DPDeno: A Post-Processing Framework for Releasing Differentially Private Spatio-Temporal Mobility Features

Xinyue Sun, *Member, IEEE*, Xiaoyu Liu, Qingqing Ye, *Member, IEEE*, Haibo Hu, *Senior Member, IEEE*, Renyu Yang, *Member, IEEE*, Hui He, *Member, IEEE*, and Weizhe Zhang, *Senior Member, IEEE*

***Abstract*—The spatio-temporal (ST) mobility patterns derived from trajectory data are crucial for applications such as location-based services and urban analytics. However, releasing these mobility features raises significant privacy concerns, as they may expose sensitive personal location information. Differential privacy (DP) is widely used to safeguard individual privacy during data releases, but existing methods for releasing ST features often suffer from utility loss because their high dimensionality requires injecting substantial noise to meet privacy guarantees. Several recent approaches attempt to address this issue by reducing noise in differentially private spatio-temporal (DPST) features, but they either discard valuable information while compressing noisy data representations or rely solely on restrictive road network topology constraints, resulting in only modest utility improvements.**

In this paper, we present DPDeno, a post-processing framework designed to significantly enhance the utility of DPST features. First, DPDeno generates synthetic trajectory datasets using public information (e.g., road network data) and applies existing DP methods to create paired DPST (noisy) and ST (clean) features. It then trains a spatio-temporal graph autoencoder (STGAE), which models each feature as a graph, with road segments as nodes and transitions over time as edges. By minimizing node- and edge-level reconstruction losses between the noisy and clean pairs, STGAE learns to refine DPST inputs toward the structural consistency of their clean counterparts, thereby improving their practical utility. The trained model is then used to post-process real DPST features. Importantly, DPDeno preserves the original DP guarantee, as STGAE is trained solely on synthetic data generated from public sources without accessing any private information. Experimental results on two real-world trajectory datasets show that DPDeno significantly improves both the statistical accuracy and practical usability of released mobility features.

***Index Terms*—Differential privacy, trajectory features releasing, data privacy, trajectory data**

Xinyue Sun is with the School of Cyberspace Science, Harbin Institute of Technology, Harbin, 150001, Heilongjiang, China (e-mail: xysun@hit.edu.cn).

Xiaoyu Liu is with the School of Software, Beihang University, Beijing, 100191, China (e-mail:xyliumim@buaa.edu.cn).

Qingqing Ye is the Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail: qqing.ye@polyu.edu.hk).

Haibo Hu is with the Department of Electrical and Electronic Engineering, Hong Kong Polytechnic University, Hong Kong (e-mail: haibo.hu@polyu.edu.hk).

Renyu Yang is with the School of Software, Beihang University, Beijing, 100191, China (e-mail:renyuyang@buaa.edu.cn).

Hui He is with the School of Cyberspace Science, Harbin Institute of Technology, Harbin, 150001, Heilongjiang, China (e-mail: hehui@hit.edu.cn).

Weizhe Zhang is with the School of Cyberspace Science, Harbin Institute of Technology, Harbin & Shenzhen, 150001 & 518055, Heilongjiang & Guangdong, China (e-mail: wzzhang@hit.edu.cn).

Corresponding author: Weizhe Zhang.

I. Introduction

In recent years, spatio-temporal (ST) trajectory data has become increasingly valuable in various fields, including urban planning, traffic management, and location-based services [1]–[3]. The mobility features derived from this data provide critical insights into human movement patterns and behaviors, enabling more informed decision-making and personalized services. However, sharing such sensitive features has raised significant privacy concerns [4]. The detailed nature of ST mobility features can unintentionally expose individuals' locations, movements, and habits, potentially leading to privacy breaches. Thus, developing methods to release ST mobility features while preserving user privacy has become a critical priority.

A promising approach is differential privacy (DP), which has emerged as the gold standard for protecting individual privacy in data analysis and release [5]–[7]. It provides a robust framework that safeguards personal data while still allowing for the extraction of valuable insights from aggregated datasets. The core principle of DP involves adding calibrated noise to data outputs, ensuring that the presence or absence of any single individual's data does not significantly alter the overall results. Due to its reliability, DP has been widely adopted by companies and government agencies [8], [9].

Challenges. In the existing literature, DP has mainly been applied to the release of two-dimensional spatial mobility features [10]–[19]. Current DP methods generally fall into three categories: Direct Noise Injection (DNI) [5], [10], Sample-based Noise Injection (SNI) [9], [11], [12], [19], and Normalized Count-based Noise Injection (NCNI) [13]–[15], [17], [18]. DNI directly injects noise into the ground-truth spatial mobility features to meet ϵ-DP. However, this introduces excessive noise, as it must account for the worst-case scenario in any trajectory. To reduce the amount of noise needed to satisfy ϵ-DP, SNI samples a subset of data points from each trajectory and injects noise only into the features derived from these samples, while NCNI limits the influence of each trajectory on the mobility features by constraining their impact to a fixed value, applying noise only to the adjusted features. Despite substantial progress, extending these techniques to three-dimensional spatio-temporal (ST) mobility features remains challenging. Adding the temporal dimension reveals evolving spatial patterns but also significantly raises the global sensitivity of the release function. In this case, to

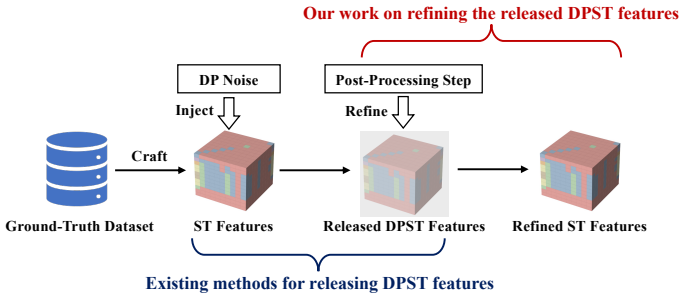


Fig. 1: From DPST feature release to post-processing refinement.

satisfy a given ϵ-DP guarantee, more noise must be added—often overwhelming the resulting differentially private spatio-temporal (DPST) features and severely degrading their utility.

Motivations. As illustrated in Figure 1, inspired by the post-processing invariance property of differential privacy, *i.e., any post-processing applied to the output of a DP algorithm does not consume additional privacy budget*, this work shifts the focus from directly releasing DPST features to refining the DPST features released by existing methods through post-processing. Our refinement aims to achieve two objectives: (1) enhancing the utility of the released DPST features in downstream tasks, such as mobility trend prediction and hotspot recommendation, and (2) strictly adhering to DP's post-processing invariance to ensure the original privacy guarantee remains intact. Existing studies [19] attempt to compress noisy histograms into a low-dimensional latent space to reduce noise, but this process inevitably discards some genuine signals. Other methods [18] utilize road-segment connectivity to filter out implausible transitions. However, their single-constraint design yields only limited improvements in utility.

Our Contributions. To achieve our objectives, we propose a post-processing framework named DPDeno, which aims to enhance the utility of released DPST features while preserving the original privacy guarantees. DPDeno is based on the insight that ST mobility features are shaped by publicly available real-world constraints, such as road network topology, segment lengths, and speed limits. These structural patterns reflect common travel behavior and can be leveraged to refine DPST features released by DP mechanisms (e.g., NCNI, DNI, and SNI). Building on this, DPDeno incorporates a spatio-temporal graph autoencoder (STGAE) that represents each ST feature as a graph where nodes correspond to spatial units and edges represent temporal transitions. STGAE enhances the utility of released DPST features by refining them toward real-world structural patterns.

However, training STGAE under DP poses a key challenge: a learned model may leak information about the data it was trained on [20], [21]. Therefore, the ground-truth data cannot be used for training. To overcome this, DPDeno avoids using ground-truth data altogether and instead generates synthetic trajectories based on publicly available real-world knowledge. Specifically, it first collects auxiliary information related to the target dataset (e.g., road topology and traffic rules), and then synthesizes realistic trajectories under these constraints. ST features are computed from the synthetic data and perturbed using the DP methods to create paired training samples, i.e., clean ST features and their noisy DPST counterparts. DPDeno then trains STGAE to refine the DPST features toward the structural consistency of their clean versions. Once trained, STGAE is applied to refine DPST mobility matrices generated from the ground-truth data. Finally, we formally show that the refinement performed by DPDeno satisfies the post-processing invariance of DP, thereby preserving the original privacy guarantee without consuming any additional privacy budget. In addtion, since DPDeno focuses on the post-processing of DP method outputs, it is compatible with both current and future DPST feature releasing methods without requiring modifications to these methods.

We evaluate DPDeno on two real-world trajectory datasets using the three DP methods. The results show that DPDeno effectively enhance the utility of the DPST mobility matrices, preserving statistical properties approximating those of the ground-truth data and performing well in various practical applications.

In summary, the key contributions of our work are as follows:

- We propose DPDeno, the first post-processing framework specifically designed to enhance the utility of DPST mobility features while maintaining privacy guarantees. It improves feature quality by refining the released data to align with real-world structural patterns, derived from publicly available geographic and traffic information.
- We introduce a STGAE in DPDeno to refine the utility of DPST features. This autoencoder effectively captures the temporal dynamics and spatial dependencies of road segment transitions, enabling more accurate representation and enhancing the utility of DPST features.
- We design a novel training strategy for the STGAE that enables utility improvement while preserving privacy. STGAE is trained entirely on synthetic data generated from public knowledge and never accesses the ground-truth dataset, ensuring strict compliance with DP.
- We conduct comprehensive experiments on two real-world trajectory datasets. Results show that DPDeno consistently improves statistical accuracy and downstream task performance compared to both existing DP methods and post-processing methods.

The remainder of this paper is organized as follows. Section II reviews related work, and Section III introduces the necessary preliminaries. Section IV defines the problem and presents three straw-man DP methods for releasing DPST mobility matrices. Section V details the proposed DPDeno framework and formally shows that it operates as a post-processing step of the underlying DP mechanisms. Section VI reports the experimental results, and Section VII concludes the paper and outlines future research directions.

II. Related Work

A. Differential Privacy

Differential Privacy (DP) has become the gold standard in privacy protection due to its strong, provable privacy guarantees. DP algorithms enable the release of privacy-preserving

features or datasets, allowing data consumers to work with these resources without accessing users' true data. Early research in DP focused on creating specialized algorithms tailored to various data types, including medical data [10], location data [22]–[24], social networks [25], [26], time-series data [27]–[29], and learning problems [19], [30]–[32]. Recently, some works have tried to release the mobility features of trajectories or the trajectories themselves under DP.

B. Releasing Differentially Private Mobility Features

Over the past decade, privacy concerns surrounding trajectory data have been extensively studied, leading to various proposed solutions [5], [10]–[18], [33], [34]. Most of these methods focus on extracting differentially private two-dimensional spatial mobility features from original datasets and either releasing them directly or using them to reconstruct trajectories, all while maintaining DP privacy guarantees. These methods can be categorized into three groups.

