$$\begin{aligned}\text{Loss}(s_{u,v}, \hat{s}_{u,v}) &= \text{BCE}(s_{u,v}, \hat{s}^{\text{loc}}_{u,v}) + \text{BCE}(s_{u,v}, \hat{s}^{\text{lvl}}_{u,v}) \\ &\quad + \text{BCE}(s_{u,v}, \hat{s}^{\text{len}}_{u,v}) + \text{BCE}(s_{u,v}, \hat{s}^{\text{lne}}_{u,v}), \quad \text{(2a)}\end{aligned}$$

$$\text{BCE}(x, \hat{x}) = \hat{x} \log \frac{1}{1+e^{-x}} + (1-\hat{x}) \log \frac{e^{-x}}{1+e^{-x}}. \quad \text{(2b)}$$

The training of $\mathbf{S}$ is accomplished through the multi-objective optimization problem of minimizing (2) using the RMSProp optimizer. In essence, if segments u and v satisfy one or more of the aforementioned similarity metrics, $\mathbf{e}_u$ and $\mathbf{e}_v$ will gradually converge within the embedding space, thereby producing similar embeddings.

4.3 Contrastive Route Generation

With the aim of generating contrasting routes that exhibit similar travel times to the real route, CRATE addresses this question by dividing it into two distinct aspects: the generation of arbitrary contrasting routes and the determination of route similarity in terms of travel time without explicit time information. The former is tackled through a tailor-made algorithm leveraging segment embeddings, and the latter will be expounded upon in the subsequent section with the introduction of route2vec.

The proposed algorithm for contrastive route generation encompasses three key elements: initial location, segment propagation rule, and termination criteria. Given an arbitrary real route $\mathcal{R}$, the algorithm randomly decides whether to initiate from the origin or the destination of the contrastive route to be generated. Subsequently, cosine similarities between the real origin/destination u (v_1 or v_N c.f. Definition 2) and all segments $v \in \mathcal{E}$ within the road network are computed, defined by

$$\cos(u, v) = \mathbf{e}_u \cdot \mathbf{e}_v / \|\mathbf{e}_u\| \|\mathbf{e}_v\|. \quad (3)$$

The contrastive origin/destination is randomly selected from $\mathcal{E}$ based on the softmax-ed cosine similarity as the probability:

$$\sigma(u, v) = e^{\cos(u,v)} / \sum\nolimits_{v \in \mathcal{E}} e^{\cos(u,v)}. \quad (4)$$

With the initial location of the contrastive route determined, the generation algorithm proceeds to propagate the route throughout the road network. Here we take the start-from-origin mode as an example. Given the real route $\mathcal{R}$ and a partially finished contrastive route $\widetilde{\mathcal{R}}_c$, the next segment of $\widetilde{\mathcal{R}}_c$ is randomly chosen from all feasible next-hop segments with a probability defined as follows:

$$\varsigma(\mathcal{R}, \widetilde{\mathcal{R}}_c, v) = \frac{e^{\cos(\|\mathcal{R}\|, \|\widetilde{\mathcal{R}}_c\| + \mathbf{e}_v)}}{\sum_{v \in \mathcal{E}(v_{c,N_c})} e^{\cos(\|\mathcal{R}\|, \|\widetilde{\mathcal{R}}_c\| + \mathbf{e}_v)}}, \quad (5)$$

where the vector norm of a route is defined as $\|\mathcal{R}\| = \sum_{u \in \mathcal{R}} \mathbf{e}_u$, and $\mathcal{E}(u)$ denotes the set of road segments that serve as next-hops from u. Principally, this probability makes the expanded contrastive route closely aligned with the real one in terms of their cosine similarity.

Given the iterative nature of the previous segment propagation, a termination criterion is vital for the route generation process. CRATE introduces two distinct termination criteria tailored for different purposes of the contrastive route generation algorithm: travel time-based and similarity-based. The former is employed at the server during the training of route2vec to be introduced subsequently, where the travel times of the real and contrastive routes can be estimated using historical data. In this context, route generation will cease if adding a new segment would exacerbate the discrepancy in travel time between $\mathcal{R}$ and $\widetilde{\mathcal{R}}_c$. The latter is utilized when employing route2vec for contrastive route inference at the edge, which will be elaborated on next.

It is important to note that the contrastive route generation algorithm proposed in this section is not exclusive. Instead, any heuristic or learning-based algorithm that takes a real route and the road network as inputs can be employed as long as it stochastically produces alternative routes. This leaves room for the possibility of more advanced generation algorithms in future research endeavors.

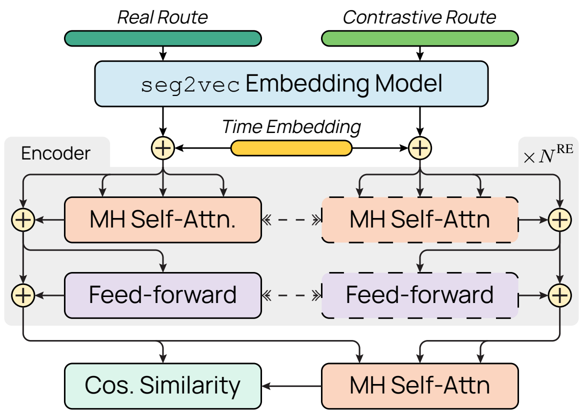

Fig. 3. Representation learning model of routes, `route2vec`.

4.4 Representation and Contrastive Learnings of Routes

Assuming we have a real route $\mathcal{R}$ and a set of randomly generated contrastive routes $\widetilde{\mathcal{R}}$, CRATE aims to select the most similar contrastive routes on the client side in terms of their unknown travel times for uploading to the TTE server. However, the representation learning model presented in Section 4.2 solely involves the static features of road segments and disregards their time-variant characteristics. To overcome this limitation, we propose a `route2vec` model to capture the hidden representation of routes.

Given an arbitrary route $\mathcal{R}$, `seg2vec` is first adopted to generate the corresponding sequence of road segment embeddings:

$$\texttt{seg2vec}(\mathcal{R}) = \{\mathbf{e}_{v_1}, \mathbf{e}_{v_2}, \dots, \mathbf{e}_{v_N}\}. \tag{6}$$

Subsequently, time features — encompassing absolute time and the sequence of road segments — are superposed on the embeddings, resulting in a pre-encoding representation $\mathbf{P}_{\mathcal{R}}$ of the route:

$$\mathbf{P}_{\mathcal{R}} = [\mathbf{e}_{v_1} + \mathbf{p}_1, \mathbf{e}_{v_2} + \mathbf{p}_2, \dots, \mathbf{e}_{v_N} + \mathbf{p}_N]^\mathsf{T} \in \mathbb{R}^{N\times D}, \tag{7a}$$
$$\mathbf{p}_i = \left(\mathrm{PE}[i] + T^{\mathrm{D}} + T^{\mathrm{W}} + T^{\mathrm{Y}}\right)/4, \tag{7b}$$

where $\mathrm{PE}[i]$ denotes the Positional Encoding of position i, and T^{D}, T^{W}, T^{Y} represent the cosine-encoded values for time-of-day, day-of-week, and week-of-year values, respectively. The (expected) departure time of $\mathcal{R}$ is utilized here to compute these absolute time features, while the positional encoding serves to differentiate between various segments within the route. Consequently, geographically identical routes at different time exhibit distinct representations.

