Resisting TUL Attack: Balancing Data Privacy and Utility on Trajectory via Collaborative Adversarial Learning

Yandi Lun[1], Hao Miao[2], Jiaxing Shen[3], Renzhi Wang[1], Xiang Wang[4], Senzhang Wang[1*]

[1*]Central South University, Changsha, China.
[2] Aalborg University, Aalborg, Denmark.
[3]Lingnan University, Hong Kong, China.
[4]National University of Defense Technology, Changsha, China.

*Corresponding author(s). E-mail(s): szwang@csu.edu.cn;
Contributing authors: yandilun@csu.edu.cn; haom@cs.aau.dk;
jiaxingshen@ln.edu.hk; rzwang@csu.edu.cn; xiangwangcn@nudt.edu.cn;

Abstract

Nowadays, large-scale individual trajectories can be collected by various location-based social network services, which enables us to better understand human mobility patterns. However, the trajectory data usually contain sensitive information of users, raising considerable concerns about the privacy issue. Existing methods for protecting user trajectory data face two major challenges. First, existing methods generally emphasize on data privacy but largely ignore the data utility. Second, most existing work focus on protecting the privacy of users' check-in locations, which is not sufficient to protect against the trajectory-user linking (TUL) attack that infers a user's identity based on her/his trajectories. In this paper, we for the first time propose a collaborative adversarial learning model named **BPUCAL** to effectively resist the TUL attack and preserve the data utility simultaneously. The general idea is to fool the TUL model by adding a small perturbation on the original trajectory data to balance the data utility and privacy. BPUCAL perturbs a few numbers of carefully identified check-ins of a trajectory which are pivotal for a TUL model to infer the identity of a user. Specifically, BPUCAL contains three parts: a perturbation generator, a discriminator, and a TUL model. The generator aims to produce learnable noise and adds it to the original trajectories for obtaining perturbed trajectories. The perturbed trajectories with a minimal changes compared to the original trajectories

can deceive both the discriminator and the TUL model. Extensive experiments are conducted over two real-world datasets. The results show the superior performance of our proposal in balancing data privacy and utility on trajectory data by comparison with baselines.

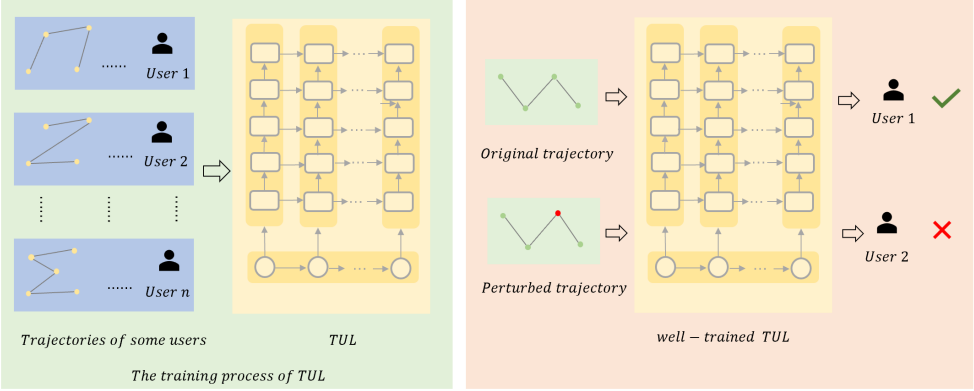
Keywords: Trajectory Privacy Protection, Data Utility, Adversarial Learning, Trajectory-User Linking

1 Introduction

With the wide application of sensor techniques in geographical information systems, various location-based services emerge, where individuals can expose their traces to service providers. A large number of individual trajectories provide us unprecedented opportunities to study human mobility patterns and facilitate many real applications such as POI recommendation [3, 21], human mobility prediction [46, 47], event crowd management [44], and traffic forecasting [4, 22, 45]. The massive available human trajectory data can facilitates people's travel, contributes to commercial development, government decision-making and so on.

However, releasing individual trajectories directly without any processing may lead to the risk of privacy leakage, because some sensitive information about users, such as addresses, financial situations, and habits can be inferred from their trajectories [39]. Due to the great concern on privacy leakage, users are not willing to share their trajectories with public. This limits the potential of mining valuable information from users' trajectories to facilitate various applications. To address this issue, a general practice is to remove the identifiers and publish the anonymous data. Some service providers, such as ridesharing services, protect user's privacy by generating large volumes of anonymous trajectories. However, these trajectories may still pose privacy risks as the spatio-temporal patterns and semantic information in the trajectories can be used as quasi-identifiers for linking the trajectories to their corresponding users. Recently, some attackers can infer the user identity of an anonymous trajectory by comparing the similarity between the anonymous trajectory and user's historical trajectories. With more trajectory-user linking (TUL) techniques developed to accurately link anonymous trajectories to users who have generated them, it becomes more difficult to prevent users from being re-identified. Figure 1 (a) illustrates the training process of a TUL model. the TUL model is trained in a supervised manner, using trajectories as input and using users as labels. A well-trained TUL model captures the behavior preferences of users, and when an anonymous trajectory is input into TUL, the model tries to infer the user who has generated the trajectory.