- The first group injects a significant amount of noise directly into the original ST mobility features to satisfy ϵ-DP [5], [10]. However, because this method must account for the worst-case impact of an individual's trajectory, it often results in excessive noise that reduces the utility of the released features.
- The second group [11], [12] involves sampling the dataset before applying ϵ-DP noise to the mobility features derived from the sampled data. By aggregating the sampled data, this method reduces the worst-case influence of a single user's trajectory, allowing for less noise while still satisfying ϵ-DP.
- The third group [13]–[15], [17], [18] releases mobility features by constraining the overall change each trajectory contributes to the mobility features to a fixed value. This method effectively reduces the sensitivity of the mobility features, thereby requiring less noise to satisfy ϵ-DP.

While these techniques have shown success in the spatial domain, extending them to ST mobility features introduces new challenges. Adding the temporal dimension reveals dynamic mobility patterns but also significantly increases the global sensitivity of the release function. As a result, satisfying ϵ-DP requires injecting substantially more noise, thereby degrading the utility of the released features.

Recent studies have proposed various techniques to address the challenges of balancing privacy protection with data utility. For instance, Cao et al. [35] combined geo-indistinguishability with perturbation privacy through Hilbert curves and a Permute-and-Flip mechanism, optimizing service quality while preserving privacy. Wu et al. [2] tackled correlations in multi-user trajectories by introducing a data augmentation method, which improved privacy protection and utility across multiple users. Yuan et al. [36] focused on preserving semantic correlations in trajectory data, using hidden Markov models to balance privacy with service quality. Despite these advancements, the added noise often distorts important patterns in the data. Thus, the resulting noisy data often reduces the utility of the generated DPST mobility features, making them less effective for downstream tasks.

C. Post-Processing Techniques

A natural strategy for improving the utility of DPST mobility features is to reduce the added noise. Deep learning-based denoising techniques [37]–[40] have demonstrated strong performance in recovering clean signals from noisy inputs, particularly in domains like image processing. However, these methods require access to ground-truth data during training, which poses privacy risks in the DP setting. Models trained on original datasets may inadvertently leak sensitive information [20], [21], making such approaches incompatible with strict privacy guarantees.

To mitigate noise while preserving privacy guarantees, recent studies have leveraged the post-processing invariance of DP (Definition 4). Ahuja et al. [19] proposed the Variational Density Release (VDR) method, which denoises DP-generated histograms using a variational autoencoder. By encoding noisy histograms into a low-dimensional latent space and decoding them to produce denoised outputs, their approach avoids accessing private data and does not consume additional privacy budget. However, since the model is trained to reconstruct noisy inputs rather than ground-truth features, it struggles to recover accurate signals, limiting its practical utility. Sun et al. [18] proposed PUTS, which improves noisy trajectory data by removing implausible transitions using publicly available road-network topology. This rule-based approach adheres to DP guarantees and enhances data plausibility, but its utility gains are limited because it only incorporates basic topological constraints, ignoring richer structural information such as segment lengths, speed limits, and access restrictions.

To address these limitations, we propose DPDeno, a framework specifically designed for **the post-processing stage of released DPST mobility features**. DPDeno enhances the utility of these features without compromising their existing privacy guarantees by: (1) leveraging a spatio-temporal graph autoencoder combined with publicly available geographic structural information to reconstruct DPST features that more accurately reflect real mobility patterns, and (2) strictly adhering to the post-processing property of DP to ensure the original privacy guarantees remain intact.

III. Preliminaries

This section introduces the notation of the traffic data model and the basic terminology of DP. Table I summarizes these notations related to this work.

A. Traffic Data Model

Raw Trajectory. A raw trajectory is denoted as a sequence of GPS points, each associated with a timestamp, denoted as $\langle lat_i, lon_i, ts_i\rangle_{i=1}^{K} = (\langle lat_1, lng_1, ts_1\rangle, \langle lat_2, lng_2, ts_2\rangle, \cdots, \langle lat_K, lng_K, ts_K\rangle)$, where K is the length of the trajectory.

Example 1 (Raw Trajectory)**.** *Consider a trajectory* $(\langle 34.0401, -119.2013, 13578741\rangle, \cdots, \langle 34.0371, -119.1934, 13578884\rangle)$ *from the origin s to the destination d, as shown in Figure 2 (a). Each* $\langle lat_i, lon_i, ts_i\rangle$ *represents a GPS coordinate and the corresponding timestamp when the vehicle*

TABLE I: Summary of Notations

Symbol	Description
$\langle lat_i, lon_i, ts_i \rangle_{i=1}^{K}$	Raw trajectory of K GPS points with timestamps
$G = \langle I, R \rangle$	Road network with intersections I and segments R
r_i	The ith road segment in a trajectory
td_i, tc_i	Departure and travel time on r_i
$\tau = \langle r_i, td_i, tc_i \rangle$	Map-matched trajectory sequence
$T = \{t_1, \ldots, t_{\lvert T \rvert}\}$	Sequence of $\lvert T \rvert$ time intervals
$\mathcal{M}$	Spatio-temporal mobility matrix sequence over T
$\mathcal{M}(i, j, k)$	Matrix entry for $r_i \to r_j$ during t_k
$\tilde{\mathcal{M}}$, $\mathcal{M}'$	Noisy or normalized versions of $\mathcal{M}$ under DP
D, D_s	Original or synthetic trajectory dataset
ϵ, Δf	DP privacy budget and sensitivity of function f
$\mathcal{G} = (N, E, X, Y)$	ST graph with nodes N, edges E, features X, Y
$X \in \mathbb{R}^{\lvert T \rvert \times n \times d}$	Node features over time (e.g., speed, location)
$Y \in \mathbb{R}^{\lvert T \rvert \times m \times 1}$	Edge (transition) features over time
H_Y, $H(e, t)$	Latent edge embeddings from TLs at time t
n_{e_0}, n_{e_1}	Source and destination nodes of edge e
$\tilde{H}(e, t)$	Refined edge embedding at time t after MLP
MLP$(\cdot)$, concat$(\cdot)$	Multi-layer perceptron; feature concatenation
W_e	Spatial edge weight
$\mathcal{L}_{edge}$, $\mathcal{L}_{node}$, $\mathcal{L}$	Loss for edge, node, and total reconstruction
θ	Trainable model parameters

passes that point. The total travel time from s to d is the difference between the timestamps at these two points, which is 143 seconds in this example.

Road Networks. A road network is modeled as a directed graph $G = \langle I, R \rangle$, where I represents the set of intersections, and $R = \{r_1, r_2, ..., r_n\}$ denotes the set of road segments, with n being the total number of segments in R. Each segment $r \in R$ connects two intersections in I. For simplicity, we use undirected edges in Examples 1 and 2.

Map-Matched Trajectory. A raw trajectory, represented as a sequence of GPS points with timestamps, can be aligned with a road network to create a map-matched trajectory. Typically, a map-matching algorithm projects each GPS point onto the nearest road segment, transforming the raw data into a trajectory consistent with the road network structure [41], as illustrated in Example 2. The resulting map-matched trajectory is represented as $\tau = \langle r_i, td_i, tc_i \rangle_{i=1}^{|\tau|}$, where r_i denotes a road segment, td_i is the departure time at the start of r_i, tc_i is the travel time for r_i (i.e., $tc_i = td_{i+1} - td_i$), and $|\tau|$ is the length of the trajectory τ.

Example 2 (Map-Matched Trajectory)**.** *Consider a map-matched trajectory $\tau = (\langle r_1, td_1, tc_1 \rangle, \langle r_2, td_2, tc_2 \rangle, \langle r_3, td_3, tc_3 \rangle)$ in Figure 2 (b). Each road segment in τ is derived from the raw trajectory in Figure 2 (a) using a map-matching algorithm. The departure time td_i and travel time tc_i for each segment $i \in \{1, 2, 3\}$ are calculated using linear interpolation [30], [42] based on the GPS coordinates and timestamps from Figure 2 (a). The trajectory's origin and destination are aligned with the corresponding road segments, with offsets* offset_s *and* offset_d.

Spatio-Temporal Mobility Matrix. Building on the definition of spatial mobility features from previous studies [13]–[15], [17], [18], we define the spatio-temporal (ST) mobility matrix as follows. Given a trajectory dataset D and its associated road network $G = \langle I, R \rangle$, the ST mobility matrix $\mathcal{M}$ consists of a sequence of transition probability values for each consecutive

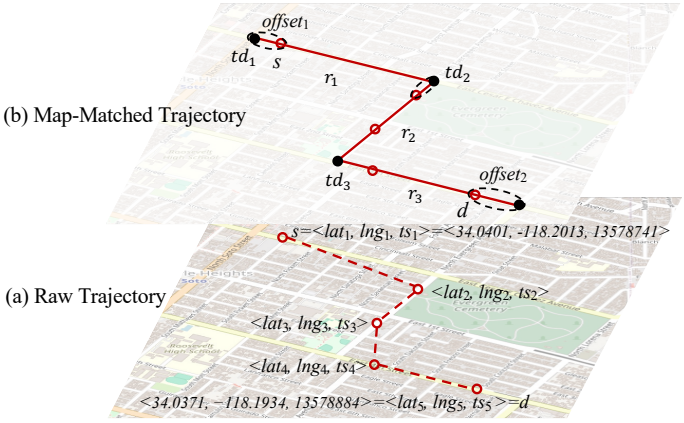


Fig. 2: Raw trajectories and map-matched trajectories.

time interval $T = t_1, t_2, \ldots, t_{|T|}$. For each time period t_k, the matrix entry $\mathcal{M}(i, j, k)$ represents the number of transitions from road segment r_i to road segment r_j during the time interval t_k.

This matrix effectively captures the transition dynamics between road segments in R across different time intervals, serving as a valuable tool for various trajectory-related applications, such as trajectory analysis, spatio-temporal road segment access histograms, and the generation of synthetic trajectory data.

B. Differential Privacy

Differential Privacy. Differential Privacy (DP) [5], [10] is a mathematical framework that quantifies the privacy protection provided by a randomized algorithm when applied to an input dataset. Essentially, DP ensures that an individual's presence or absence in the dataset does not significantly affect the output of that algorithm. The formal definition of DP is as follows.

Definition 1 (ϵ-DP)**.** *Given two neighboring datasets D and D' that differ by a single record, a randomized mechanism $\mathcal{A}$ satisfies ϵ-DP if*

$$Pr[\mathcal{A}(D) \in S] \leq e^{\epsilon} \cdot Pr[\mathcal{A}(D') \in S], \tag{1}$$

where ϵ is the privacy budget, with smaller values of ϵ providing stronger privacy guarantees.

Laplace Mechanism. One of the most widely used mechanisms to achieve ϵ-DP is the Laplace mechanism [10].

Definition 2 (Laplace Mechanism [5])**.** *Let $Lap\left(\frac{\Delta f}{\epsilon}\right)$ represent a random variable sampled from a Laplace distribution with mean 0 and scale parameter $\frac{\Delta f}{\epsilon}$. For any function f, the Laplace mechanism is defined as:*

$$\mathcal{A}(f, D) = f(D) + Lap\left(\frac{\Delta f}{\epsilon}\right), \tag{2}$$

where Δf is the sensitivity of the function $f(\cdot)$, defined as the maximum difference in f's output over any two neighboring datasets D and D'.

Definition 3 (Sensitivity [5]). *Let $f : D \to \mathbb{R}^m$ be a function that maps a dataset D to a vector of m real numbers. The sensitivity of f is defined as:*

$$\Delta f := \max_{D,D'} ||f(D) - f(D')||, \tag{3}$$

where $||\cdot||$ denotes the L_1 norm, and the maximum is taken over all pairs of neighboring datasets D and D'.