`route2vec` further follows the principles of Transformer [32] and non-contrastive Siamese network [33], and incorporates a series of N^{RE} route encoder blocks to generate the route embedding based on $\mathcal{P}_{\mathcal{R}}$. Within each encoder block, the input $\mathbf{X}$ (initially $\mathbf{P}_{\mathcal{R}}$ and subsequently the output of the previous block) undergoes multi-head self-attention, allowing the generated representation to capture dependencies within the road segment sequence of $\mathcal{R}$. The multi-head self-attention is defined as follows:

$$\mathbf{Y} = \mathrm{softmax}\left(\frac{\mathbf{X}\mathbf{W}^{\mathrm{Q}} \cdot (\mathbf{X}\mathbf{W}^{\mathrm{K}})^\mathsf{T}}{\sqrt{D}}\right)\mathbf{X}\mathbf{W}^{\mathrm{V}}, \tag{8}$$

where $\mathbf{W}^{\mathrm{Q}}$, $\mathbf{W}^{\mathrm{K}}$, $\mathbf{W}^{\mathrm{V}}$ are trainable weight parameters of the shape $D \times D$. The output is then passed through a feed-forward neural network consisting of two linear transformations with Rectified Linear Unit (ReLU) [34] activation:

$$\mathbf{Z} = \mathrm{LN}\left(\mathrm{ReLU}(\widetilde{\mathbf{Y}}\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}^2 + \widetilde{\mathbf{Y}}\right), \tag{9a}$$
$$\widetilde{\mathbf{Y}} = \mathrm{LN}(\mathbf{Y} + \mathbf{X}), \tag{9b}$$

where LN denotes to layer normalization aiming to reduce the impact of covariant shift during model training. Additionally, $\mathbf{W}^1$, $\mathbf{W}^2$, $\mathbf{b}^1$, $\mathbf{b}^2$ represent trainable weight and bias parameters.

To compute the similarity between routes, we adopt a non-contrastive Siamese network architecture [33] as shown in Fig. 3 that generates comparable output vectors for different input route embeddings. The model is trained using the contrastive learning paradigm [35]. Given the real route $\mathcal{R}$ and one of its contrastive route $\widetilde{\mathcal{R}}_c$, their embeddings can be computed by using the aforementioned `route2vec` and denoted as $\mathbf{H}_0$ and $\mathbf{H}_{\tilde{c}}$, respectively. To fully leverage the inter-route segment interactions within $\mathcal{R}$ and $\widetilde{\mathcal{R}}_c$, we incorporate a cross-attention between $\mathbf{H}_0$ and $\mathbf{H}_{\tilde{c}}$, achieved by slightly modifying (8) as follows:

$$\widetilde{\mathbf{H}}_{\tilde{c}} = \mathrm{softmax}\left(\frac{\mathbf{H}_{\tilde{c}}\mathbf{W}^{\mathrm{Q}'} \cdot (\mathbf{H}_0\mathbf{W}^{\mathrm{K}'})^\mathsf{T}}{\sqrt{D}}\right)\mathbf{H}_0\mathbf{W}^{\mathrm{V}'}, \tag{10}$$

where $\mathbf{W}^{\mathrm{Q}'}$, $\mathbf{W}^{\mathrm{K}'}$, and $\mathbf{W}^{\mathrm{V}'}$ are trainable weight parameters.

Note that although $\mathbf{H}_0$ and $\mathbf{H}_{\tilde{c}}$ may have different shapes ($N \times D$ and $N_c \times D$, respectively), the cross-attention output $\widetilde{\mathbf{H}}_{\tilde{c}}$ shares the shape of $\mathbf{H}_{\tilde{c}}$, i.e., $N_c \times D$. Consequently, the similarity between $\mathcal{R}$ and $\widetilde{\mathcal{R}}_c$ can be computed as the averaged cosine similarity of all rows:

$$\varphi(\mathcal{R}, \widetilde{\mathcal{R}}_c) = \sum_{i=0}^{N_c-1} \cos(\widetilde{\mathbf{h}}_{\tilde{c},i}, \mathbf{h}_{\tilde{c},i})/N_c. \tag{11}$$

During training of the Siamese network on the server, the objective is to find optimal parameters such that routes with similar travel times receive high similarity scores according to (11). This is achieved by generating contrastive routes with similar travel times based on the road network and historical traffic speed (as described in Section 4.3). Given such routes, the following loss function is minimized:

$$\mathrm{Loss}(\mathcal{R}, \widetilde{\mathcal{R}}_c) = 1 - \varphi(\mathcal{R}, \widetilde{\mathcal{R}}_c) + \mathrm{NLL}(\widetilde{\mathbf{H}}_{\tilde{c}}, \mathbf{H}_{\tilde{c}}), \tag{12}$$

where $\mathrm{NLL}(\cdot,\cdot)$ denotes the negative log-likelihood.

Once the model is well-trained, it can be used to generate contrastive routes with similar travel times solely based on $\varphi(\cdot,\cdot)$ at the edge clients. In particular, the aforesaid route generation algorithm iteratively expands the generated route, and at each step, it calculates a new route similarity score using (11). When no higher similarity is achieved with the past ten added road segments, the generation process terminates, and the most similar contrastive route is selected and sent to the TTE server. In practice, CRATE produces N^{CR} contrastive routes to enhance the generalizability and stability of TTE results.

4.5 Contrastive Travel Time Estimation

After defining the contrastive route generation and representation learning components, we can now estimate the travel time of a real route with the generated contrastive ones. Once the N^{CR} contrastive routes are generated on the edge client, along with their route similarity scores as defined in (11), they are transmitted to the TTE server. The server utilizes existing TTE algorithms to calculate a travel time value T_c for each contrastive route. Finally, the edge client calculates the weighted arithmetic mean of these times, where the similarity scores serve as the weights. The resulting value is used as an estimate for the travel time of the real route $\mathcal{R}$.

Note that CRATE is not limited to the base TTE algorithm to be executed at the server side: any one as simple as historical average or as complex as the state-of-the-art deep learning models can be integrated with ease. This enables CRATE to fully utilize the past advancement of TTE algorithms and better achieve PPTTE. In the case studies to be presented in Section 5, we use the intuitive historical average algorithm to calculate the travel times of the contrastive routes for its simplicity and generalizability. Indeed, more advanced models that incorporate various external factors such as weather conditions, traffic patterns during rush hours, and real-time traffic updates can be integrated to further improve the TTE performance and make the system robust against various external conditions. Thus, while the historical average model serves as a straightforward and effective baseline, the flexibility of CRATE allows for the inclusion of comprehensive TTE models that leverage external data, significantly boosting the overall performance of the system.