Most existing work on geographical data privacy protection focus on protecting the location privacy [29, 30], which are not sufficient to prevent the TUL model from identifying users through analyzing their entire trajectories [27]. Another line of studies focus on privacy protection for trajectory data, such as k-anonymity. However, k-anonymity has shown to be not effective to defend a user's identity when an attacker (e.g. a TUL model) has additional background knowledge [13]. Differential privacy [32]



(a) (b)

Fig. 1: An illustration of the training process of a TUL model and the basic idea of our method. (a) shows the training process of a TUL model. (b) illustrates the basic idea of our proposed method for resisting TUL attacks.

has been applied to confuse the attacker by adding noise to the raw data. Adding too much noise will remarkably change the semantic information of the trajectories, and thus reduce their utility. Recently, a deep learning-based method called LSTM-TrajGAN [8] is proposed to resist the attack of TUL for data publication. However, LSTM-TrajGAN will also reduce the utility of the trajectories due to its serious data distortion problem. Different from existing methods, in this paper, we aim to study such a novel problem: *can we propose a model that can resist the TUL attack, and at the same time do not sacrifice the data utility much?*

The challenges of the studied problem are two-fold. First, it is challenging to achieve a balance between data utility and privacy protection. For example, some methods protect the privacy by generating synthetic trajectories. However, while these synthetic trajectories might reduce the risk of user identification, they may differ significantly from the original trajectories, resulting in reduced utility. To preserve the utility of the data, most check-ins of a trajectory should not be perturbed. Therefore, a small number of check-ins that are critical for TUL attack should be identified for perturbation, rather than adding perturbation to all check-ins. As the goals of the two tasks are actually incompatible, it is very challenging to preserve data utility and protect data privacy simultaneously. Second, it is also challenging to identify a small number of check-ins in a trajectory that are more critical for a TUL attack. Not all check-ins on a trajectory are equally important for a TUL model to identify the user. For example, Judy and Lucy are roommates, they leave from the same house every day, go to the same subway station, and go to their companies for work. Judy's route can be represented as *(home1, subway station1, office1)*, and, Lucy's route can be represented as *(home1, subway station1, office2)*. We can find that only the *office* check-in is helpful

for distinguishing Lucy and Judy. However, in real application scenarios it is very difficult to use a simple statistics based method to compare the trajectories of different users to determine which check-ins are more critical for a TUL model to identify users.

In this paper, we propose a collaborative adversarial learning-based model BPUCAL to address the above challenges. To solve the first challenge, we propose to perturb a few points that are more critical for TUL attack. As for the second challenge, we employ a TUL module to enable our perturbation generator to produce perturbations for the identified critical check-ins in a trajectory. Specifically, BPUCAL contains three modules, the trajectory embedding module, the collaborative adversarial learning module, and the perturbed trajectory generation module. Given the original trajectory Tr, we first employ the trajectory embedding module to obtain the trajectory embedding X. Then, X is input into the collaborative adversarial learning module to obtain a perturbed trajectory embedding $\hat{X}$. The perturbed trajectory generation module takes $\hat{X}$ as input to further produce a slightly perturbed trajectory $\hat{Tr}$. The collaborative adversarial learning module consists of a perturbation generator G, a discriminator D, and a TUL model. G takes X as input and generates perturbations to generate the perturbed trajectory embedding $\hat{X}$. To make $\hat{X}$ generated by G follows the same data distribution with X and effectively fools the TUL model, D and the TUL model collaboratively assist the training of G. Figure 1 (b) shows the idea of the perturbed trajectory generated by our model. When the original trajectory are input to the TUL model, the TUL model can correctly identify the user. However, when the lightly perturbed trajectory generated by our model is input into the TUL model, the TUL model cannot correctly identify the user. The core contributions of this paper can be summarized as follows:

- To our knowledge, we for the first time study the novel problem of resisting the TUL attack by balancing the data utility and privacy on trajectories. A collaborative adversarial learning model is proposed to effectively address it.
- Our collaborative adversarial learning model cleverly incorporates a TUL model to guide the generation of perturbations, with the aim of allowing our perturbations to perturb a few check-ins that are crucial for the TUL model to identify the user.
- We compare our proposal with baselines on three TUL tasks, and the results show that our model has better privacy protection performance when changing the same ratio of check-ins. The experiments conduct on a POI recommendation task further verifies the utility of our perturbed trajectories.

The remainder of the paper is organzied as follows: in the related work section, we introduce existing work for protecting location privacy and trajectory privacy, and discuss some existing TUL models. In the preliminaries section, we introduce some concepts employed in this paper and define the studied problem. In the method section, we detail the design of each module in our model. The experiment section shows the datasets and baselines used in our paper, and also provides extensive experimental results to validate the effectiveness of our proposed model.

2 Related Work

2.1 Privacy Protection on Locations

To reduce the risk of user location privacy leakage, numerous methods have been proposed, which can be mainly fell into two categories: obfuscation-based methods and anonymization-based methods.

Obfuscation-based methods protect privacy by obfuscating location information in data, but they reduce the data accuracy simultaneously. Therefore, these methods are typically used in scenarios where precise location information is not required, such as traffic flow analysis. The obfuscation-based methods include cloaking, dummy locations, differental privacy. [37] employed a cloaking area to hide the precise locations of users for privacy protection. Using large cloaking areas can cause a degradation in the precision of the user's location, thereby affecting the quality of the service provided by LBS server. Dummy locations based methods protect users' location privacy by spamming the adversary with fake locations. In order to prevent attackers with background knowledge from easily distinguishing dummy locations from real ones and causing fake locations to be ineffective, the dummy locations based methods must ensure that the fake locations are natural. Some methods have been proposed to ensure the naturalness of the generated dummy locations. The method proposed by [38] considered side information that may be exploited by attackers when selecting fake locations. They chose dummy locations based on the entropy metric, and enhanced the algorithm to make sure the selected dummy locations are spread far away. Differental privacy protects location privacy by adding controlled random noise to user's location. A famous location privacy protection method, known as Geo-Indistinguishability, is actually implemented based on differential privacy [15]. Geo-Indistinguishability adds random noise to the location data so that attackers cannot infer the exact location of individuals.

Anonymization-based methods aim to prevent attackers from inferring user's identity through the analysis of their location, which can be divided into two categories: k-anonymity and mix-zone. K-anonymity creates anonymity sets for user location information, whereby the location of users within the anonymity set cannot be distinguished from k-1 other users located in the same anonymity set. [40] proposed a k-anonymity based method, which focused on reducing the correlation between the user and the request. The method also maintained the service quality while protecting the privacy. Mix-Zone establishes a specific area, in which k users simultaneously change their pseudonyms to achieve K-anonymity. [41] aimed to protect location privacy in vehicular networks, and proposed a dynamic mix-zone method ,which dynamically created a mix-zone at the time the vehicle requests it.