DP Properties. Additionally, we outline key composition properties of DP below.

Definition 4 (Composition Properties [10]). *Let $\mathcal{A}_1, ..., \mathcal{A}_k$ be k algorithms. For each $i \in \{1, 2, ..., k\}$, $\mathcal{A}_i$ satisfies ϵ_i-DP. The following properties apply:*

- ***Parallel Composition:*** *If these algorithms are applied to disjoint subsets of D, they satisfy ($\max_{i\in[1,...,k]} \epsilon_i$)-DP.*
- ***Post-Processing Invariance:*** *Any post-processing of the output of a differentially private algorithm does not consume additional privacy budget.*

IV. Problem Formulation and Straw-Man Approaches

A. Problem Formulation

System Model. We consider a common scenario in which a data curator collects a trajectory dataset $D = \{\tau_1, \ldots, \tau_N\}$, where each trajectory τ_i records a user's time-ordered transitions across a road network $G = (I, R)$. The curator aims to construct and release a ST mobility matrix based on D, facilitating applications such as frequent pattern mining and range count queries.

Threat Model. We assume the data recipient (or downstream user) is untrusted (i.e., a potential adversary), which has arbitrary background knowledge (e.g., geographic data) and can access to the released ST mobility features. In this case, the adversary can launch various privacy attacks based on the ST mobility features, including differential attacks, trajectory reconstruction, and linkage attacks. For example, in a differential attack, the adversary compares released outputs with and without a specific user's data to determine whether that user's trajectory is present in the original dataset. These differences can then be used to infer the user's movement patterns and visits to sensitive locations (e.g., home address, workplace, medical facilities).

Objective. Our objective is to release an ST mobility matrix that provides high utility (i.e., accurately preserving statistical properties of the original data and effectively supporting real-world analytical tasks) while ensuring rigorous privacy guarantees. Specifically, the released matrix must prevent an adversary from confidently inferring sensitive individual information. Ideally, the released ST mobility matrix should closely resemble the original data in terms of statistical accuracy and practical applicability.

B. Straw-Man Approaches

To satisfy both the privacy requirements and good utility, the closest solution is DP model. In what follows, we introduce

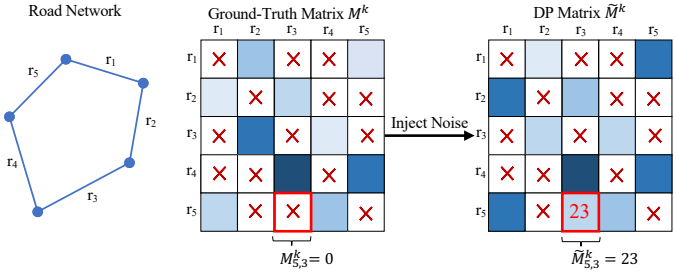


Fig. 3: Raw trajectories and map-matched trajectories.

three straw-man approaches for releasing the ST moblity features under DP, which are dapted and extended from the existing works [5], [10], [13]. We then establish their privacy guarantees and analyze their key technical challenges.

1) Straw-Man Approaches: The first method, Direct Noise Injection (DNI) [10], applies the Laplace mechanism directly to release the DPST mobility matrix $\tilde{\mathcal{M}}$ from the trajectory dataset D. Given a privacy budget ϵ, the data curator first constructs the ground-truth ST mobility matrix $\mathcal{M}$ by calculating the transition counts $\mathcal{M}(i, j, k)$ for each time period $t_k \in T$, as described in Section IV-A. The sensitivity of $\mathcal{M}$ is determined by the maximum number of segment transitions a user contributes, i.e., $|\tau_{\max}| - 1$. To ensure ϵ-differential privacy, Laplace noise $Lap((|\tau_{\max}| - 1)/\epsilon)$ is added to each entry of $\mathcal{M}$, resulting in the noisy matrix $\tilde{\mathcal{M}} = \mathcal{M} + Lap((|\tau_{\max}| - 1)/\epsilon)$. The released DPST mobility matrix $\tilde{\mathcal{M}}$ preserves user privacy while maintaining the statistical structure of the original transitions.

The second method, Sample-based Noise Injection (SNI) [10], reduces the noise needed to satisfy ϵ-DP by capping the number of segment transitions used to create the ST mobility matrix to a maximum of $\ell \leq |\tau_{\max}| - 1$ per trajectory. Specifically, the data curator selects a subset D_{sp} from D: for users with more than ℓ transitions, ℓ transitions are sampled uniformly at random, while all transitions are retained for users with ℓ or fewer transitions. The curator then constructs the ST mobility matrix $\mathcal{M}_{\text{sp}}$ using D_{sp}, thereby reducing its sensitivity to ℓ. Laplace noise $Lap(\ell/\epsilon)$ is then added to each entry of $\mathcal{M}_{\text{sp}}$, resulting in the noisy matrix $\tilde{\mathcal{M}}_{\text{sp}} = \mathcal{M}_{\text{sp}} + Lap(\ell/\epsilon)$. The DPST mobility matrix produced by SNI is denoted as $\tilde{\mathcal{M}} = \tilde{\mathcal{M}}_{\text{sp}}$. By reducing the noise in $\tilde{\mathcal{M}}_{\text{sp}}$, this method satisfies the privacy requirements of ϵ-DP while preserving accurate mobility features derived from the sampled data.

The third method, Normalized Count-based Noise Injection (NCNI) [13], normalizes the transition counts for each trajectory by limiting its overall impact on the mobility features to a fixed value of 1. This method effectively reduces the sensitivity of the mobility matrix, thus lowering the amount of noise required to meet ϵ-DP. Specifically, the data curator constructs a normalized ST mobility matrix $\mathcal{M}'$ for the dataset D, where each entry $\mathcal{M}'(i, j, k)$ represents the normalized transition count from road segment $r_i \in R$ to $r_j \in R$ during the time period $t_k \in T$. Formally, $\mathcal{M}'(i, j, k)$ is calculated as:

$$\mathcal{M}'(i, j, k) = \frac{\#\text{ transitions from } r_i \text{ to } r_j \text{ in trajectory } \tau \text{ during } t_k}{|\tau| - 1}, \tag{4}$$

where r_i and r_j are consecutive segments in the trajectory τ, and $|\tau| - 1$ is the total number of segment transitions in τ.

According to the definition of DP [5], the sensitivity of $\mathcal{M}'$ is determined by comparing neighboring trajectory datasets D and D', where $D = D' \cup \{\tau'\}$ or $D' = D \cup \{\tau'\}$, with τ' representing a single user's trajectory. Without loss of generality, assume $D = D' \cup \{\tau'\}$, and let $\mathcal{M}'(D)(i,j,k)$ represent the normalized count for D. The L_1 norm of the difference between the normalized matrices $\mathcal{M}'(D)$ and $\mathcal{M}'(D')$ is given by:

$$\begin{aligned} &||\mathcal{M}'(D) - \mathcal{M}'(D')|| \\ &= \sum_{k=1}^{|T|}\sum_{i=1}^{n}\sum_{j=1}^{n}(\mathcal{M}'(D)(i,j,k) - \mathcal{M}'(D')(i,j,k)) \\ &= \sum_{k=1}^{|T|}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathcal{M}'(\tau')(i,j,k) = 1. \end{aligned} \tag{5}$$

Thus, the sensitivity $\Delta\mathcal{M}'$ is:

$$\Delta\mathcal{M}' = \max_{D,D'} ||\mathcal{M}'(D) - \mathcal{M}'(D')|| = 1. \tag{6}$$

To ensure ϵ-DP, Laplace noise $Lap(1/\epsilon)$ is added to each entry of $\mathcal{M}'$, resulting in the noisy matrix $\tilde{\mathcal{M}}'$:

$$\tilde{\mathcal{M}}' = \mathcal{M}' + Lap(1/\epsilon). \tag{7}$$

The DPST mobility matrix generated by NCNI is denoted as $\tilde{\mathcal{M}} = \mathcal{M}'$. By normalizing the features, the sensitivity is effectively controlled, allowing for a reduction in the amount of noise required. This method ensures compliance with ϵ-DP while maintaining a high level of accuracy in the released mobility features.

2) Privacy Analysis: In this subsection, we demonstrate that constructing $\tilde{\mathcal{M}}$ using DNI, SNI, or NCNI satisfies ϵ-DP.

Theorem 1. *Constructing $\tilde{\mathcal{M}}$ using DNI, SNI, or NCNI satisfies ϵ-DP.*

Proof. As outlined in Section IV-B1, the sensitivity of $\mathcal{M}$ in DNI is $|\tau_{\max}-1|$, in SNI it is ℓ, and in NCNI it is 1. According to the Laplace mechanism, Laplace noise $Lap((|\tau_{\max}-1|)/\epsilon)$, $Lap(\ell/\epsilon)$, and $Lap(1/\epsilon)$ is added to $\mathcal{M}$, $\mathcal{M}_{\text{sp}}$, and $\mathcal{M}'$, respectively, resulting in the noisy matrices $\tilde{\mathcal{M}}$, $\tilde{\mathcal{M}}_{\text{sp}}$, and $\tilde{\mathcal{M}}'$. Collectively, we refer to any of these noisy matrices as $\tilde{\mathcal{M}}$, depending on the method applied. By the post-processing property of DP, generating $\tilde{\mathcal{M}}$ through any of these methods still satisfies ϵ-DP. □

3) Key Technical Challenges: Although the three strawman approaches (DNI, SNI, and NCNI) satisfy ϵ-DP, applying them to ST mobility features presents serious utility problems. Adding the time dimension increases the amount of detail captured in the data, but it also raises the sensitivity of the release function. As a result, more noise must be added to meet the same privacy guarantee, which often overwhelms the data and damages its usefulness. Figure 3 shows a typical example. Suppose the real trajectory data never includes a transition from road segment r_5 to r_3, meaning $\mathcal{M}^k_{5,3} = 0$ for all $t_k \in T$. After adding noise, however, the released DPST matrix might show a large value like $\tilde{\mathcal{M}}^k_{5,3} = 23$, even though such a transition is impossible. These errors can mislead downstream tasks and make the released data much less reliable.

To address these limitations, our proposed DPDeno framework, described next, effectively refines $\tilde{\mathcal{M}}$ toward real-world knowledge and constraints, thereby enhancing the utility of the released matrix.

V. DPDeno Framework

This section presents DPDeno, a post-processing framework that refines DPST mobility features while preserving DP guarantees. It consists of three components: knowledge-driven data generation, model development and training, and post-processing with the trained model. We also demonstrate that the refinement step satisfies the post-processing property of DP and incurs no additional privacy cost.

A. Overview

Motivation. DPDeno is based on the observation that ground-truth ST mobility features are often heavily influenced by publicly available real-world factors, such as the road network topology and traffic rules. These elements directly shape users' movement patterns over time. Thus, both the ground-truth ST mobility matrix $\mathcal{M}$ and its DP counterpart $\tilde{\mathcal{M}}$ are expected to follow structural patterns shaped by these real-world constraints. This insight allows us to extract structural patterns from public knowledge and leverage them to refine DPST features generated by existing DP methods (e.g., DNI, SNI, and NCNI), thereby enhancing their utility. To achieve this, we introduce a spatio-temporal graph autoencoder (STGAE), which is particularly well-suited for modeling the complex spatial and temporal dependencies that define real-world mobility patterns. By learning both the spatial relationships between road segments and their time-varying attributes, the autoencoder can guide DPST features toward realistic mobility patterns, improving the accuracy and utility of the DPST matrices.