4.6 Contrastive Travel Time Calibration

During the end-to-end training of CRATE using the previous data manipulation schemes, we observed non-stationary time shifts between the aggregated travel time and the ground truth. This error can be effectively alleviated by introducing a data-driven bias block, or as we call it, travel time calibration model. In particular, the calibration model consists of an additional two-layer feed-forward neural network with residual connections and layer normalization, as defined in (9). This neural network compresses the route embeddings of all contrastive routes along the road segment dimension. The result is then concatenated to the estimated travel time and the similarity score to produce the model input:

$$\mathbf{X}^{\mathrm{cal}} = \{\mathbf{x}_c^{\mathrm{cal}}\} \in \mathbb{R}^{N^{\mathrm{CR}} \times (D+2)}, \tag{13a}$$

$$\mathbf{x}_c^{\mathrm{cal}} = \left(\sum_{i=0}^{N_c-1} \widetilde{\mathbf{h}}_{\tilde{c},i} \middle\| \left[T_c, \varphi(\mathcal{R}, \widetilde{\mathcal{R}}_c) \right] \right)^{\mathsf{T}}. \tag{13b}$$

The network outputs a single time bias value, which is trained using an L2-norm loss function against the discrepancy between the aggregated travel time and the real one.

In summary, when a TTE request is sent to the server, it is also passed to the calibration model, which calculates a time bias. This time bias is sent back to the edge client and added to the aggregated estimated travel time. Later on, when this particular route is finished, the travel time difference is also uploaded to the model, enabling it to refine its parameters and provide more accurate time biases for future requests.

4.7 Privacy Analysis of CRATE

We now establish that CRATE achieves the privacy guarantee formalised in Def. 5. The core argument is that the only information about an honest client that ever leaves her device is the multiset of dummy routes $\mathbf{R}_i$ produced by the proposed contrastive route generation algorithm, termed CRG in the sequel. CRG draws randomness at three stages (origin/destination sampling, segment propagation, termination), each time using probabilities that depend only on the route embedding $\mathbf{e} \in \mathbb{R}^d$ produced by the `route2vec` model. Intuitively, `route2vec` is trained to preserve travel time rather than geometry, so many semantically distinct routes map to adjacent embedding vectors. For any two embeddings $\mathbf{e}, \mathbf{e}'$ with identical travel-time label, the softmax probabilities that CRG uses for origin/destination and segment propagation differ by at most $\mathsf{negl}(\lambda)$ in total variation distance. This follows from the high dimensionality of $\mathbf{e}$ and the cosine-similarity concentration phenomenon [36].

Provided that the dummy routes are distributed independently of the real route, the adversary's view can be perfectly simulated without access to the real route.

Lemma 1. For any pair of real routes $r_0, r_1 \in \mathcal{O}$ and every PPT distinguisher $\mathcal{D}$,

$$\left| \Pr\left[\mathcal{D}\big(\mathsf{CRG}(r_0)\big) = 1\right] - \Pr\left[\mathcal{D}\big(\mathsf{CRG}(r_1)\big) = 1\right] \right| \le \mathsf{negl}(\lambda), \tag{14}$$

where λ is the security parameter and $\mathsf{negl}(\cdot)$ is a negligible function.

Sketch. Algorithm CRG draws dummy routes $\mathbf{R}_i$ by pseudorandomly sampling from the public route pool $\mathcal{O}$, using only fresh local randomness and no component of the client's true route $\mathcal{R}_i^*$. Consequently, the joint geographical distribution of the resulting multiset $\mathbf{R}_i$ is identical — up to negligible pseudo-random generator error — no matter which $\mathbf{R}_i$ is given as input. Any PPT distinguisher therefore gains at most negligible advantage. ■

Theorem 1 (Privacy of CRATE)**.** Under Lem. 1, CRATE provides (λ)-trajectory privacy in the sense of Def. 5.

Proof. We construct a PPT simulator $\mathcal{S}$ that, given the public parameters and the internal states of all corrupted parties, outputs a view that is computationally indistinguishable from the adversary's real view $\mathsf{View}_{\mathcal{A}}^{\mathrm{Real}}$.

Simulator $\mathcal{S}$. For every corrupted client C_j, $\mathcal{S}$ inserts the genuine state supplied by the environment, including R_j and $\mathbf{R}_j$. For each honest client C_i, $\mathcal{S}$ samples a dummy multiset $\widetilde{\mathbf{R}}_i \leftarrow \mathsf{CSG}(\mathcal{R}_i^{\dagger})$ using an arbitrary fixed route $\mathcal{R}_i^{\dagger}$ and the same dummy-set size d. Because server-side processing (encoding, aggregation, travel-time estimation, decoding) depends solely on the union of uploaded dummies, $\mathcal{S}$ can execute the entire server program locally using $\big(\bigcup_i \widetilde{\mathbf{R}}_i\big) \cup \big(\bigcup_j \mathbf{R}_j\big)$, thereby generating all messages and outputs that $\mathcal{A}$ would observe.

Indistinguishability. The real and simulated views differ only in the dummy multisets originating from honest clients: $\mathbf{R}_i$ against $\widetilde{\mathbf{R}}_i$. By Lem. 1, these distributions

TABLE 1
Key Statistics of Datasets

Dataset	# Records	# Routes	Avg. Time	Sampling
Xi'an	1 536 641 807	6 733 235	399.05 s	2 s to 4 s
Chengdu	1 096 618 422	6 105 003	581.57 s	2 s to 4 s
Porto	83 423 824	1 710 990	942.39 s	15 s

are computationally indistinguishable for every C_i, hence for their concatenation across all honest clients. All subsequent protocol messages are deterministic or pseudorandom functions of those dummies; the hybrid/pseudorandom function argument extends indistinguishability to the entire transcript. Therefore, $\mathsf{View}_{\mathcal{A}}^{\text{Real}} \overset{c}{\approx} \mathsf{View}_{\mathcal{S}}^{\text{Sim}}$, satisfying Def. 5. ■

The theorem confirms that CRATE leaks no additional information about an honest users true route beyond what corrupted parties already know, even when the server is compromised and colludes with any subset of clients. The guarantee rests on the sole cryptographic assumption that CSG is input-oblivious (Lemma 1), which holds by construction in our implementation.

5 Case Studies

In this section, we perform a series of comprehensive case studies using three real-world trajectory datasets to evaluate the proposed CRATE. We begin by introducing the details of the datasets and the experiment configurations. Next, we conduct an extensive comparative study to assess the accuracy of travel time estimation achieved by CRATE while preserving privacy. We then deploy the centrally trained CRATE on several representative mobile devices to evaluate the usability of the proposed learning models. Additionally, we analyze the impact of hyper-parameters on CRATE's performance.Finally, we examine the generalizability of the contrastive travel time calibration model on different datasets.

5.1 System Configurations and Baselines

In our case studies, we utilize anonymized historical travel time index datasets provided by DiDi Chuxing. These datasets were collected by ride-hailing vehicles in two major cities in China, Xi'an (from October 1st, 2016 to November 30th, 2016) and Chengdu (from November 1st, 2016 to November 30th, 2016), with a non-static sampling interval ranging from 2 s to 4 s. Additionally, we use the taxi trajectory dataset from Porto, Portugal, collected from July 1st, 2013 to June 30th, 2014, with a sampling interval of 15 s. Table 1 records some key statistics of these three datasets. The topology and static attributes of the transportation networks are obtained from OpenStreetMap.