However, methods for protecting location privacy typically protect the real-time location information of users when requesting location-based services, and may not effectively prevent attackers from obtaining users' identity by analyzing their complete trajectories. Different from these methods, our method focuses on preventing attackers from inferring user identity by analyzing the trajectories.

2.2 Privacy Protection on Trajectories

One popular method for privacy protection on trajectories is k-anonymity [12]. The basic idea is to group similar users together and make their trajectories become indistinguishable, thus protecting their privacy. However, when an attacker has some background knowledge, anonymization-based methods cannot effectively protect privacy [13]. The emergence of differential privacy overcomes this deficiency. Differential privacy adds some random perturbation into the user's original trajectories to obscure their personal sensitive information, thus protecting their privacy [15]. [19] proposed a differential privacy scheme base on the recurrent neural network, which can protect the privacy of real-time dynamic trajectories effectively. Nevertheless, differential privacy may decreases the accuracy and quality of data during the process of adding noise, which could affect the availability of data. There are also some methods that use deep learning to protect privacy of trajectories. [8] used generative adversarial networks to generate synthetic trajectories that are similar to the original trajectories in the temporal-spatial and semantic dimension. The synthetic trajectories generated by this method bring down the accuracy of a TUL model. However, the synthetic trajectories are totally different from the original ones, and thus largely affect the utility of the data in downstream tasks. Federated learning is a distributed machine learning technique that allows model training to be performed across multiple devices without centralizing the data set [42], and has become increasingly popular in recent years. In federated learning, each device holds a portion of the data and uses this data to locally train a machine learning model, which is then aggregated to create a new global model. By doing this, federated learning can train models while protecting user privacy and avoiding the risk of privacy leakage. The existing privacy protection methods cannot effectively resist attacks from attackers (e.g. a TUL model) with background knowledge, and usually add noises to all points on a trajectory for protecting privacy, which damages the utility of the data. This paper proposes a privacy protection method for TUL attacks, which aims to perturb a small number of points of a trajectory (for keeping utility) to make the TUL model incorrectly infer the user.

2.3 Trajectory-User Linking

TULER [1] employed the method in NLP to embed check-ins in trajectories into a low-dimensional space and adopted LSTM to model the mobility patterns of a user at the check-ins level. However, RNN-based methods suffer from data sparsity issues and are not capable of capturing the multi-periodic character of human mobility data. To alleviate the data sparsity problem, TULVAE [14] adopted a semi-supervised variational autoencoder to take advantage of numerous unlabeled data to improve the performance of the TUL task. Nevertheless, it still failed to fully utilize the features and take multi-periodic into account. DeepTUL [2] integrated multiple features and learned multi-periodic characters of human mobility from unlabeled historical data based on an attentive recurrent network model. TULSN [48] is a Siamese network-based model, which used a Siamese network to capture semantic information in a trajectory. It overcame the disadvantage of having to retrain the entire model once a new user is added. Only a small amount of geotagged data is required for the TUL

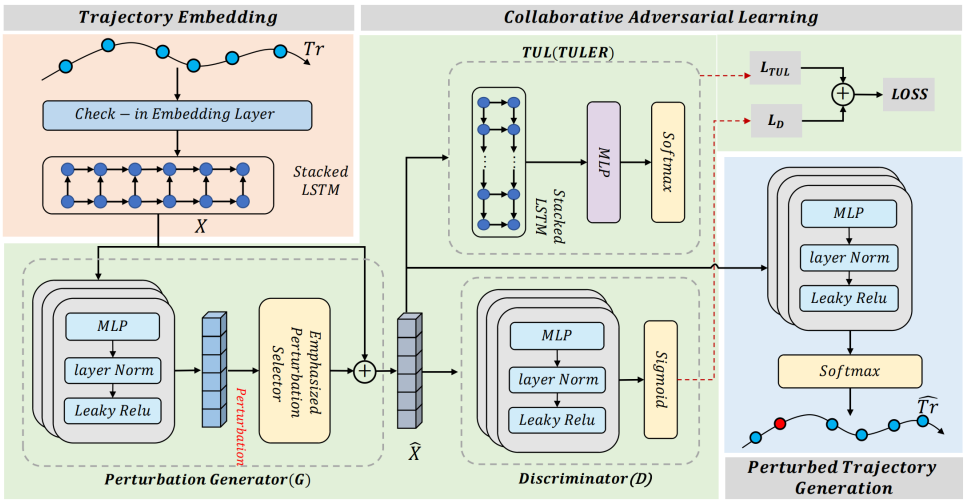


Fig. 2: The overall architecture of BPUCAL.

when a new user is added. MainTUL [16] utilized history trajectories by trajectory augmentation to relieve data sparsity problems. MainTUL used RNN to model the original trajectories and encoded the augmented trajectories by a temporal-aware transformer. Messages are transferred between the two trajectory encoders by mutual distillation. The recent TUL technique has greatly improved the utility of anonymous trajectories, but also raises more concerns about privacy leakage, it is necessary to propose new privacy protection methods to resist TUL attacks.

3 Preliminaries

In this section, we first give some terminology definitions, and then formulate the studied problem.

Definition 1 (Check-in). The check-in (i.e., point) data can be denoted as a triplet (u, t, p) representing user u checks in at POI p at time step t. POI stands for Point of Interest, for example, hospitals and schools can both be considered as a POI. p is also a triplet (id, c, l), where id is the identifier of a POI, $l = (latitude, longitude)$ represents the location of a POI and c is the identifier of the POI category.