Challenge and Solution. However, a key challenge arises under the DP setting: training on the ground-truth dataset may lead to privacy leakage, as repeated access could compromise the overall privacy guarantee [20], [21]. To address this, DPDeno avoids using ground-truth data altogether. Instead, it simulates realistic trajectories using public geographic and behavioral knowledge (e.g., road topology, segment lengths, speed limits) to construct synthetic ST features. These synthetic features are then perturbed using existing DP methods to create paired training samples: noisy DPST inputs and their clean counterparts. STGAE is trained on these pairs to refine noisy DPST features toward the structural consistency found in their clean counterparts. Once trained, the model is used to refine real DPST outputs generated from ground-truth data. As this process is entirely based on publicly available information, it functions as a post-processing step under the DP framework. Therefore, it neither compromises DP guarantees nor requires additional privacy budget.

Based on these principles, we develop DPDeno, as illustrated in Figure 4, which consists of three phases: ①

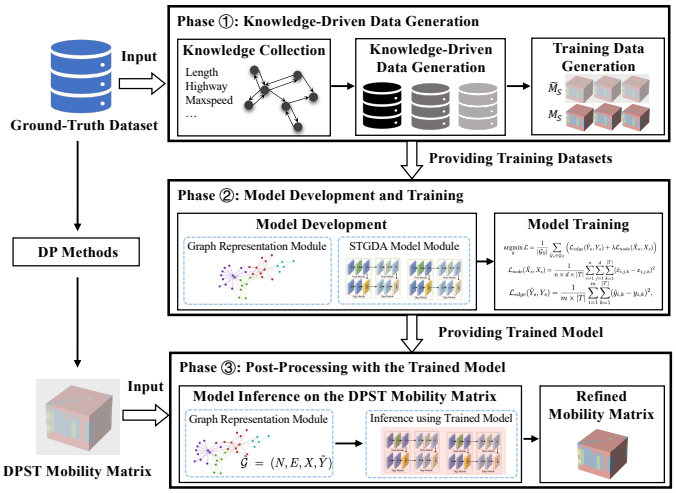


Fig. 4: An overview of DPDeno framework.

knowledge-driven data generation, ② model development and training, and ③ post-processing with the trained model.

Phase ①: Knowledge-Driven Data Generation. DPDeno begins by collecting the target city's road network G and its associated attributes D_A (e.g., speed limits) from public sources. Based on this information, it generates L synthetic trajectory sets $D_S = \{D_{s_i}\}_{i=1}^{L}$, each containing N trajectories that simulate daily travel behaviors. From D_S, we compute a set of ST mobility matrices $\mathcal{M}_S$ and apply a chosen DP method (e.g., DNI, SNI, or NCNI) to produce the corresponding DPST matrices $\tilde{\mathcal{M}}_S$. These datasets (i.e., G, D_A, $\mathcal{M}_S$, and $\tilde{\mathcal{M}}_S$) are then used for model training in the next phase. Details are provided in Section V-B.

Phase ②: Model Development and Training. DPDeno uses STGAE, an encoder-decoder architecture that refines DPST matrices by refining them toward structural patterns derived from G and D_A. The encoder maps each noisy matrix $\tilde{\mathcal{M}}_{s_i}$ into a latent representation capturing real-world mobility structure, while the decoder reconstructs the corresponding clean matrix $\mathcal{M}_{s_i}$ from the latent representation. During training, STGAE minimizes a combined loss comprising node-level loss $\mathcal{L}_{\text{node}}$ and edge-level loss $\mathcal{L}_{\text{edge}}$, ensuring accurate reconstruction of the clean mobility features. Further details are provided in Section V-C.

Phase ③: Refining with the Trained Model. After training, DPDeno applies the same DP method to the ground-truth dataset D to generate a DPST mobility matrix $\tilde{\mathcal{M}}$. It then uses STGAE to refine $\tilde{\mathcal{M}}$, producing the final output $\hat{\mathcal{M}}$ for release. Further details are provided in Section V-D.

B. Knowledge-Driven Data Generation

This subsection corresponds to Phase ① of DPDeno, which focuses on knowledge collection and knowledge-driven data generation.

Knowledge Collection. We obtain the target city's road network G and its attribute set D_A from OpenStreetMap (OSM) using OSMNX [43]. These attributes—including speed limits, segment lengths, access permissions, and one-way restrictions—define the real-world constraints for our trajectory synthesis. A comprehensive list of supported attributes is provided in [43].

Knowledge-Driven Data Generation. Guided by the observation that, in practice, drivers often choose shortest-time routes (e.g., via navigation apps), we generate synthetic trajectory datasets using a shortest-travel-time strategy under the constraints in D_A. We then apply a DP mechanism (e.g., DNI, SNI, NCNI) to produce paired ST and DPST mobility matrices. Algorithm 1 outlines the overall procedure.

Algorithm 1 comprises three stages: (1) time- and rule-aware graph construction, (2) shortest-travel-time trajectory synthesis, and (3) training data generation. The overall goal is to produce DPST matrices that capture real-world traffic behaviors and can support downstream training.

(1) Rule- and Time-Aware Graph Construction (lines 1-5). To ensure the synthesized trajectories conform to real-world constraints, we first transform the road network G into a directed graph G_d through two steps:

1) **Filtering.** We remove road segments that are inaccessible due to physical constraints, violate one-way traffic rules, or are blocked by turn bans. This step ensures the generated paths mimic legal and feasible driving routes.
2) **Time-dependent weighting.** For each remaining edge $e \in E$, we assign a travel-time weight based on temporal traffic speed:

$$w_e(t) = \frac{\text{length}(e)}{\text{speed}(e,t)}, \tag{8}$$

where $\text{length}(e)$ denotes the physical length of segment e, and $\text{speed}(e,t)$ represents the estimated travel speed at time t. When available, we compute $\text{speed}(e,t)$ using historical two-hour average data; otherwise, we default to the legal speed limit. These weights serve as time-dependent costs in the graph G_d.

(2) Shortest-Travel-Time Trajectory Synthesis (lines 6-14). To create realistic synthetic trajectories, we randomly sample a starting intersection v_s, a destination v_d, and a starting time t_s. Using Dijkstra's algorithm [44], we compute the shortest-travel-time path from v_s to v_d under the current time-dependent weights. During this process, each segment's traversal time is adjusted dynamically based on t_s and its temporal speed. If no valid path is found (e.g., due to disconnected components or extreme congestion), the algorithm resamples the pair (v_s, v_d) and repeats. This ensures that each trajectory complies with traffic rules and reflects plausible urban mobility patterns. The process continues until we generate N valid trajectories, forming the synthetic trajectory set D_s.

(3) Training Data Generation (line 15). We transform D_s into an ST mobility matrix $\mathcal{M}_s$ following the discretization process described in Section III-A. Then, a DP mechanism is applied to $\mathcal{M}_s$ to obtain its DPST version $\tilde{\mathcal{M}}_s$, simulating the noisy input the system would encounter during deployment.

To improve generalizability and ensure adequate training diversity, DPDeno repeats Algorithm 1 L times, producing a collection of matrices $\mathcal{M}_S = \{\mathcal{M}_{s_i}\}_{i=1}^{L}$ and their differentially private counterparts $\tilde{\mathcal{M}}_S = \{\tilde{\mathcal{M}}_{s_i}\}_{i=1}^{L}$. These DPST matrices are then used to train STGAE in Phase ② of DPDeno.

Algorithm 1 Knowledge-Driven Training Data Generation

Input: Road network G, attribute data D_A, number of trajectories N

Output: ST mobility matrix $\mathcal{M}_s$, DPST matrix $\tilde{\mathcal{M}}_s$

// 1. Rule- and Time-Aware Graph Construction

$G_d \leftarrow G$

Remove segments that are inaccessible, violate one-way rules, or are restricted by turn bans

for all edge $e \in G_d$ **do**

  $w_e(t) \leftarrow \text{length}(e)/\text{speed}(e,t)$

end for

// 2. Shortest-Travel-Time Trajectory Synthesis

Initialize $D_s \leftarrow \emptyset$

while $|D_s| < N$ **do**

  Randomly select start v_s, destination v_d, and time t_s

  $\tau \leftarrow \text{Dijkstra}(G_d, v_s, v_d, t_s)$

  if v_d not reached **then**

    continue

  end if

  Append τ to D_s

end while

// 3. Training Data Generation

Create $\mathcal{M}_s$ and $\tilde{\mathcal{M}}_s$ from D_s using a DP method

return $\mathcal{M}_s$, $\tilde{\mathcal{M}}_s$

C. Model Development and Training

This subsection corresponds to Phase ② of DPDeno. We propose an STGAE model to handle the noise in the DPST mobility matrix. In what follows, we introduce the development and training of the model.

1) Model Development: We now detail the development of our model, which consists of two key modules: graph representation module and STGAE model module.

Graph Representation Module. The first module, the graph representation module, models the spatial properties of the road network and the ST mobility matrix as directed graphs, providing a more precise and information-rich representation for subsequent model training. To integrate the ST mobility matrix with detailed road network information, we model the road network G, the geographic attribute data D_A, and both the ST mobility matrix $\mathcal{M}_s$ and the DPST mobility matrix $\tilde{\mathcal{M}}_s$ as directed graphs: $\mathcal{G}_s = (N, E, X, Y_s)$ and $\tilde{\mathcal{G}}_s = (N, E, X, \tilde{Y}_s)$, respectively. Specifically, given any ST mobility matrix $\mathcal{M}$, we first construct the set of segment transition states $E = \{e_1, e_2, \ldots, e_m\}$ by considering all possible segment transitions, combining any two segments r_i and r_j in G into directed transitions $(r_i \rightarrow r_j)$, and removing transitions where r_i and r_j are not directly connected in the topology of G. Next, we define the graph $\mathcal{G} = (N, E, X, Y)$, where (1) the set of road segments R in G is represented as the set of nodes N in $\mathcal{G}$, (2) the set of segment transition states E is represented as the set of edges E in $\mathcal{G}$, (3) X is a node attribute tensor $\in \mathbb{R}^{|T|\times n\times d}$ derived from D_A, and (4) Y is an edge attribute tensor $\in \mathbb{R}^{|T|\times m\times 1}$ derived from $\mathcal{M}$. Here, n denotes the number of nodes (i.e., road segments), m denotes the number of edges (i.e., transition states), and d represents the dimensionality of the features for each node. Since the road segments are fixed, the graph structure remains static, meaning the nodes and edges do not change over time. However, both X and Y vary over time: in X, each node r_i has a time series $x_i \in \mathbb{R}^{|T|\times d}$, capturing d-dimensional attributes such as location, speed limit, and accessibility; in Y, each edge e_j carries a time series $y_j \in \mathbb{R}^{|T|\times 1}$ recording the count of the transition state e_j over time. Finally, we assign weights W_e to each edge e based on the centroid distance [45], [46] between the two segments it connects:

$$W_e = \exp\left(-\frac{dis^2(e)}{2\sigma^2}\right), \tag{9}$$

where $dis(e)$ denotes the Euclidean distance between the centroids of the two segments connected by e, and σ is the standard deviation of these distances. Before subsequent training, DPDeno models the synthetic ST mobility matrices $\mathcal{M}_s$ and $\tilde{\mathcal{M}}_s$ as $\mathcal{G}_s$ and $\tilde{\mathcal{G}}_s$

STGAE Model Module. The second module, the STGAE model, refines this graph representation by leveraging node features extracted by the node module to better align edge features derived from the edge module, helping ensure that the refined graph more accurately reflects the underlying ST patterns. This model includes both an encoder and a decoder, each utilizing node and edge modules to process and enhance the graph data. Below, we describe the roles of these modules within the encoder and decoder.