As described in Section 4.1, CRATE consists of three execution phases. The framework does not rely on historical trajectory data for training, except for the online learning of the calibration model. Therefore, we train the `seg2vec` model using the static open data of road networks until convergence, which is determined by observing no loss decrease for three consecutive epochs. We then map-match the trajectory data onto the road networks and derive the average traffic speed of the three tested cities. This information is used to train the `route2vec` model, where we randomly generate real and contrastive routes and compute their travel times using the averaged traffic speed. It is important to note that no real trajectory data (route or time) are used in this phase of training. Once these two models are well-trained, we apply CRATE chronologically to all trajectories in the three dataset and begin training the calibration model using the travel time discrepancy. We expect to observe an improvement in accuracy over time due to the calibration, which will be evaluated in Section 5.2. It is also worth mentioning that no trajectories are used at any stage of the training process, thereby preserving the location privacy.

To better simulate the practical usage of CRATE, we implement the TTE server on computing servers equipped with Intel Xeon E5 CPUs and nVidia RTX 2080Ti GPUs. The edge client part of CRATE, including the trained `route2vec` and the contrastive route generation algorithm, is deployed on a number of mobile devices with Snapdragon 430, 855, and 888 SoCs. We use PyTorch and TensorFlow for model implementation and training, and the deployment is achieved by the corresponding PyTorch Mobile and TFLite libraries in this setup. The mobile execution performance will be discussed in Section 5.3. We set the hyper-parameters of CRATE as follows, unless otherwise stated in Section 5.4: number of contrastive routes $N^{\text{CR}} = 3$, embedding dimensionality $D = 256$, number of stacked encoders $N^{\text{RE}} = 6$, number of heads in the multi-head self-attention is eight, and the hidden size of the feed-forward neural networks in the encoders and calibration model is 2048. The training mini-batch size is set to 128 for `seg2vec` and 32 for `route2vec`. Following contemporary TTE studies [4], [5], [11], we use the mean absolute percentage error (MAPE), root mean square error (RMSE), mean absolute error (MAE), and satisfaction rate of less than 15% (SR-15) error as accuracy metrics.

Finally, we compare the proposed CRATE with the following privacy-leaking and semi-privacy-preserving baseline methods in terms of TTE accuracy:

5.1.1 *Privacy-leaking baselines*

- **Historical Average (HA)** estimates travel time by summing up the historical average travel time of each road segment in a given route.
- **TEMP** [37] queries and averages trips with neighboring origins and destinations of a given route.
- **XGBoost** [38] adopts an ensemble of regressors for TTE.
- **Wide-Deep-Recurrent (WDR)** [1] is a state-of-the-art trajectory-based TTE model that combines wide, deep, and recurrent neural networks for estimation.
- **DeepTTE** [3] employs 1-D convolution and recurrent networks to estimate travel times from raw GPS trajectories and external features.

5.1.2 *Semi-privacy-preserving*

- **WDR-F** [11] utilizes standard federated learning to train WDR by partitioning the road network into eight regions. Privacy is preserved among multiple regions, but data within each region is unprotected.

TABLE 2
Performance Comparisons of CRATE and Baseline Algorithms

	Metric	CRATE	HA	TEMP	XGBoost	WDR	DeepTTE	WDR-F	DeepTTE-F	CATTUE	Geo-I
	Privacy?	✓	✗	✗	✗	✗	✗	✘	✘	✘	✘
Xi'an	MAPE (%)	14.45	26.51	22.38	21.35	17.66	13.59	20.69	19.11	16.33	15.28
	RMSE (s)	137.92	331.25	193.11	178.42	218.41	132.23	221.79	182.71	148.05	157.72
	MAE (s)	81.99	151.11	125.56	118.35	100.96	77.30	117.48	108.56	92.45	86.83
	SR-15 (%)	59.42	35.03	41.12	42.57	50.61	62.24	43.91	47.22	53.89	56.70
Chengdu	MAPE (%)	14.40	24.44	24.63	16.82	16.04	13.35	19.99	18.09	15.20	15.00
	RMSE (s)	132.85	444.84	172.26	109.96	239.66	235.34	325.58	269.72	165.30	273.22
	MAE (s)	59.47	105.21	98.61	65.32	68.54	57.89	85.91	77.22	63.80	65.13
	SR-15 (%)	59.54	37.74	37.52	52.31	54.44	63.04	45.18	49.32	57.06	57.47
Porto	MAPE (%)	14.58	25.56	26.22	23.65	21.88	15.19	23.01	20.36	18.54	16.61
	RMSE (s)	174.72	310.06	294.12	246.95	258.07	188.62	277.31	250.60	224.90	200.88
	MAE (s)	103.48	181.97	184.01	163.76	155.39	108.39	163.76	145.10	131.71	118.32
	SR-15 (%)	60.12	37.15	36.53	39.75	42.86	58.16	40.70	45.63	49.28	53.88

- **DeepTTE-F** is similar to WDR-F, with the base model changed to DeepTTE.
- **Cross-area travel time uncertainty estimation (CATTUE)** [11] quantifies the uncertainty of travel time data to assist accurate TTE with inter-region privacy preservation.
- **Geo-Indistinguishability (Geo-I)** [39] is a privacy-preserving method that uses differential privacy to protect user location data.

All baseline approaches are trained using configurations presented in the respective literature with the chronologically first 75% of the dataset (3:1 training-to-validation ratio), and are tested by the remaining 25% trajectories. It is important to note that none of these baselines fully protect user location privacy. While DeepTTE-FedAvg and CATTUE achieve semi-protection, user data within a geographical region is still collected by the edge servers. Further, Geo-I preserves exact location privacy by adding noise to the trajectory data, which may be nullified by the TTE server if map-matching is applied. Therefore, all compared baselines have an unfair TTE accuracy advantage over the proposed CRATE due to their relaxed privacy preservation assumptions.

5.2 Travel Time Estimation Accuracy

We commence our investigation by examining the deviation between predicted travel times and ground truth values, a pivotal aspect when evaluating the performance of Travel Time Estimation (TTE) methods. This case study deploys the proposed CRATE method alongside the aforementioned baseline algorithms on the three real-world datasets, evaluating their predictions using the four key performance metrics. In Table 2, we present the result comparison, with the first row after the header indicating whether each corresponding algorithm is privacy-preserving (✓), semi-privacy-preserving (✘), or privacy-leaking (✗).

From our comparative analysis, it becomes evident that the proposed CRATE method notably outperforms all semi-privacy-preserving algorithms, including WDR-F, DeepTTE-F, CATTUE, and Geo-I across all four performance metrics, all while improving privacy safeguards. In comparison to the leading algorithm in this category, Geo-I, CRATE achieves a substantial reduction in Mean Absolute Percentage Error (MAPE), ranging from 5.4% (Xi'an dataset) to 12.2% (Porto dataset). Despite both CRATE and the baselines utilizing historical trajectory data for route estimation, we assert that CRATE's unique contrastive route design significantly contributes to this improved performance. Furthermore, it is important to note that CRATE achieves full privacy preservation, surpassing existing federated learning-based TTE solutions. The results also demonstrate the remarkable generalization capabilities of CRATE across datasets from different sources with varying characteristics, thereby advocating its practical applicability.