Definition 2 (Trajectory). The trajectory is a sequence of check-ins, e.g., $Tr^u = \{(u, t_1, p_1), \cdots, (u, t_k, p_k)\}$, where k is the trajectory length.

Definition 3 (Perturbed Trajectory). Given the original trajectory Tr^u, we change a certain ratio of points of Tr^u to obtain a perturbed trajectory $\hat{Tr^u} : \{(u, t_1, \hat{p_1}), (u, t_2, \hat{p_2}), \cdots, (u, t_k, \hat{p_k})\}$.

Definition 4 (Trajectory-User Linking). Given a set of unlinked trajectories $\{Tr_1, Tr_2, \cdots, \mathrm{Tr}_n\}$ which are produced by a set of users $U = \{u_1, u_2, \cdots, \mathrm{u}_m\}$. Trajectory-User Linking (TUL) aims to find a mapping function f$(\cdot)$ between unlinked

trajectories and the corresponding users.

Based on the above definitions, we formally define the studied problem as follows. ***Problem Definition***. Given the users $U = \{u_1, u_2, \cdots, u_m\}$ and their trajectories $T = \{Tr^{u_1}, Tr^{u_2}, Tr^{u_3}, \cdots, Tr^{u_m}\}$, our goal is to learn a generation function Γ to generate the perturbed trajectories $\hat{T} = \{\hat{Tr^{u_1}}, \hat{Tr^{u_2}}, \hat{Tr^{u_3}}, \cdots, \hat{Tr^{u_m}}\}$ by adding a small number of perturbation. The perturbed trajectories $\hat{T} = \{\hat{Tr^{u_1}}, \hat{Tr^{u_2}}, \hat{Tr^{u_3}}, \cdots, \hat{Tr^{u_m}}\}$ can protect against the TUL attack and should also be as similar to the raw trajectories as possible to keep the data utility.

4 Methodology

As shown in Figure 2, BPUCAL consists of three modules, the trajectory embedding module, the collaborative adversarial learning module, and the perturbed trajectory generation module. The original trajectory Tr is first input into the trajectory embedding module to obtain a low-dimensional representation X. Then X is input into the collaborative adversarial learning module to generate the perturbed trajectory embedding $\hat{X}$. Feeding $\hat{X}$ into the perturbed trajectory generation module, we finally get the perturbed trajectory $\hat{Tr}$ with the minimal variation to the original trajectory Tr to fool the TUL model. Next, we will introduce the model in detail.

4.1 Trajectory Embedding

To reduce computation complexity and facilitate representation learning, we first conduct trajectory segmentation. Specifically, we divide the original check-in sequence of a user Tr^u by days into k successive sub-sequences $Tr^u = \{Tr_1^u, Tr_2^u, \cdots, Tr_k^u\}$. We use one-hot encoding to represent a trajectory consisting of a set of check-ins. The one-hot encoding is high-dimensional and is inefficient to interpret the relationship between check-ins. Thus given the original trajectory $Tr = \{p_1, p_2, p_3, \cdots, p_k\}$, we first map the one-hot encoding of Tr to a low-dimensional space as follows,

$$h_i = \tanh(\mathrm{W_p p_i} + \mathrm{b_p}) \tag{1}$$

where p_i is the one-hot encoding of the *i-th* POI, W_p and b_p are learnable parameters. In order to capture the temporal dependency between different check-ins of a trajectory, we input the encoded trajectories $H = \{h_1, h_2, \cdots h_k\}$ into stacked LSTM layers, which is formulated as follows,

$$\begin{aligned}
i_t &= \sigma(W_i h_t + U_i x_{t-1} + b_i), \\
f_t &= \sigma(W_f h_t + U_f x_{t-1} + b_f), \\
o_t &= \sigma(W_o h_t + U_o x_{t-1} + b_o), \\
\tilde{c_t} &= \tanh(\mathrm{W_c h_t} + \mathrm{U_c x_{t-1}} + \mathrm{b_c}), \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c_t}, \\
x_t &= o_t \odot \tanh(\mathrm{c_t}),
\end{aligned} \tag{2}$$

where i_t, f_t and o_t represent the input gate, forget gate, and output gate, respectively. W, U, b are the gate parameters. $\sigma(\cdot)$ and $tanh(\cdot)$ refer to the sigmoid function and hyperbolic tangent function. h_t is the embedding of the current check-in. x_t and x_{t-1} denote the current and last state embedding, respectively. $\odot$ is the entry-wise product. $X = \{x_1,x_2,\cdots,x_k\}$ is the final representation of the trajectory in the low-dimensional space.

4.2 Collaborative Adversarial Learning

Adversarial learning has been widely applied to various computer vision tasks, which aims to generate adversarial example to cheat the target model. Inspired by this, we propose a collaborative adversarial learning module to generate the perturbed trajectory to fool the TUL model. Adversarial learning generally adds imperceptible noise to each pixel when generating an adversarial example for the image. Unlike images, adding too much noise to each check-in of a trajectory could remarkably change the original trajectory, and thus causes severe data distortion issue. The perturbed trajectory generated in this way may severely hurt the utility of the original trajectory data. It motivates us to design a method for generating a perturbed trajectory that can effectively fool the TUL model while keeping most points on a trajectory invariant. Previous studies on computer vision [25] demonstrate that only partial pixels are critical for image classification. Inspired by this, we assume that only a small number of check-ins of a trajectory are critical for a TUL model to re-identify the user. Therefore, we need to identify such critical check-in points as our target for perturbation. To this end, we design a collaborative adversarial learning framework that consists of three modules, a perturbation generator G, a discriminator D, and a TUL model. The novelty of our collaborative adversarial learning model is that we incorporate a TUL model to guide the generation of perturbations cleverly, with the aim of allowing our perturbations to perturb a small number of check-ins that are crucial for the TUL model to identify the user. Next, we will introduce each module in detail.