Node Module. The node module consists of a spatial layer (SL) positioned between two temporal layers (TL), as depicted in Figure 4. In the encoder, the node module maps the input node features X into low-dimensional node embeddings H_X, effectively capturing deep features of the road segments. In the decoder, this module restores H_X to its original dimensions, producing the reconstructed node features $\hat{X}$. This ensures that the node features maintain their spatial and temporal integrity throughout the reconstructed process.

Edge Module. The edge module consists of two TLs and a multi-layer perceptron (MLP), as illustrated in Figure 4. In the encoder, the edge module first maps the input edge features Y into low-dimensional embeddings H_Y through the TLs, capturing deep representations of the transition states. At each time step t, for an edge e, we denote its embedding as $H(e,t)$. Let n_{e_0} and n_{e_1} be the source and target nodes connected by e, their embeddings at time t are $H_Y(n_{e_0}, t)$ and $H_Y(n_{e_1}, t)$, respectively. These three vectors are concatenated and fed into the MLP to produce a refined edge representation $\tilde{H}(e,t)$:

$$\tilde{H}(e,t) = \text{MLP}\left(\text{concat}(H_Y(n_{e_0}, t), H_Y(n_{e_1}, t))\right). \tag{10}$$

In the decoder, the edge module mirrors this process, reconstructing the original edge features $\hat{Y}$ from the refined edge representations $\tilde{H}(e,t)$.

Temporal Layer. Both the node and edge modules use TLs to capture temporal dependencies in node and edge features, such as correlations in the time series of road attributes and segment transitions. While the structure of the TLs in both the encoder and decoder is similar, they differ in the type of convolution applied. In the encoder, the TLs use gated 1D convolutions to aggregate neighboring values in the time

series of road features and segment transitions. In contrast, the decoder uses transposed gated 1D convolutions, which perform deconvolution to restore the time series to its original length, effectively reconstructing the time series from the aggregated data in the encoder. Gated linear units (GLUs) GLU_x and GLU_y are used as activation functions, defined as follows:

$$\text{GLU}_x = (\Omega * x) \odot \sigma(\Omega * x), \quad \text{GLU}_y = (\Omega * y) \odot \sigma(\Omega * y), \tag{11}$$

where σ is the Sigmoid function, and $\odot$ represents element-wise multiplication. For both the encoder and decoder, we use a filter kernel size of 4, a stride of 2, and padding of 1. This configuration halves the time series length in the encoder and doubles it in the decoder, ensuring that the input and output time series dimensions remain consistent across the STGAE model.

Spatial Layer. To effectively capture spatial features of nodes, we designed SLs in both the node and edge modules, using graph convolutions to aggregate information from neighboring nodes. For this, we employ the Chebyshev spectral graph convolution operator (ChebConv) [45]–[47] as our SL. ChebConv is particularly efficient for aggregating multi-hop neighbor information through the use of Chebyshev polynomials. This operator significantly reduces the computational complexity of spectral convolutions, making it a computationally efficient and powerful method for capturing spatial dependencies in large graph datasets.

2) Model Training: During the model training phase, DPDeno optimizes the STGAE model by minimizing the reconstruction loss, which is composed of two components: (1) edge reconstruction loss $\mathcal{L}_{edge}$ and (2) node reconstruction loss $\mathcal{L}_{node}$. For a given training graph $\mathcal{G}_s$ and its noisy counterpart $\tilde{\mathcal{G}}_s$, $\mathcal{L}_{edge}$ is defined as the mean squared error (MSE) between the reconstructed edge features $\hat{Y}_s$ and the original, noise-free features Y_s:

$$\mathcal{L}_{edge}(\hat{Y}_s, Y_s) = \frac{1}{m \times |T|} \sum_{i=1}^{m} \sum_{k=1}^{|T|} (\hat{y}_{i,k} - y_{i,k})^2, \tag{12}$$

where $\hat{y}_{i,k}$ and $y_{i,k}$ represent the counts of the transition state e_i at time period t_k in $\hat{Y}_s$ and Y_s, respectively. Similarly, $\mathcal{L}_{node}$ is defined as the MSE between the reconstructed node features $\hat{X}_s$ and the original node features X_s:

$$\mathcal{L}_{node}(\hat{X}_s, X_s) = \frac{1}{n \times d \times |T|} \sum_{i=1}^{n} \sum_{j=1}^{d} \sum_{k=1}^{|T|} (\hat{x}_{i,j,k} - x_{i,j,k})^2, \tag{13}$$

where $\hat{x}_{i,j,k}$ and $x_{i,j,k}$ denote the jth dimensional attribute of node r_i at time period t_k in $\hat{X}_s$ and X_s, respectively. Finally, let θ represent all trainable parameters. The learning objective of our STGAE model is to optimize θ by minimizing the total loss $\mathcal{L}$ across the entire dataset $\mathcal{G}_S$:

$$\underset{\theta}{\operatorname{argmin}} \mathcal{L} = \frac{1}{|\mathcal{G}_S|} \sum_{\mathcal{G}_s \in \mathcal{G}_S} \left(\mathcal{L}_{edge}(\hat{Y}_s, Y_s) + \lambda \mathcal{L}_{node}(\hat{X}_s, X_s) \right), \tag{14}$$

where λ is a hyper-parameter that balances the influence of the two loss components. The edge reconstruction loss allows the model to refine $\tilde{Y}_S$, while the node reconstruction loss ensures that deep features from X_S are effectively learned to aid in the reconstructed process.

D. Refining with the Trained Model

This subsection corresponds to Phase ③ of DPDeno. After training in Phase ②, DPDeno applies the trained STGAE to the DPST mobility matrix $\tilde{\mathcal{M}}$, generating a refined matrix $\hat{\mathcal{M}}$ that better approximates the ground-truth matrix $\mathcal{M}$. Specifically, we first create $\tilde{\mathcal{M}}$ from the ground-truth dataset D using the same DP method employed in Phase ①. We represent it as a directed graph $\tilde{\mathcal{G}} = (N, E, X, \tilde{Y})$, which incorporates the road network G and geographic attributes D_A, as described in Phase ②. The trained STGAE then processes $\tilde{\mathcal{G}}$ to produce a refined graph $\hat{\mathcal{G}}$. The encoder transforms node features X and edge features $\tilde{Y}$ into latent representations, while the decoder reconstructs node features $\hat{X}$ and refines edge features $\hat{Y}$ with $\hat{X}$, allowing $\hat{Y}$ to better capture realistic ST patterns. Finally, the refined edge features $\hat{Y}$ are converted into the refined ST mobility matrix $\hat{\mathcal{M}}$, which more closely reflects the ground-truth matrix $\mathcal{M}$ than the noisy input $\tilde{\mathcal{M}}$. The refined matrix $\hat{\mathcal{M}}$ can then be released for downstream tasks such as mobility trend analysis and hotspot recommendation.

E. Privacy Analysis

DPDeno maintains the original privacy guarantee of the DPST features provided by upstream DP methods. Both the model training and refinement steps are conducted solely on the outputs of the DP methods, with no access to the original sensitive data. Since these steps are purely post-processing, they do not introduce additional privacy risks or weaken the privacy guarantees. This is formalized in the following theorem.

Theorem 2. *If the upstream DP method producing the DPST matrix $\hat{\mathcal{M}}$ satisfies ϵ-DP, then DPDeno also satisfies ϵ-DP.*

Proof. The DPST matrix $\hat{\mathcal{M}}$ used by DPDeno is generated by an upstream DP method that satisfies ϵ-DP. In DPDeno, the model training step is performed using only publicly available data, completely independent of the original sensitive dataset. The refinement process operates solely on $\hat{\mathcal{M}}$, without any reference to the original data. Therefore, DPDeno always adheres to the post-processing property of DP (Definition 4), ensuring that the privacy guarantee of ϵ-DP remains intact throughout the process. □

VI. Experimental evaluation

In this section, we first introduce the detailed experimental setup. Then, we conduct experiments on utility to illustrate the superiority of our proposed DPDeno framework.

A. Experimental Setup

Datasets. Our experiments are conducted using two real-world trajectory datasets, as summarized in Table II.

- **Didi**[1]: This dataset was collected by Didi Chuxing from the Chengdu urban area between November 1st and 31st, 2016.

[1]https://gaia.didichuxing.com

TABLE II: Statistics of the datasets used in our experiments.

Dataset	Size	Average Length	Sampling Interval
Didi	194,303	3.71 km	3 sec
Porto	587,346	3.42 km	15 sec

For our analysis, we focus on 194,303 trajectories extracted from November 15th.

- **Porto**[2]: This dataset includes taxi trajectories collected over 8 months in the city of Porto. We use the entire dataset, consisting of 587,346 trajectories, treating it as representative of a single day's movement data for our experiments.

Road Network Extraction and Data Pre-Processing. We use OSMNX [43] to extract road networks and their associated attributes from OSM for each dataset. To process each raw trajectory, we first apply the FMM algorithm [41] to match its geographic coordinates to the road network, identifying the corresponding road segment route. Next, using a constant speed between consecutive geographic coordinates, we calculate the departure time and travel time for each road segment in the route using a linear interpolation method [30], [42]. This process results in an updated route with departure and travel times, referred to as the map-matched trajectory.

Baselines. To evaluate the effectiveness of our proposed framework DPDeno, we compare it against state-of-the-art baselines, which fall into two categories: (1) DPST mobility feature release methods [10], [13], and (2) refinement methods that enhance the utility of released DPST features [18], [19].

- **DPST feature release methods:** This category includes DNI, SNI, and NCNI, which directly generate DPST mobility matrices from the input data. These methods are detailed in Section IV-B1.
- **Post-processing refinement methods:** This category includes VDR [19] and PUTS [18], which apply post-processing to improve the utility of DPST features after release. These methods are introduced in Section II.

For a fair comparison, we first apply DNI, SNI, and NCNI to generate initial DPST mobility matrices. These matrices are then refined using DPDeno, VDR, and PUTS, resulting in combined methods: DNI-DPDeno, SNI-DPDeno, NCNI-DPDeno, DNI-VDR, SNI-VDR, NCNI-VDR, DNI-PUTS, SNI-PUTS, and NCNI-PUTS.

Parameter Settings. The key parameters involved in our method DPDeno and the baseline approaches include: (i) ϵ, the privacy budget used to construct DPST matrices directly from real data; (ii) L, the number of synthetic trajectory datasets generated during Phase ① (knowledge-driven data generation) of DPDeno; (iii) ϵ^*, the internal privacy budget used in Phase ① of DPDeno to construct DPST matrices from synthetic trajectories; and (iv) ℓ, the trajectory length used in the SNI mechanism.