Furthermore, it's noteworthy that CRATE consistently produces top-tier estimations, even when compared with the privacy-leaking group of baseline algorithms, where DeepTTE serves as the state of the arts. Notably, CRATE demonstrates superior estimation performance on the Porto dataset compared to all other baseline algorithms. This achievement may be attributed to the unique characteristics of the Porto dataset, which boasts a smaller total number of trajectories and a broader time span than the other datasets. The outcome highlights CRATE's potential applicability in relatively sparsely populated regions.

Meanwhile, it's crucial to emphasize that CRATE's strengths are not limited to less densely populated areas. CRATE maintains the second-best TTE accuracy on the Xi'an and Chengdu datasets. Moreover, it achieves full privacy preservation. As such, the proposed algorithm can make a significant contribution to the development of privacy-preserving ITS in regions of all types.

To provide a more vivid depiction of the predicted Travel Time Estimations (TTE) and ground truths, we present trimmed scatter plots for CRATE and the best-performing baseline algorithms: WDR(-F), DeepTTE(-F), and CATTUE, as shown in Fig. 4. Each scatter plot includes 2000 randomly selected routes from the Xi'an dataset, with their actual and predicted travel times displayed. The horizontal axes represent the ground truth travel times in seconds, while the vertical axes illustrate the TTE values generated by the respective algorithms. To enhance clarity, we also depict the 5th and 95th percentiles of the prediction errors with two red lines, including annotations indicating their slopes at the endpoints. These scatter plots strongly support our earlier conclusion that CRATE offers exceptional TTE performance

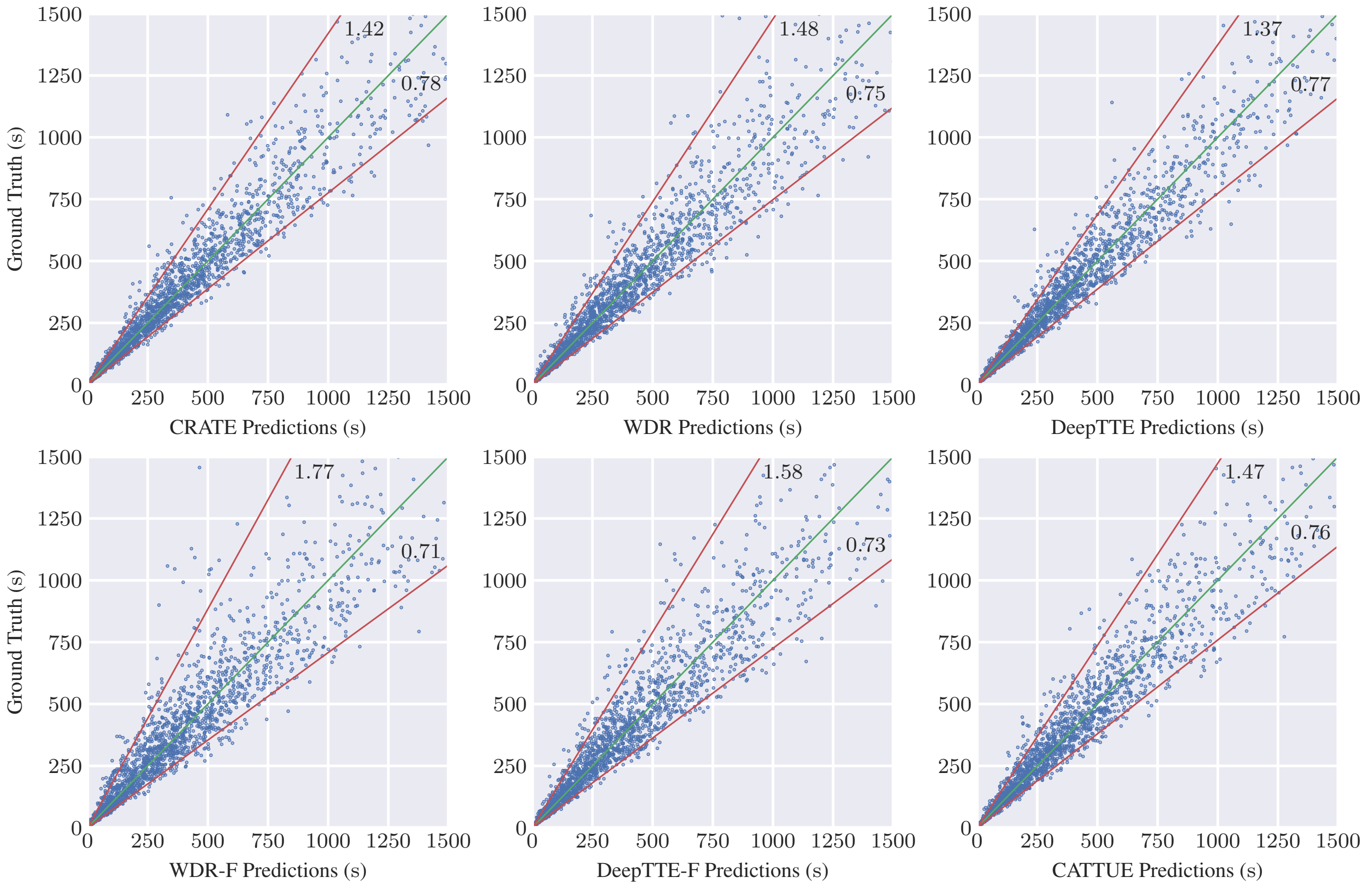

Fig. 4. Scatter plots of travel time ground truths and estimates by CRATE and best-performing baselines on the Xi'an dataset. This figure is best viewed printed or with Acrobat Reader.

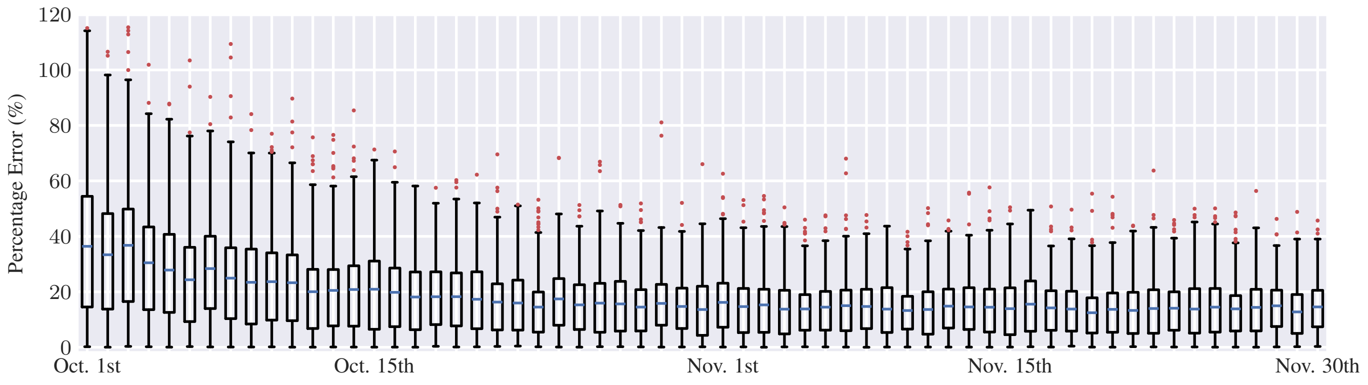

Fig. 5. Box plot of percentage errors of TTE values by CRATE over time on the Xi'an dataset.

while uniquely providing full privacy preservation capabilities.