Perturbation Generator G**.** The aim of G is to generate the perturbations for a trajectory. By adding the perturbations to the trajectory embedding $X = \{x_1, x_2, \cdots, x_k\}$, we obtain the perturbed trajectory embedding $\hat{X} = \{\hat{x_1}, \hat{x_2}, \cdots \hat{x_k}\}$. The trajectory embedding X is input into the perturbation generator to generate perturbation for each check-in point as follows,

$$\begin{aligned} R &= \text{LayerNorm}(\text{MLP}(\text{X})), \\ Q &= \text{LeakyRelu}(\text{R}), \end{aligned} \tag{3}$$

where MLP is multi-layer perceptions. LeakyReLU denotes the Leaky ReLU activation. $Q = \{q_1, q_2, q_3, \cdots, q_k\}$ denotes the perturbations generated for the trajectory embedding X and q_i is the perturbation generated for x_i.

To make the generator focus more on perturbing critical points which are important for TUL to infer the user, intuitively, we should not add perturbations to all the POIs with the equal probability. Thus we design the emphasized perturbation selector to attenuate the perturbations added to the unimportant POIs and emphasize on critical ones. The emphasized perturbation selector takes Q as input, it reserves the largest

perturbation on a trajectory and attenuates the remaining perturbations, which can be formulated as:

$$q_{max} = \max(\mathrm{Q}), \tag{4}$$

$$mask = \begin{cases} 1, & if \quad q_i = q_{max}, \\ \beta, & if \quad q_i \neq q_{max}, \end{cases} \tag{5}$$

$$\Delta = Q * mask, \tag{6}$$

where $\beta \in [0, 1]$ is the damping decrement, Δ is the ultimate emphasized perturbation. Adding the perturbation Δ with X, we obtain the perturbed trajectory representation in the low-dimensional space, which is formulated as

$$\hat{X} = X + \Delta. \tag{7}$$

Discriminator D. The goal of the discriminator D is to distinguish the trajectory embedding and the perturbed trajectory embedding. D drives G to generate perturbed trajectory embedding following the original data distribution, which can be expressed as:

$$\begin{aligned} I &= \mathrm{LayerNorm}(\mathrm{MLP}(\mathrm{X})), \\ D &= \sigma(\mathrm{LeakyRelu}(\mathrm{I})), \end{aligned} \tag{8}$$

Trajectory-User Linking Model We adopt the TULER proposed by [1] as our TUL model. Our goal is to employ TULER to help find the pivotal points which are important for TUL to infer users, and then we let the generator produce noises for these points, so that a small modification of the original trajectories can cause the TUL model misjudgement. Therefore, we need the TUL model to have strong judgement to motivate the generator to produce more effective noise. To achieve this, We use the trajectories as training data and employ user-trajectory pairs to train TULER in advance. After training, we fix the TULER model and use it to guide G for perturbation generation. The TULER model aims to drive G to generate perturbations that can mislead itself. Inputting $\hat{X}$ to the TULER, $\hat{X}$ generated by G tries to mislead TULER, TULER feeds back the effect to G to motivate G to generate a more effective noise.

4.3 Perturbed Trajectory Generation

In previous modules, we map the trajectory into a low-dimensional space and generate perturbations for a trajectory embedding to get the perturbed trajectory embedding. In this module, we generate the perturbed trajectory from the perturbed trajectory embedding, which is represented by the following formula:

$$\hat{Tr} = \mathrm{softmax}(\mathrm{MLP}(\hat{\mathrm{X}})), \tag{9}$$

where $\hat{Tr} = \{\hat{p_1}, \hat{p_2}, \hat{p_3}, \cdots, \hat{p_k}\}$. During the perturbation generation stage, the more perturbation generated for a point, the harder it is to restore to the original point.

4.4 Loss Function

The training process contains the pretraining and model training steps. Next we introduce the two steps in detail.

Pretraining We first pretrain the trajectory embedding module TE and the perturbed trajectory generation module PTG. The two modules are then fixed during the entire model training. The trajectory embedding module TE aims to map the check-in points in the high-dimensional space to a low-dimensional space. The perturbed trajectory generation module denoted as PTG generates the perturbed trajectory from the perturbed trajectory embedding. We pretrain the two modules by minimizing the reconstruction error as follows.

$$\begin{aligned} X &= TE(Tr), \\ \mathcal{L}_{\mathcal{R}} &= -\frac{1}{k}\sum_{i=1}^{k} PTG(x_i) log p_i \end{aligned} \tag{10}$$

where Tr ={p_1, p_2, p_3, $\cdots$,p_k}, and p_i is the one-hot encoding of the i-th check-in. X = {x_1,x_2,$\cdots \mathrm{x}_k$}, where x_i is the dense representation of the i-th check-in.

Model Training. To train the perturbation generator G that generates perturbed trajectory embedding which can both mislead the TUL model and the discriminator, we design the loss function to drive the training process. In general, we optimize two networks, i.e., the perturbation generator network G and the discriminator network D. We design loss functions for these two networks separately. For G, the loss function is as follows

$$\mathcal{L}_G = \mathcal{L}_{TUL} + \mathcal{L}_{dc} \tag{11}$$

$$\mathcal{L}_{TUL} = -E_{\hat{X}}[-TULER(\hat{X}) log u)] \tag{12}$$

$$\mathcal{L}_{dc} = E_X[log(1 - D(\hat{X})] \tag{13}$$

where $E_X[\cdot]$ represents the expectation operator over the training samples, $\hat{X}$ is the perturbed sample of the original sample X, and u denotes the user of X. $\mathcal{L}_{\mathcal{TUL}}$ drives G to generate $\hat{X}$ to fool the TUL model. $\mathcal{L}_{dc}$ propels G to produce perturbed trajectory embedding that follows the original data distribution. D $(\hat{X}) \in [0, 1]$ denotes the probability that the perturbed sample is classified as geniue by the discriminator D. For D, the loss function is defined as follows.