By default, we set $\epsilon = 1.0$, following standard practices in DP-based trajectory publishing [10], [13], [18]. The number of synthetic trajectories L is set to 100, which ensures sufficient sample diversity to significantly enhance DPDeno's training performance while maintaining low computational cost. To ensure consistent noise distribution between the training and inference phases, we align the internal privacy budget with the external one, i.e., $\epsilon^* = \epsilon$. This alignment helps mitigate the distribution shift between synthetic and real DPST matrices and thus improves inference utility. The trajectory length ℓ in SNI is fixed at 50, which covers over 80% of trajectories in the real-world datasets used. In addition, we systematically evaluate the impact of varying the parameters on DPDeno's overall utility, including $\epsilon \in [0.5, 2.0]$, $L \in [20, 200]$, and $\epsilon^* \in [0.5, 2.0]$, as presented in Section VI-C.

The STGAE model in DPDeno is trained using the Adam optimizer [48]. A grid search is conducted to optimize hyper-parameters by exploring a discrete grid space [49]. Parameters considered include Chebyshev filter sizes $\in \{1, 2, 3, 4, 5, 6\}$, batch sizes $\in \{1, 2, 4, 8, 16, 32, 64\}$, kernel sizes $\in \{3, 4\}$, strides $\in \{2, 4\}$, $\lambda \in [0.04, 0.1]$, padding set to 1, $k = 3$, and the number of epochs $\in \{25, 50, 100\}$. The learning rate is set to 0.001 with a decay rate of 0.02.

Environment Settings. All experiments were conducted using PyTorch Geometric Temporal [50] on a system equipped with a 13th Gen Intel(R) Core(TM) i9-13900HX CPU @ 2200 MHz, 32 GB of memory, 24 CPU cores, and an 24GB NVIDIA GeForce RTX 4090 Laptop GPU.

B. Evaluation Metrics

We evaluate the performance of the released DPST mobility features from two complementary perspectives: statistical accuracy and practical utility in downstream tasks.

Statistical Accuracy. Let $\mathcal{M}$ denote the ST mobility matrix derived from the ground-truth dataset D, and let $\hat{\mathcal{M}}$ represent the released mobility matrix. The statistical accuracy evaluates the distance between $\mathcal{M}$ and $\hat{\mathcal{M}}$, which is calculated using the mean squared error (MSE) defined as:

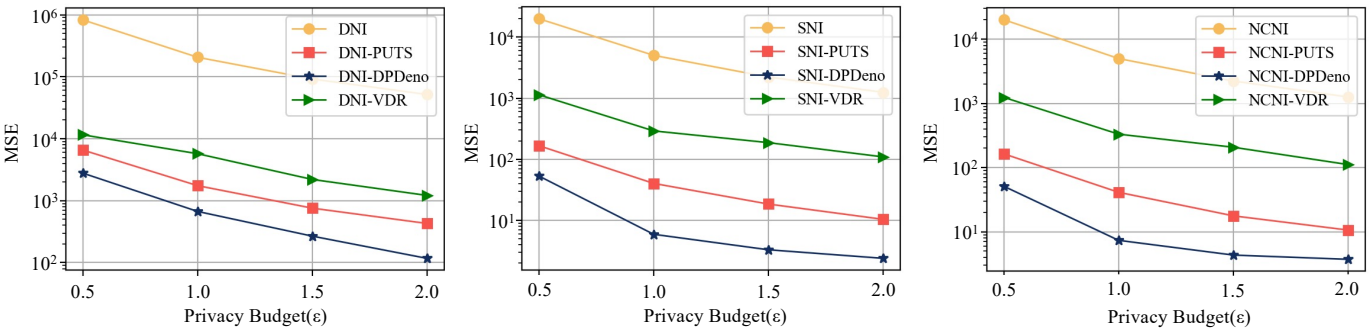
$$\text{MSE} = \frac{1}{n \times n \times |T|} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{|T|} (\mathcal{M}(i,j,k) - \hat{\mathcal{M}}(i,j,k))^2. \tag{15}$$

Practical Utility. We evaluate the practical utility of the released mobility matrix through two common real-world applications: frequent pattern (FP) mining and range count queries (RCQs) [14], [18].

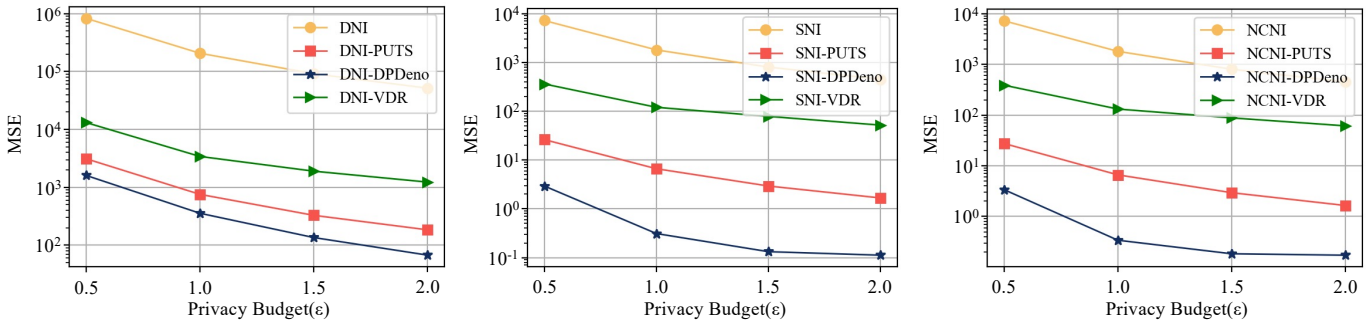
Utility in Frequent Pattern Mining. FP mining is essential in trajectory data analysis, often used to predict traffic flow and mobility trends. To evaluate the utility of the released mobility matrix in FP mining, we first define users' movement patterns as segment transitions, $P : r_i \rightarrow r_j$. The support of a pattern, $\text{supp}(\mathcal{M}, P, T_c)$, denotes the number of times P occurs in $\mathcal{M}$ during a specified time window $T_c \subseteq T$. We then mine the top-k frequent patterns in $\mathcal{M}$ over T_c, denoted as $\mathcal{F}_k(\mathcal{M}, T_c)$, representing the k patterns with the highest support. To evaluate the difference between $\mathcal{M}$ and $\hat{\mathcal{M}}$, we calculate the average error in support for each pattern $P \in \mathcal{F}_k(\mathcal{M}, T_c)$ using the following formula:

$$\text{FP AvE} = \frac{1}{k} \sum_{P \in \mathcal{F}_k(\mathcal{M}, T_c)} |\text{supp}(\mathcal{M}, P, T_c) - \text{supp}(\hat{\mathcal{M}}, P, T_c)|. \tag{16}$$

[2]http://www.geolink.pt/ecmlpkdd2015-challenge/dataset.html



(a) Didi Dataset



(b) Porto Dataset

Fig. 5: MSE Results on Different Datasets with Varying Privacy Budgets.

Utility in Range Count Queries (RCQs). RCQs play a crucial role in spatio-temporal data analysis, frequently utilized for applications such as location recommendation and hotspot detection. While the released mobility matrices cannot be directly applied to RCQs, they can be leveraged to generate synthetic trajectory datasets, which are suitable for RCQs and other tasks. As such, we first use the refined matrix $\tilde{\mathcal{M}}$ to generate a synthetic trajectory dataset D_{syn}. For all methods, the length of synthetic trajectories is set to $(|\tau_{\max}| + 1)/2$. A segment r_s is randomly chosen from R as the starting point, and subsequent segments are generated via random walk on $\tilde{\mathcal{M}}$. This process is repeated until D_{syn} is created, containing $|D_{\text{syn}}| = |D|$ synthetic trajectories. Finally, we assess its similarity to the ground-truth dataset D by performing RCQs. Specifically, RCQs are defined as: "Retrieve the number of trajectories passing through a specified region." A query set Q consisting of 200 RCQs is generated, with each query centered on a randomly selected point in the dataset. Following [13], [14], we vary the query window side length uniformly from one-third to two-thirds of the study area's maximum dimension. This range balances overall coverage and local focus, enabling realistic range queries and systematic evaluation of ST mobility feature performance under different privacy settings. For all queries, we compute the average relative error (AvRE) using the following formula:

$$\text{RCQ AvRE} = \frac{1}{|Q|} \sum_{q \in Q} \frac{|q(D) - q(D_{\text{syn}})|}{\max\{q(D), \psi\}}, \tag{17}$$

where ψ is set to $1\% \times |D|$ to prevent division by zero, and $q(D)$ and $q(D_{\text{syn}})$ are the query results for the ground-truth and synthetic datasets, respectively.

In summary, a method is considered more effective if it produces a mobility matrix $\tilde{\mathcal{M}}$ with lower values in all three metrics. Specifically, lower MSE indicates better statistical fidelity, smaller FP AvE suggests improved frequent pattern recovery, and reduced RCQ AvRE reflects more accurate responses to range count queries.

C. Performance Comparison with Baselines

In this subsection, we first compare DPDeno to the baseline methods, evaluating their performance in terms of statistical accuracy and practical utility. Next, we assess the effect of varying L on the performance of DPDeno when releasing DPST mobility matrices using the Didi dataset.

1) Statistical Accuracy: Figure 5 presents the MSE results for the compared methods across both datasets. The key observations are as follows:

DPDeno Effectively Reduces Noise in Released DPST Mobility Features. The results demonstrate that DPDeno consistently reduces the MSE of released ST mobility features compared to the baseline methods. For example, NCNI-DPDeno achieves an average MSE reduction of 99.68% compared to NCNI, 73.79% compared to NCNI-PUTS, and 96.67% compared to NCNI-VDR on the Didi dataset. Similarly, on the Porto dataset, NCNI-DPDeno reduces MSE by 99.96% compared to NCNI, 89.46% compared to NCNI-PUTS, and 99.38% compared to NCNI-VDR.

These improvements are largely attributed to STGAE, the spatio-temporal graph autoencoder at the core of DPDeno. Unlike PUTS, which improves utility by filtering transitions that violate topological constraints, STGAE corrects noise-induced distortions even when the transitions are topologically valid but statistically implausible. It learns to refine noisy DPST

features toward realistic patterns by leveraging both spatial dependencies (e.g., road connectivity and directionality) and temporal dynamics (e.g., speed limits). Compared to VDR, which compresses noisy histograms into low-dimensional representations but often discards useful structural information in the process, STGAE enforces consistency between node-level and edge-level representations during decoding, enabling more accurate recovery of the ST mobility matrix.

DPDeno is Robust Across Privacy Budgets and Datasets. DPDeno consistently achieves strong performance across varying privacy budgets (ϵ) and different datasets. While all methods benefit from increased ϵ due to reduced DP noise, DPDeno maintains high utility even under small ϵ values. This robustness is supported by two key factors: (1) the model structurally refines DPST features toward realistic mobility patterns derived from public data, and (2) it is trained on synthetic datasets perturbed by DP methods with noise levels tailored to each target privacy budget. This ensures better generalization across noise intensities.