Within CRATE, the travel time calibration model serves as a crucial component where user inputs play a pivotal role in enhancing model training while safeguarding users' geo-privacy. As previously discussed, we anticipated observing a gradual improvement in TTE accuracy. To demonstrate this process, we provide a box plot of TTE values by CRATE, using the Xi'an dataset as the basis for this analysis. The Xi'an dataset was chosen due to its best balance between the number of days and the number of routes per day. In Fig. 5, each box represents the percentage error of 200 randomly selected routes within a day. The plot includes the first and third quartiles (depicted as the box), the mean value (represented by the blue bar), 1.5 times the inter-quartile range whiskers, and any outliers (indicated by red dots). The figure clearly illustrates how the calibration model benefits from reported TTE errors, leading to a progressive enhancement of CRATE's performance over time.

5.3 Contrastive Route Generation on Mobile Devices

Diverging from conventional TTE algorithms in the literature, CRATE strategically offloads a portion of route computation from the server to edge clients, ensuring compre-

TABLE 3
`route2vec` Inference Time (ms) on Mobile Devices

Snapdragon SoC	888	855	430
Available Since	Q1 2021	Q1 2019	Q2 2016
PyTorch Mobile	79.90	95.37	536.08
TFLite	28.92	34.55	202.24

hensive privacy preservation. However, concerns may arise regarding potential delays in computation due to the limited resources available on mobile devices.

According to Fig. 1, primary computations on the edge are concentrated in the `route2vec` embedding process during contrastive route generation. To demonstrate the effectiveness of CRATE on the edge, we conducted a series of benchmark tests on mobile devices. `route2vec` initially developed and trained centrally with PyTorch in an offline manner, is deployed to mobile devices leveraging PyTorch Mobile and TFLite libraries on Snapdragon 430, 855, and 888 SoCs. During testing, mobile devices optimize performance by clearing the system environment, terminating other applications, and utilizing four big SoC cores for model inference. Both PyTorch Mobile and TFLite libraries are preloaded into memory for warming up.

The summarized results of generating 1000 contrastive routes for each library on each SoC are presented in Table 3. The `route2vec` inference times on modern SoCs within the past five years indicate that the proposed CRATE is highly efficient on the edge, achieving averaged route embedding times of around 30 ms with TFLite. Although multiple `route2vec` inferences are necessary for generating a contrastive route due to the auto-regressive nature of the generation algorithm (as discussed in Section 4.3), the total computation time remains well below one second for medium-length routes with around 30 segments.

Further, CRATE exhibits the potential to be applied to earlier mobile platforms with inferior computational capabilities. On the Snapdragon 430 SoC released in 2016, each route embedding consumes an average of 202 ms. The benchmarking indicates that CRATE can maintain high performance across both modern and older mobile platforms, making it a versatile solution for various deployment scenarios. Furthermore, CRATE's ability to operate efficiently on edge devices ensures quick and accurate TTE, which is crucial for real-time applications. By leveraging the processing power of mobile devices, CRATE not only preserves privacy but also enhances the responsiveness and scalability of the system. This dual benefit of privacy and efficiency sets CRATE apart from other TTE frameworks that either compromise on privacy or require significant mobile-side resources. Considering the prospect of more efficient and optimized route generation algorithms in the future, coupled with the continuous strengthening of mobile computational power, CRATE can potentially cater to the full spectrum of consumer-level mobile devices.

5.4 Hyper-parameter Sensitivity Analysis

CRATE incorporates several hyperparameters to define its contrastive route aggregation scheme and the architecture of the embedding models. These include the number of contrastive routes N^{CR}, the dimensionality of road segment and route embeddings D, and the number of stacked encoders in `route2vec`, N^{RE}. Intuitively, increasing N^{CR} provides more information to the aggregation scheme, potentially improving estimation at the cost of heavier computation. Similarly, increasing D and N^{RE} expands the embedding models' capacity but may increase the difficulty of model training.

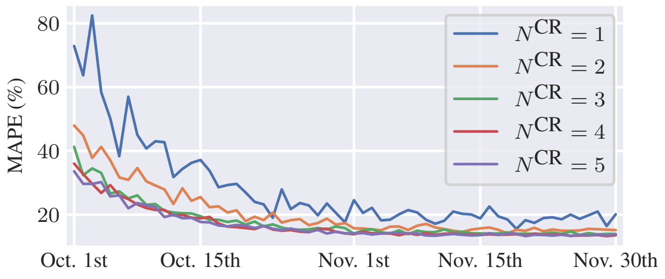

Fig. 6. Line plot of MAPE comparison with changing N^{CR} values.

d

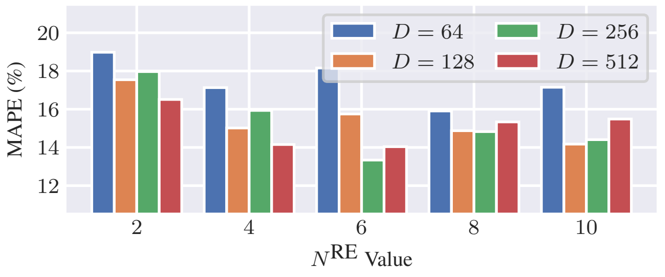

Fig. 7. Bar plots of MAPE comparisons with changing D and N^{RE} values.

In this sub-section, a series of hyperparameter sensitivity analyses are conducted to validate these hypotheses and offer an initial guideline on parameter selection. We begin by testing CRATE performance over time with $N^{\text{CR}} \in [1, 5]$ on the Xi'an dataset. The averaged MAPE results are depicted in Fig. 6. The plots reveal that a moderate number of contrastive routes is essential for achieving satisfactory TTE performance. Too few routes ($N^{\text{CR}} \leq 2$) incorporate insufficient travel time context for the edge device to infer the real time. Meanwhile, further increasing the number beyond $N^{\text{CR}} = 3$ results in diminishing returns on estimation accuracy, accompanied by a linearly increasing local computation time (as discussed in Section 5.3), making it a less favorable choice.