$$\mathcal{L}_D = \frac{1}{2}(E_X[log(1 - D(X))] + E_X[log(D(\hat{X}))]). \tag{14}$$

5 Experiments

5.1 Datasets and Baselines

We use two widely used check-in mobility datasets [34, 35] collected from two popular location-based social network platforms, i.e., Foursquare[1] and Weeplaces[2] for evaluation. For the two datasets, we choose top 531 and 420 users with the most check-ins

[1] http://sites.google.com/site/yangdingqi/home/foursquare-dataset
[2] http://www.yongliu.org/datasets.html

for evaluation respectively. The Foursquare dataset contains 50,414 trajectories, 9,458 POIs, and 248 categories of POI. Weeplaces has 101,771 trajectories, 22,140 POIs and 660 categories of POI. In the experiment, we use the 60% of the trajectories of each user for training, 20% for testing and the remaining 20% data as the validation set.

As this paper studies a new problem, we do not find suitable baselines that are directly comparable. LSTM-TrajGAN is a trajectory generation model proposed in 2021, which generates a totally different trajectory from the raw one for privacy protection, but the data utility is not considered. As our method and LSTM-TrajGAN both focus on resisting against TUL attacks for privacy protection, we compare our proposed method with it. To make a more comprehensive evaluation of our model, we design two baselines for comparison. AMF is a heuristic algorithm that identifies and replaces the most frequently visited POIs by each user, because such POIs may contain the most informative features for user identification. The Random baseline randomly selects some check-ins and replaces them. The following is a more detailed introduction for our baselines.

- **Originals** This method uses the raw trajectory data without any processing.
- **Random** This method randomly changes a portion of the check-ins of the original trajectories. We randomly select a POI from the entire POI collection to replace the selected one as the perturbation.
- **AMF** This method perturbs the check-ins that are visited with the highest frequency for each user. As these points may appear on multiple trajectories of the user, it may be recognized as an quasi-identifier by a TUL model for inferring the user.
- **LSTM-TrajGAN** It is a recent generative adversarial model proposed by [8] to generate synthetic trajectoris which reserves spatial, temporal, and semantic characteristics of the original trajectories . However, it generates a totally different trajectory from the raw one for privacy protection, and thus the data utility is not considered.

5.2 Evaluation Metrics and Parameter Settings

To assess the utility and privacy protection performance of the perturbed trajectories, we propose the following two indicators, similarity that represents the reserve ratio of the original data, and the prediction performance of a TUL model. A better prediciton performance of the TUL model means a higher possibility of user identification and the worse privacy protection performance. We validate the privacy protection performance of our perturbed trajectories on three TUL tasks, MainTUL [16], TULER [1], and S2TUL [43]. We use the Acc@k, Macro-Precision, Macro-Recall, and Macro-F1 to evaluate the model performance. ACC@K is to evaluate the accuracy of the TUL model, which can be represented as follows:

$$ACC@K = \frac{correctly\ linked\ trajectories@K}{the\ number\ of\ trajectories} \tag{15}$$

Macro-F1 is the harmonic mean of the macro-P and macro-R, averaged across all classes:

$$macro - F1 = 2 \times \frac{macro - P \times macro - R}{macro - P + macro - R} \tag{16}$$

We also test the performance of the perturbed trajectories on a POI recommendation task to verify the utility. We adopt the top-k recall rates, Recall@5 and Recall@10, to evaluate the recommendation performance. Recall@k counts the rate of the true positive samples in all positive samples. We employ the STAN model proposed by [36] as our evaluation model. For our model, we set check-in embedding dimension d to 128, and β to 0.3. To generate perturbed trajectories, we run our method and all the baselines 10 times. For privacy evaluation, we run each TUL task 10 times. For data utility evaluation, we run STAN 3 times. We report the average result for all the methods.

5.3 Overall Performance Comparison

The comparison results of BPUCAL and baselines on three TUL tasks are presented in Tables 1, 2, and 3, respectively. The performance of perturbed trajectories generated by our model and baselines on POI recommendation tasks is shown in Table 4. From these tables, one can have the following conclusions.

- On both datasets it shows that the lower similarity to the original trajectories leads to worse performance of STAN. It verifies our idea of maintaining data utility by making a minimum variation to the original trajectory.
- The excellent performance of three TUL models(i.e. MainTUL, TULER, S2TUL) for inferring users in the unprocessed data (Originals), shows that the TUL model is indeed a powerful attacker, and confirms what we discussed before that publishing raw trajectories directly leads to a high risk of privacy leakage. Compared to the Originals baseline, our method BPUCAL makes the accuracy of three TUL model drop significantly by sacrificing tiny data utility.
- Compared to Random, under the same similarity of 0.75, Random only decreases the accuracy of MainTUL by about 0.03 on two datasets, while BPUCAL brings MainTUL down by about 0.3. This means that simply modifying a certain percentage of points will not greatly reduce the performance of a TUL model. We should modify the points that are more pivotal to a TUL model. It is difficult to find those points by Random. Our BPUCAL efficiently finds these key points.
- Trajectory points that each user visits frequently may appear on multiple trajectories of a user. Therefore, it may be a key feature for TUL model to infer the identity of a user. One can also observe from the experimental results that AMF which modifies the frequently visited POIs makes the TUL model drop significantly compared to Random. Our BPUCAL performs better than AMF on three TUL tasks. It shows that BPUCAL can identify more important points for the TUL model compared to AMF.
- LSTM-TrajGAN changes almost all points of the original trajectories and generates totally different trajectories. Although it makes the accuracy of TUL models drop significantly , the data utility has been greatly damages for downstream tasks, which

Table 1: Performance comparison with baselines on MainTUL. Macro-P/R: Macro-Precision/Recall.