By contrast, PUTS cannot restore topologically valid transitions that are distorted by heavy noise, while VDR's compression strategy tends to lose both noise and signal, limiting its effectiveness under both low and high noise levels. As a result, while their performance drops sharply under strict privacy settings, DPDeno remains stable, leading to more pronounced relative gains. Furthermore, the consistent improvements observed on both the Didi and Porto datasets—despite their differences in urban structure and traffic dynamics—demonstrate the generalizability and practical value of DPDeno in real-world deployments.

2) Practical Utility: Figures 6 and 7 present the FP AvE and RCQ AvRE results for the compared methods on the Didi dataset. The key observations are as follows:

DPDeno Significantly Enhances the Practical Utility of Released DPST Mobility Features. The results demonstrate that DPDeno consistently improves the practical utility of released DPST mobility features in real-world applications. Across all scenarios, it yields lower FP AvE and RCQ AvRE values compared to both release-only and enhancement-based methods. Specifically, in FP mining, DNI-DPDeno reduces FP AvE by 39.75% compared to DNI, 18.85% compared to DNI-PUTS, and 23.83% compared to DNI-VDR. Likewise, SNI-DPDeno reduces FP AvE by 41.29% compared to SNI, 29.72% compared to SNI-PUTS, and 32.84% compared to SNI-VDR. The greatest improvement is seen with NCNI-DPDeno, which lowers FP AvE by 47.02% relative to NCNI, 20.73% compared to NCNI-PUTS, and 26.27% compared to NCNI-VDR. In RCQ tasks, DNI-DPDeno reduces RCQ AvRE by 54.44% relative to DNI, 28.71% compared to DNI-PUTS, and 44.93% compared to DNI-VDR. For SNI, the reductions are 58.75%, 25.76%, and 42.74%, respectively. For NCNI, DPDeno lowers RCQ AvRE by 46.49% compared to NCNI, 20.18% over NCNI-PUTS, and 34.90% over NCNI-VDR.

These improvements are primarily attributed to the design of STGAE, which learns from real-world structural constraints and spatio-temporal patterns to refine DPST features. Compared to PUTS, which only uses topological filtering, and VDR, which compresses information into low-dimensional latent space, DPDeno achieves superior refinement with realistic mobility behaviors, thereby enhancing practical utility.

DPDeno Demonstrates Strong Robustness in Real Applications. DPDeno consistently delivers robust performance across different DP methods and privacy budgets, highlighting its broad applicability in real-world scenarios. As shown in Figures 6 and 7, DPDeno provides steady improvements in both FP mining and RCQ tasks under varying levels of privacy. Notably, its advantage becomes more pronounced under tighter privacy settings (e.g., $\epsilon = 0.5$ and $\epsilon = 1.0$), where the DP methods typically suffer from high noise. In these cases, DPDeno maintains low error rates and achieves significantly better utility, demonstrating its suitability for applications requiring strong privacy guarantees. This robustness stems from STGAE's ability to capture real-world structural constraints, making it an effective post-processing solution regardless of the noise level introduced by upstream DP mechanisms.

D. Parameter Study of DPDeno

In this subsection, we systematically evaluate the impact of the number of synthetic trajectories (L) and the internal privacy budget (ϵ^*) on the utility of DPDeno. All experiments are conducted on the Didi and Porto datasets, with other parameters set to their default values.

Effect of L. As shown in Figure 8, we vary the number of synthetic trajectories L from 20 to 200 and observe the performance of DPDeno under the three DP mechanisms: DNI, SNI, and NCNI. The results exhibit two notable trends. First, as L increases, the overall MSE of DPDeno consistently decreases. This is because a larger L provides a richer set of synthetic data samples, enabling the spatio-temporal graph autoencoder in DPDeno to be more effectively trained. As a result, it can better reconstruct the clean version of the ST matrix from the noisy DPST input, thereby improving the refinement effect on real DPST matrices. Second, the improvement in utility gradually diminishes as L grows, showing a trend of diminishing returns. This indicates that once L reaches a sufficient scale, further increasing the number of synthetic samples brings only marginal gains in the utility of the refined DPST matrix.

Effect of ϵ^*. We further evaluate the effect of the internal privacy budget $\epsilon^* \in \{0.5, 1.0, 1.5, 2.0\}$ on the utility of DPDeno. As shown in Figure 9, the best refinement results are achieved when $\epsilon^* = \epsilon = 1.0$, meaning the synthetic DPST matrices used in training and the real DPST matrices used in inference share the same noise level. This consistency helps mitigate distributional shifts and enhances the refinement performance of DPDeno on real DPST matrices. In contrast, when $\epsilon^* = 0.5$, the synthetic DPST matrices contains excessive noise, causing DPDeno to over-correct the real DPST matrices and degrade utility. On the other hand, when $\epsilon^* = 1.5$ or 2.0, the noise in the synthetic DPST matrices is insufficient, leading DPDeno to under-correct the real DPST matrices, which also harms utility. These results demonstrate that consistent privacy budgets between the training and inference stages are essential for achieving optimal refinement performance.

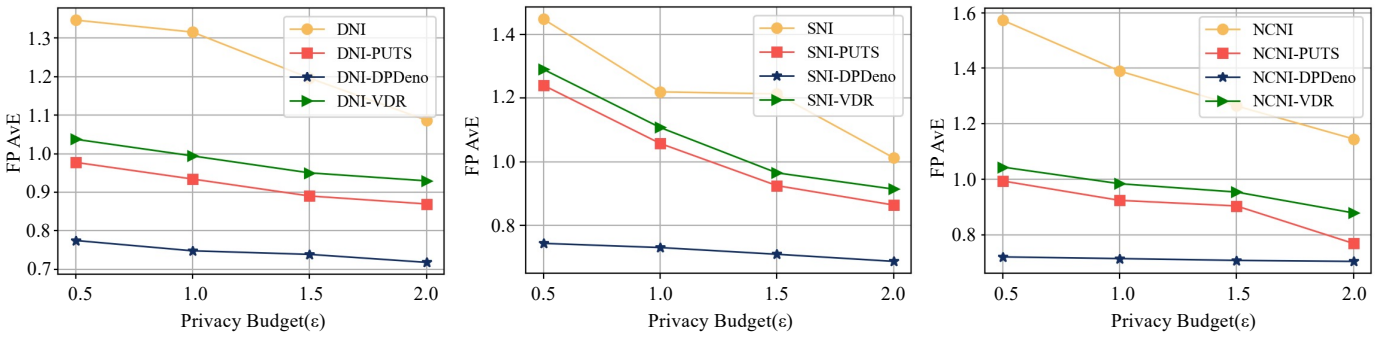


Fig. 6: FP AvE Results on Didi Datasets with Varying Privacy Budgets.

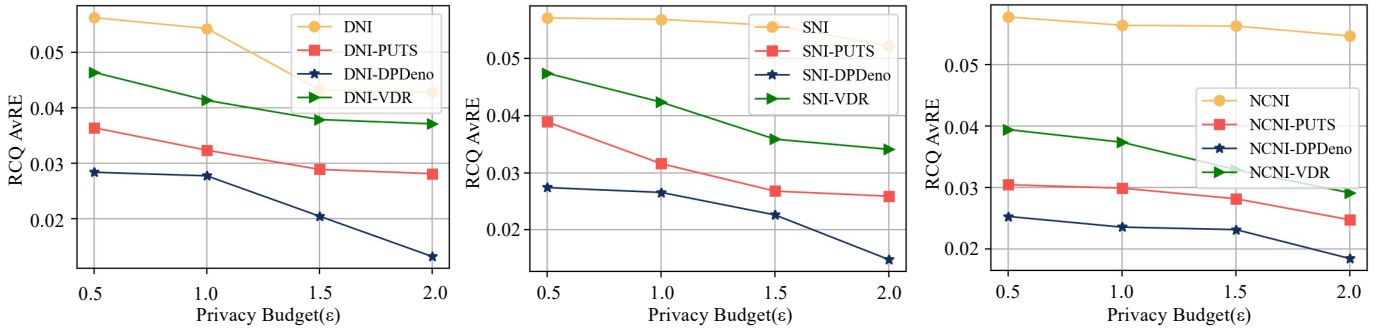


Fig. 7: RCQ AvRE Results on Didi Datasets with Varying Privacy Budgets.

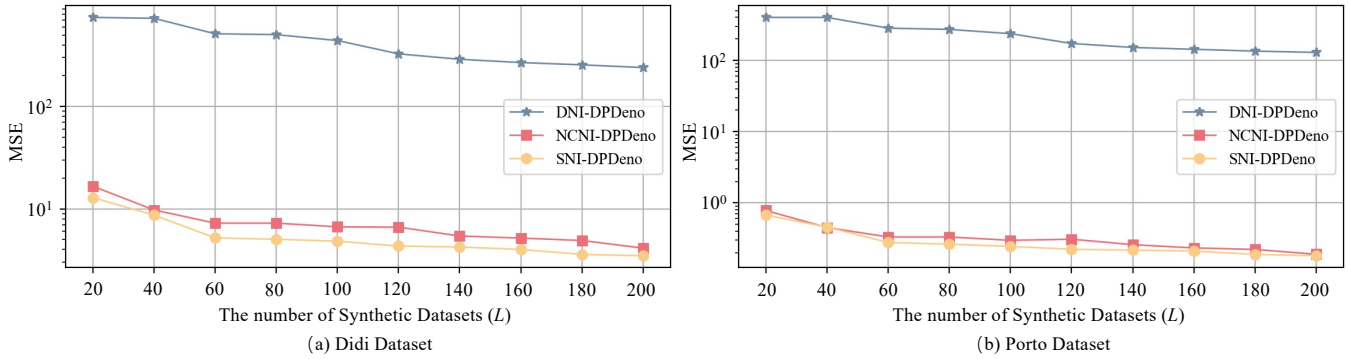


Fig. 8: MSE on Didi Datasets with Varying Synthetic Trajectory Numbers (L).

VII. Conclusions

In this paper, we presented DPDeno, a novel framework for releasing high-utility ST mobility matrices while ensuring ϵ-DP. By leveraging the STGAE model, DPDeno extracts patterns from real-world knowledge to effectively refine the DPST mobility matrices generated by existing DP methods. Extensive experiments on real-world datasets, including Didi and Porto, demonstrated that DPDeno significantly enhances the utility of the released mobility features in terms of both statistical accuracy and practical applications. In future work, we aim to extend DPDeno to support more complex mobility patterns and explore its applicability to other privacy-sensitive domains.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 62402137, 62472122, 62402024 and 62372122), the National Key R&D Program of China (Grant No. SQ2025YFE0200461 and 2024YFB4505901), the Postdoctoral Fellowship Program of CPSF under Grant Number GZC20242205, the Funds for International Cooperation and Exchange of the NSFC (No. 62461160332), the Joint Funds of the NSFC (Grant No. U22A2036), the Beijing Natural Science Foundation (No. L241050).

References

[1] Z. Zheng, Z. Li, H. Jiang, L. Y. Zhang, and D. Tu, "Semantic-aware privacy-preserving online location trajectory data sharing," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2256–2271, 2022.

[2] L. Wu, C. Qin, Z. Xu, Y. Guan, and R. Lu, "Tcpp: Achieving privacy-preserving trajectory correlation with differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4006–4020, 2023.