We then trained numerous `route2vec` models with $D \times R^{\text{RE}} \in \{64, 128, 256, 512\} \times \{2, 4, 6, 8, 10\}$, and the TTE accuracy and model training time are plotted in Figs. 7 and 8, respectively. The results confirm the expected increase in model training time with rising trainable parameters. However, the MAPE values suggest that stacked encoders reach saturating performance at $R^{\text{RE}} = 6$. We hypothesize that introducing additional encoders improves model expressiveness but also increases model training difficulty, leading to slightly worse MAPE values with $R^{\text{RE}} > 6$. Similar observations are made for D, and we determine that $D = 256$ and $R^{\text{RE}} = 6$ are a satisfactory pair of

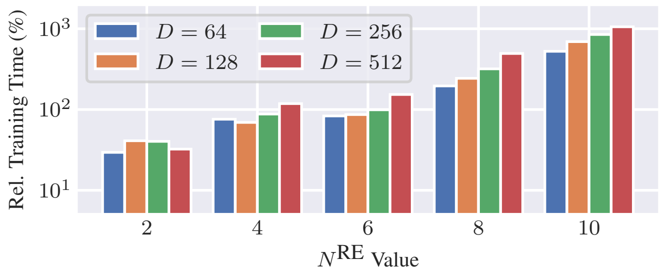

Fig. 8. Bar plots of relative training time comparisons with changing D and N^{RE} values.

hyperparameter choices.

5.5 Generality of Travel Time Calibration

From the analysis in Section 4.6, later confirmed by Fig. 5, it is clear that the travel time calibration model plays an important role in refining the aggregated travel time. In practical deployment, however, there might be cases where no user reports on their real travel times are readily available for calibration model training, rendering CRATE less performant on its initial deployment to new regions. Nevertheless, we argue that the calibration model, grounded on the high-dimensional route embeddings developed by `route2vec`, can be intrinsically transferred between cities to bootstrap initial CRATE deployment.

To verify this assertion, we perform a transfer-ability test of the calibration model in this sub-section. In particular, we adapt the calibration model trained on the Xi'an dataset to the Chengdu and Porto datasets on their first day of deployment and observe their daily MAPE changes while fine-tuning the pre-trained calibration model. The results are summarized by box plots in Figs. 9 and 10, where the green line plots are the averaged percentage error of TTE without pre-trained calibration model from Xi'an. From the simulation results, it can be observed that employing the transferred calibration model helps both datasets to achieve an approximately 20% TTE error on the first day (Chengdu) and week (Porto). We credit the outstanding transfer-ability of the calibration model to the highly generalized road segment embeddings by `seg2vec` and the built-upon `route2vec`, which offers cross-city route similarity in the embedding space. The potential of CRATE's calibration model is not limited to transfer-ability, and we will further explore its all-round contribution to CRATE and the general TTE problem in the future.

6 Conclusions

In this paper, we introduce Contrastive Route-Advised Travel Time Estimation (CRATE) as a comprehensive solution for privacy-preserving travel time estimation in modern transportation networks. The primary objective of this research is to provide users with accurate and timely travel time estimations while ensuring the protection of their geolocation privacy.

The fundamental concept behind CRATE is based on the idea that for any given route in a transportation network, there exist numerous randomly generated routes that would result in the same travel time. CRATE identifies these random routes, known as contrastive routes, through the utilization of road segment and route embeddings learned via self-supervised learning and an iterative contrastive route generation heuristic. Furthermore, CRATE employs a lightweight deep learning model for aggregating and calibrating the travel times of contrastive routes and develop the final travel time. This comprehensive approach ensures that CRATE not only achieves outstanding estimation accuracy but also maintains full privacy preservation capabilities.

By conducting a series of case studies on real-world trajectory datasets, we demonstrate the efficacy of CRATE in outperforming non- and semi-privacy-preserving baselines and achieving full privacy preservation. The results highlight CRATE's generalization capabilities across diverse datasets and its applicability in sparsely populated regions. Additionally, we deploy the proposed CRATE on various mobile devices to evaluate its model usability, and the results indicate its high efficiency.

In the future, we will further explore more advanced and learning-based contrastive route generation algorithms to improve the overall model accuracy. Investigating algorithms to incorporate ancillary information (external factors such as meteorology and incident) into the algorithm is another promising extension of CRATE.

References

[1] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *Proc. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18, London, United Kingdom, 2018, pp. 858–866.

[2] X. Fang, J. Huang, F. Wang, L. Zeng, H. Liang, and H. Wang, "ConSTGAT: contextual spatial-temporal graph attention network for travel time estimation at baidu maps," in *Proc. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug. 2020, pp. 2697–2705.

[3] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? estimating travel time based on deep neural networks," in *Proc. AAAI Conference on Artificial Intelligence*, ser. AAAI '18, vol. 32, no. 1, New Orleans, LA, US, Apr. 2018.

[4] J. J. Q. Yu, "Citywide estimation of travel time distributions with bayesian deep graph learning," *IEEE Trans. Knowl. Data Eng.*, pp. 2366–2378, Mar. 2023.

[5] U. Mori, A. Mendiburu, M. Álvarez, Lozano, and A. Jose, "A review of travel time estimation and forecasting for advanced traveller information systems," *Transportmetrica A: Transport Science*, vol. 11, no. 2, pp. 119–157, 2015.

[6] N. Zhou, W. X. Zhao, X. Zhang, J.-R. Wen, and S. Wang, "A general multi-context embedding model for mining human trajectory data," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 1945–1958, Aug. 2016.

[7] X. Zhan, Y. Zheng, X. Yi, and S. V. Ukkusuri, "Citywide traffic volume estimation using trajectory data," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 272–285, Feb. 2017.

[8] J. J. Q. Yu, "Sybil attack identification for crowdsourced navigation: a self-supervised deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4622–4634, Jul. 2021.

[9] H. Wang, Z. Zhang, Z. Fan, J. Chen, L. Zhang, R. Shibasaki, and X. Song, "Multi-task weakly supervised learning for origin–destination travel time estimation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 11, pp. 11 628–11 641, Nov. 2023.

[10] Z. Zheng, Y. Ye, Y. Zhu, S. Zhang, and J. J. Q. Yu, "Data-driven methods for travel time estimation: a survey," in *Proc. IEEE International Conference on Intelligent Transportation Systems*, Bilbao, Spain, Sep. 2023.

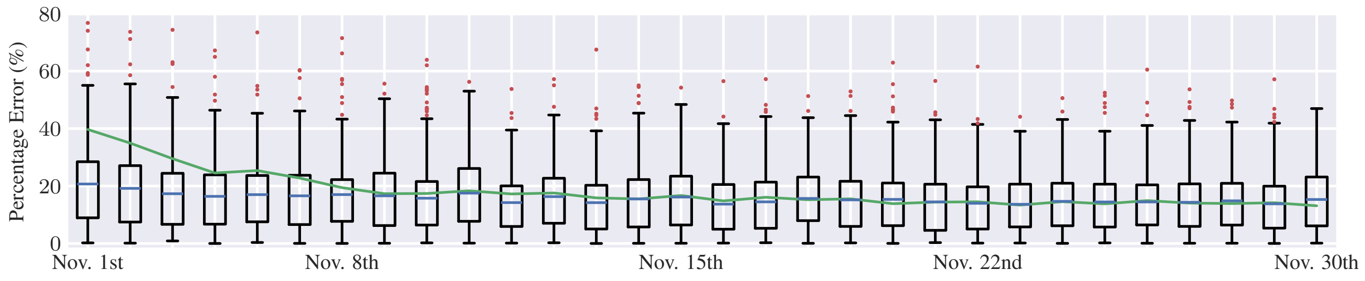

Fig. 9. Box plot of percentage errors of TTE values by CRATE with Xi'an calibration model over time on the Chengdu dataset.