Dataset	Similarity	Methods	MainTUL				
			Acc@1	Acc@5	Macro-P	Macro-R	Macro-F1
Foursquare	1.00	Originals	0.7252	0.8433	0.7316	0.7024	0.7167
	0.90	AMF-10	0.6285	0.7892	**0.6972**	0.6342	0.6691
		Random-10	0.7241	0.8432	0.7305	0.7009	0.7154
		BPUCAL-10	**0.6120**	**0.7683**	0.7166	**0.5783**	**0.6411**
	0.85	AMF-15	0.5811	0.7632	**0.6772**	0.6325	0.6541
		Random-15	0.7225	0.8414	0.7286	0.6999	0.7139
		BPUCAL-15	**0.5649**	**0.7307**	0.7171	**0.5412**	**0.6168**
	0.80	AMF-20	0.5382	0.7336	**0.6538**	0.6032	0.6274
		Random-20	0.7135	0.8361	0.7208	0.6893	0.7047
		BPUCAL-20	**0.5018**	**0.6916**	0.7033	**0.4663**	**0.5608**
	0.75	AMF-25	0.4567	0.6693	**0.5786**	0.4877	0.5292
		Random-25	0.7092	0.8342	0.7190	0.6855	0.7018
		BPUCAL-25	**0.4313**	**0.6545**	0.7090	**0.3999**	**0.5114**
	0.70	AMF-30	0.4093	0.6328	**0.5542**	0.4536	0.4988
		Random-30	0.6909	0.8207	0.6995	0.6661	0.6824
		BPUCAL-30	**0.3697**	**0.6160**	0.6901	**0.3427**	**0.4581**
	0.08	LSTM-TrajGAN	0.0687	0.1042	0.0525	0.0737	0.0613
Weeplaces	1.00	Originals	0.7796	0.8903	0.7614	0.7319	0.7464
	0.90	AMF-10	0.6921	0.8132	0.7254	0.6544	0.6881
		Random-10	0.7777	0.8896	0.7601	0.7303	0.7449
		BPUCAL-10	**0.6748**	**0.7956**	**0.6949**	**0.6098**	**0.6495**
	0.85	AMF-15	0.6528	0.7796	0.6877	0.6284	0.6567
		Random-15	0.7739	0.8873	0.7574	0.7264	0.7416
		BPUCAL-15	**0.6311**	**0.7538**	**0.6803**	**0.5321**	**0.5972**
	0.80	AMF-20	0.6336	0.7538	**0.6377**	0.6098	0.6234
		Random-20	0.7638	0.8801	0.7475	0.7168	0.7318
		BPUCAL-20	**0.5864**	**0.7222**	0.6549	**0.5231**	**0.5816**
	0.75	AMF-25	0.5436	0.6623	**0.5622**	0.5145	0.5372
		Random-25	0.7549	0.8723	0.7402	0.7086	0.7239
		BPUCAL-25	**0.4671**	**0.5993**	0.6599	**0.4028**	**0.5002**
	0.70	AMF-30	0.5071	0.6262	**0.5581**	0.4637	0.5065
		Random-30	0.7466	0.8681	0.7334	0.7001	0.7164
		BPUCAL-30	**0.3911**	**0.5782**	0.7074	**0.3442**	**0.4631**
	0	LSTM-TrajGAN	0.0129	0.0422	0.0031	0.0152	0.0052

can be confirmed by the prediction results of STAN. Obviously, this method which sacrifices much data utility for privacy protection is inefficient and unadvisable.

- Overall, our method achieves the best performance on two public datasets for balancing data utility and privacy.

Table 2: Performance comparison with baselines on TULER. Macro-P/R: Macro-Precision/Recall.

Dataset	Similarity	Methods	TULER				
			Acc@1	Acc@5	Macro-P	Macro-R	Macro-F1
Foursquare	1.00	Originals	0.6582	0.7892	0.5671	0.5671	0.5671
	0.90	AMF-10	0.5724	0.7432	0.5231	0.5416	0.5321
		Random-10	0.6555	0.7855	0.5632	0.5636	0.5634
		BPUCAL-10	**0.5524**	**0.7127**	**0.4931**	**0.4761**	**0.4844**
	0.85	AMF-15	0.5112	0.7177	0.4877	0.4635	0.4752
		Random-15	0.6534	0.7847	0.5607	0.5621	0.5614
		BPUCAL-15	**0.4937**	**0.6828**	**0.4731**	**0.4474**	**0.4599**
	0.80	AMF-20	0.4483	0.6324	0.4312	0.4294	0.4302
		Random-20	0.6311	0.7726	0.5392	0.5368	0.5381
		BPUCAL-20	**0.4291**	**0.6199**	**0.4241**	**0.3854**	**0.4037**
	0.75	AMF-25	0.3684	0.5572	0.3988	0.3212	0.3558
		Random-25	0.6287	0.7708	0.5376	0.5366	0.5372
		BPUCAL-25	**0.3511**	**0.5332**	**0.3677**	**0.3191**	**0.3416**
	0.70	AMF-30	0.3102	0.5005	0.3205	0.2945	0.3069
		Random-30	0.6037	0.7501	0.5133	0.5113	0.5123
		BPUCAL-30	**0.2898**	**0.4724**	**0.3174**	**0.2764**	**0.2954**
	0.08	LSTM-TrajGAN	0.0512	0.0518	0.0231	0.0312	0.0265
Weeplaces	1.00	Originals	0.6968	0.8309	0.5974	0.5968	0.5971
	0.90	AMF-10	0.6012	0.7623	0.4922	0.5014	0.4967
		Random-10	0.6899	0.8276	0.5879	0.5874	0.5876
		BPUCAL-10	**0.5899**	**0.7311**	**0.4852**	**0.4882**	**0.4867**
	0.85	AMF-15	0.5444	0.6996	0.4672	0.4518	0.4593
		Random-15	0.6805	0.8199	0.5791	0.5778	0.5784
		BPUCAL-15	**0.5296**	**0.6711**	**0.4523**	**0.4467**	**0.4495**
	0.80	AMF-20	0.5219	0.6648	0.4032	0.4101	0.4066
		Random-20	0.6607	0.8058	0.5574	0.5571	0.5572
		BPUCAL-20	**0.4935**	**0.6511**	**0.4011**	**0.4035**	**0.4023**
	0.75	AMF-25	0.4812	0.6399	0.3868	0.3923	0.3896
		Random-25	0.6367	0.7903	0.5325	0.5329	0.5327
		BPUCAL-25	**0.3641**	**0.5173**	**0.3383**	**0.3098**	**0.3234**
	0.70	AMF-30	0.4528	0.6129	0.3629	0.3678	0.3653
		Random-30	0.6282	0.7838	0.5254	0.5243	0.5249
		BPUCAL-30	**0.3334**	**0.5022**	**0.3209**	**0.2901**	**0.3047**
	0	LSTM-TrajGAN	0.0112	0.0216	0.0145	0.0132	0.0138