[3] S. Gao, J. Ma, W. Shi, G. Zhan, and C. Sun, "Trpf: A trajectory privacy-preserving framework for participatory sensing," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, pp. 874–887, 2013.

[4] R. Li, H. Hu, and Q. Ye, "Rftrack: Stealthy location inference and tracking attack on wi-fi devices," *IEEE Transactions on Information Forensics and Security*, 2024.

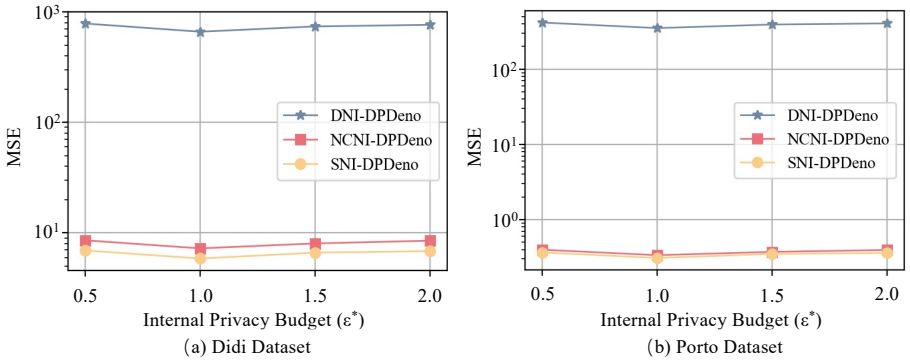


Fig. 9: MSE on Didi Datasets with Varying Internal Privacy Budget (ϵ^*).

[5] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*, 2006.

[6] T. Wang, J. Q. Chen, Z. Zhang, D. Su, Y. Cheng, Z. Li, N. Li, and S. Jha, "Continuous Release of Data Streams under both Centralized and Local Differential Privacy," in *ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1237–1253.

[7] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, "CALM: Consistent adaptive local marginal for marginal release under local differential privacy," in *ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 212–229.

[8] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized aggregatable privacy-preserving ordinal response," in *ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 1054–1067.

[9] R. Chen, B. C. Fung, B. C. Desai, and N. M. Sossou, "Differentially private transit data publication: a case study on the montreal transportation system," in *Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 213–221.

[10] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, 2014.

[11] R. Chen, G. Acs, and C. Castelluccia, "Differentially private sequential data publication via variable-length n-grams," in *ACM SIGSAC Conference on Computer and Communications Security*, 2012, pp. 638–649.

[12] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava, "DPT: Differentially private trajectory synthesis using hierarchical reference systems," in *Proceedings of the VLDB Endowment*, 2015, pp. 2150–8097.

[13] M. E. Gursoy, L. Liu, S. Truex, L. Yu, and W. Wei, "Utility-aware synthesis of differentially private and attack-resilient location traces," in *ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 196–211.

[14] M. E. Gursoy, L. Liu, S. Truex, and L. Yu, "Differentially private and utility preserving publication of trajectory data," *IEEE Transactions on Mobile Computing*, vol. 18, no. 10, pp. 2315–2329, 2018.

[15] H. Wang, Z. Zhang, T. Wang, S. He, M. Backes, J. Chen, and Y. Zhang, "Privtrace: Differentially private trajectory synthesis by adaptive markov model," in *32nd USENIX Security Symposium*, 2023.

[16] Y. Zhang, Q. Ye, R. Chen, H. Hu, and Q. Han, "Trajectory data collection with local differential privacy," in *Proceedings of the VLDB Endowment*, vol. 16, no. 10, 2023, pp. 2591–2604.

[17] X. Sun, Q. Ye, H. Hu, Y. Wang, K. Huang, T. Wo, and J. Xu, "Synthesizing realistic trajectory data with differential privacy," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 5502–5515, 2023.

[18] X. Sun, Q. Ye, H. Hu, J. Duan, Q. Xue, T. Wo, and J. Xu, "Puts: Privacy-preserving and utility-enhancing framework for trajectory synthesization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 1, pp. 296–310, 2023.

[19] R. Ahuja, S. Zeighami, G. Ghinita, and C. Shahabi, "A neural approach to spatio-temporal data release with user-level differential privacy," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–25, 2023.

[20] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 603–618.

[21] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

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

[23] Y. Xiao and L. Xiong, "Protecting locations with differential privacy under temporal correlations," in *ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1298–1309.

[24] T. Cunningham, G. Cormode, and H. Ferhatosmanoglu, "Privacy-preserving synthetic location data in the real world," in *Proceedings of the 17th International Symposium on Spatial and Temporal Databases*, 2021, pp. 23–33.

[25] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[26] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Towards locally differentially private generic graph metric estimation," in *IEEE International Conference on Data Engineering*, 2020.

[27] Q. Ye, H. Hu, N. Li, X. Meng, H. Zheng, and H. Yan, "Beyond value perturbation: Local differential privacy in the temporal setting," in *INFOCOM*. IEEE, 2021, pp. 1–10.

[28] Q. Qian, Q. Ye, H. Hu, K. Huang, T. T.-L. Chan, and J. Li, "Collaborative sampling for partial multi-dimensional value collection under local differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3948–3961, 2023.

[29] Y. Mao, Q. Ye, H. Hu, Q. Wang, and K. Huang, "Privshape: Extracting shapes in time series under user-level local differential privacy," in *IEEE International Conference on Data Engineering*, 2024.

[30] Y. Wang, G. Li, K. Li, and H. Yuan, "A deep generative model for trajectory modeling and utilization," *Proceedings of the VLDB Endowment*, vol. 16, no. 4, pp. 973–985, 2022.

[31] N. Ponomareva, H. Hazimeh, A. Kurakin, Z. Xu, C. Denison, H. B. McMahan, S. Vassilvitskii, S. Chien, and A. G. Thakurta, "How to dp-fy ml: A practical guide to machine learning with differential privacy," *Journal of Artificial Intelligence Research*, vol. 77, pp. 1113–1201, 2023.

[32] J. Cai, Q. Ye, H. Hu, X. Liu, and Y. Fu, "Boosting accuracy of differentially private continuous data release for federated learning," *IEEE Transactions on Information Forensics and Security*, 2024.

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

[34] Y. Cao, Y. Xiao, L. Xiong, and L. Bai, "Priste: from location privacy to spatiotemporal event privacy," in *IEEE International Conference on Data Engineering*. IEEE, 2019, pp. 1606–1609.

[35] M. Cao, H. Zhu, M. Min, Y. Li, S. Li, H. Zhang, and Z. Han, "Protecting personalized trajectory with differential privacy under temporal correlations," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2024, pp. 1–6.

[36] H. Yuan, L. Wu, L. Xu, L. Ban, H. Wang, Y. Su, and W. Meng, "Sctp: Achieving semantic correlation trajectory privacy-preserving with differential privacy," *IEEE Transactions on Vehicular Technology*, 2024.

[37] D. Im Im, S. Ahn, R. Memisevic, and Y. Bengio, "Denoising criterion for variational auto-encoding framework," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.
[38] A. Krull, T.-O. Buchholz, and F. Jug, "Noise2void-learning denoising from single noisy images," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2129–2137.
[39] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," *arXiv preprint arXiv:1803.04189*, 2018.
[40] T. Pang, H. Zheng, Y. Quan, and H. Ji, "Recorrupted-to-recorrupted: Unsupervised deep learning for image denoising," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2043–2052.
[41] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden markov model with precomputation," *International Journal of Geographical Information Science*, vol. 32, no. 3, pp. 547–570, 2018.
[42] E. A. Wentz, A. F. Campbell, and R. Houston, "A comparison of two methods to create tracks of moving objects: linear weighted distance and constrained random walk," *International Journal of Geographical Information Science*, vol. 17, no. 7, pp. 623–645, 2003.
[43] G. Boeing, "Modeling and analyzing urban networks and amenities with osmnx," *Geographical Analysis*, 2024.
[44] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: his life, work, and legacy*, 2022, pp. 287–290.
[45] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018, pp. 3634–3640.
[46] Y. Fan, X. Yu, R. Wieser, D. Meakin, A. Shaton, J.-N. Jaubert, R. Flottemesch, M. Howell, J. Braid, L. Bruckman *et al.*, "Spatio-temporal denoising graph autoencoders with data augmentation for photovoltaic data imputation," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–19, 2023.
[47] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, 2016.
[48] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2022.
[49] F. J. Pontes, G. Amorim, P. P. Balestrassi, A. Paiva, and J. R. Ferreira, "Design of experiments and focused grid search for neural network parameter optimization," *Neurocomputing*, vol. 186, pp. 22–34, 2016.
[50] B. Rozemberczki, P. Scherer, Y. He, G. Panagopoulos, A. Riedel, M. Astefanoaei, O. Kiss, F. Beres, G. Lopez, N. Collignon *et al.*, "Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models," in *Proceedings of the 30th ACM international conference on information & knowledge management*, 2021, pp. 4564–4573.

Xinyue Sun is currently a postdoctoral fellow with the School of Cyberspace Science, Harbin Institute of Technology. He received his PhD degree from the School of Computer Science and Engineering, Beihang University in 2024. His research interests include data privacy and information security. His is a member of IEEE and CCF.

Xiaoyu Liu is currently a master's student at the School of Software, Beihang University, China. Her research focuses on privacy protection.

Qingqing Ye is an assistant professor in the Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University. She received her PhD degree from Renmin University of China in 2020. She has received several prestigious awards, including Hong Kong RGC Early Career Award, IEEE S&P Travel Award, and National Scholarship. Her research interests include data privacy and security, and adversarial machine learning. She is a member of IEEE, ACM and CCF.

Haibo Hu is a professor in the Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University. His research interests include adversarial machine learning, data privacy, database and network security, mobile, spatio-temporal and social databases. He has published over 170 research papers in refereed journals, international conferences, and book chapters. He is an associate editor of IEEE Transactions on Information Forensics and Security (TIFS), IEEE Transactions on Knowledge and Data Engineering (TKDE), and ACM Transactions on Privacy and Security (TOPS). He is a senior member of ACM, IEEE and CCF, and a certified Cisco CCNA Security Trainer.

Renyu Yang is currently an Associate Professor in the School of Software, Beihang University, China. He was with the University of Leeds UK, Alibaba Group China and Edgetic Ltd. UK, having industrial experience in building large-scale resource scheduling systems. He is a recipient of Alan Turing Post-Doctoral Enrichment Award, 2022. His research interests include parallel and distributed computing, and deep learning systems and applications. He is a member of IEEE.

Hui He received the PhD degree from the Department of Computer Science, Harbin Institute of Technology, China. She is currently a professor in network security center with the School of Cyberspace Science, China. Her research interests include mainly focused on distributed computing, IoT, and Big Data analysis.

Weizhe Zhang is currently a Professor with the School of Cyberspace Science, Harbin Institute of Technology, Harbin, China, and the Director of the Department of New Networks, Peng Cheng Laboratory, Shenzhen, China. He received the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 2006. He has authored or coauthored more than 130 academic papers in journals, books, and conference proceedings. His current research interests include cyberspace security, cloud computing, and high-performance computing. He is an associate editor of IEEE Transactions on Cloud Computing (TCC). He is a senior member of IEEE and ACM.