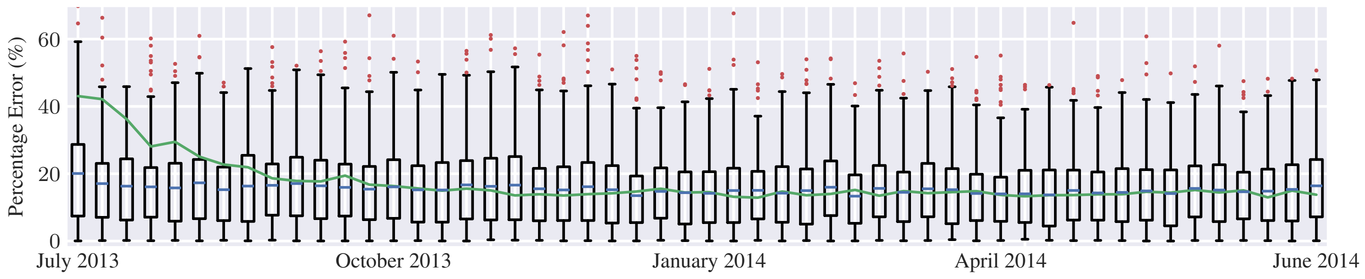

Fig. 10. Box plot of percentage errors of TTE values by CRATE with Xi'an calibration model over time on the Porto dataset. Each box stands for a week.

[11] Y. Zhu, Y. Ye, Y. Liu, and J. J. Q. Yu, "Cross-area travel time uncertainty estimation from trajectory data: a federated learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 24 966–24 978, Dec. 2022.

[12] Z. Fan, Z. Zhang, and H. Wang, "Generative personalized federated learning framework for travel time estimation," in *Proc. ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '22, Boston, MA, US, 2022, pp. 825–826.

[13] F. Xu, Z. Tu, Y. Li, P. Zhang, X. Fu, and D. Jin, "Trajectory recovery from ash: user privacy is not preserved in aggregated mobility data," in *Proc. International Conference on World Wide Web*, Perth, Australia, Apr. 2017, pp. 1241–1250.

[14] F. Jin, W. Hua, M. Francia, P. Chao, M. E. Orlowska, and X. Zhou, "A survey and experimental study on privacy-preserving trajectory data publishing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 5577–5596, Jun. 2023.

[15] Z. Tu, F. Xu, Y. Li, P. Zhang, and D. Jin, "A new privacy breach: user trajectory recovery from aggregated mobility data," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1446–1459, Jun. 2018.

[16] L. Yao, Z. Chen, H. Hu, G. Wu, and B. Wu, "Privacy preservation for trajectory publication based on differential privacy," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 3, Apr. 2022.

[17] K. Sui, Y. Zhao, D. Liu, M. Ma, L. Xu, L. Zimu, and D. Pei, "Your trajectory privacy can be breached even if you walk in groups," in *Proc. IEEE/ACM International Symposium on Quality of Service*, Beijing, China, Jun. 2016.

[18] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proc. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, New York, US, Aug. 2014.

[19] 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, Dec. 2018.

[20] L. Huang, Y. Yang, H. Chen, Y. Zhang, Z. Wang, and L. He, "Context-aware road travel time estimation by coupled tensor decomposition based on trajectory data," *Knowledge-Based Systems*, vol. 245, p. 108596, Jun. 2022.

[21] 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.

[22] T.-y. Fu and W.-C. Lee, "DeepIST: deep image-based spatio-temporal network for travel time estimation," in *Proc. ACM International Conference on Information and Knowledge Management*, Beijing, China, Nov. 2019.

[23] W. Lan, Y. Xu, and B. Zhao, "Travel time estimation without road networks: an urban morphological layout representation approach," in *Proc. International Joint Conference on Artificial Intelligence*, Macao, China, Aug. 2019.

[24] H. Zhang, H. Wu, W. Sun, and B. Zheng, "DeepTravel: a neural network based travel time estimation model with auxiliary supervision," in *Proc. International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, Jul. 2018.

[25] Z. Zhang, H. Wang, Z. Fan, J. Chen, X. Song, and R. Shibasaki, "GOF-TTE: generative online federated learning framework for travel time estimation," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 107—-24 121, Dec. 2022.

[26] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surv.*, vol. 51, no. 4, Jul. 2018.

[27] C. Dwork, "Differential privacy," in *Proc. International Colloquium on Automata, Languages, and Programming*, 2006, pp. 1–12.

[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 of Things Journal*, vol. 7, no. 8, pp. 7751–7763, 2020.

[29] X. Yuan, J. Chen, J. Yang, N. Zhang, T. Yang, T. Han, and A. Taherkordi, "FedSTN: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 8, pp. 8738–8748, 2023.

[30] M. Xia, D. Jin, and J. Chen, "Short-term traffic flow prediction based on graph convolutional networks and federated learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 1191–1203, 2023.

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "distributed representations of words and phrases and their compositionality," in *Proc. Advances in Neural Information Processing Systems*, Lake Tahoe, NV, US, Dec. 2013.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems*, Long Beach, CA, US, Dec. 2017.

[33] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021.

[34] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. International Conference on Machine Learning*, Haifa, Israel, Jun. 2010, pp. 807–814.

[35] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple

framework for contrastive learning of visual representations," in *Proc. International Conference on Machine Learning*, Jul. 2020.

[36] H. Steck, C. Ekanadham, and N. Kallus, "Is cosine-similarity of embeddings really about similarity?" in *Proc. ACM Web Conference*, May 2024, pp. 887–890.

[37] H. Wang, Y.-H. Kuo, D. Kifer, and Z. Li, "A simple baseline for travel time estimation using large-scale trip data," in *Proc. ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, San Francisco, CA, US, Oct. 2016.

[38] T. Chen and C. Guestrin, "XGBoost: a scalable tree boosting system," in *Proc. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, San Francisco, CA, US, Aug. 2016.

[39] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: differential privacy for location-based systems," in *Proc. ACM SIGSAC Conf. Computer & Communications Security*, 2013, pp. 901–914.

James Jianqiao Yu (S'11–M'15–SM'20) is a Professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen. He received the B.Eng. and Ph.D. degree in Electrical and Electronic Engineering from the University of Hong Kong, Pokfulam, Hong Kong, in 2011 and 2015, respectively. He was a post-doctoral fellow at the University of Hong Kong from 2015 to 2018. He held academic positions at Southern University of Science and Technology, China and University of York, United Kingdom from 2018 to 2024. His general research interests are in data mining, large and foundation models, and intelligent transportation. His work is now mainly on spatial-temporal data mining, multi-modal foundation model, and forecasting and logistics of future transportation. He has published over 100 academic papers in top international journals and conferences, and representative papers have been selected as ESI highly cited papers. He was the World's Top 2% Scientists since 2020 and of career by Stanford University. He is an Editor of IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS and IET SMART CITIES. He is a Senior Member of IEEE.