5.4 Ablation Study

In this section, we compare our model with three elaborate variants. We conduct experiments on all two datasets. The three variants are as follows:

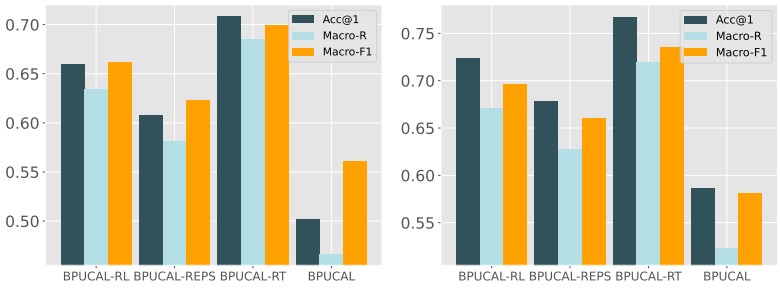
Table 3: Performance comparison with baselines on S2TUL. Macro-P/R: Macro-Precision/Recall.

Dataset	Similarity	Methods	S2TUL				
			Acc@1	Acc@5	Macro-P	Macro-R	Macro-F1
Foursquare	1.00	Originals	0.6066	0.7648	0.6346	0.5803	0.6062
	0.90	AMF-10	0.5291	0.6673	0.6428	0.4672	0.5411
		Random-10	0.6061	0.7651	0.6328	0.5799	0.6051
		BPUCAL-10	**0.4931**	**0.6597**	**0.5803**	**0.4592**	**0.5126**
	0.85	AMF-15	0.4512	0.6228	0.5789	0.4476	0.5048
		Random-15	0.6001	0.7623	0.6295	0.5738	0.6003
		BPUCAL-15	**0.4347**	**0.6189**	**0.5662**	**0.4241**	**0.4899**
	0.80	AMF-20	0.3764	0.5534	0.5612	0.3528	0.4332
		Random-20	0.5913	0.7541	0.6213	0.5646	0.5915
		BPUCAL-20	**0.3556**	**0.5276**	**0.5549**	**0.3452**	**0.4256**
	0.75	AMF-25	0.3231	0.4932	0.5019	0.3123	0.3851
		Random-25	0.5911	0.7522	0.6209	0.5621	0.5901
		BPUCAL-25	**0.2906**	**0.4453**	**0.4892**	**0.2741**	**0.3513**
	0.70	AMF-30	0.2312	0.4012	0.4537	0.2464	0.3193
		Random-30	0.5729	0.7386	0.5961	0.5451	0.5694
		BPUCAL-30	**0.2291**	**0.3675**	**0.4409**	**0.2092**	**0.2837**
	0.08	LSTM-TrajGAN	0.0921	0.1113	0.0734	0.0664	0.0697
Weeplaces	1.00	Originals	0.5804	0.6957	0.5663	0.5581	0.5621
	0.90	AMF-10	0.4921	0.6363	0.5227	0.4776	0.4913
		Random-10	0.5785	0.6951	0.5657	0.5562	0.5609
		BPUCAL-10	**0.4587**	**0.5974**	**0.5111**	**0.4118**	**0.4561**
	0.85	AMF-15	0.4432	0.5872	0.4912	0.3718	0.4232
		Random-15	0.5765	0.6926	0.5641	0.5539	0.5589
		BPUCAL-15	**0.3595**	**0.4834**	**0.4858**	**0.3166**	**0.3833**
	0.80	AMF-20	0.3928	0.5463	0.4836	0.3484	0.4051
		Random-20	0.5688	0.6871	0.5558	0.5471	0.5514
		BPUCAL-20	**0.3453**	**0.4793**	**0.4782**	**0.3091**	**0.3754**
	0.75	AMF-25	0.3564	0.4827	0.4512	0.3112	0.3683
		Random-25	0.5612	0.6775	0.5547	0.5411	0.5478
		BPUCAL-25	**0.3123**	**0.3678**	**4412**	**0.2893**	**0.3494**
	0.70	AMF-30	0.3121	0.3924	**0.4239**	0.2725	0.3317
		Random-30	0.5598	0.6712	0.5497	0.5385	0.5441
		BPUCAL-30	**0.2824**	**0.3218**	**0.3729**	**0.2642**	**0.3092**
	0	LSTM-TrajGAN	0.0721	0.1032	0.0937	0.0829	0.0879

- BPUCAL-RL: Replace the MLP layer in the perturbation generator with LSTM. This variant is used to verify the effectiveness of the MLP layer in the perturbation generator.
- BPUCAL-REPS: Remove the emphasized perturbation selector in the perturbation generator. This variant is used to verify the effectiveness of the perturbation generator.

Table 4: Performance comparison with baselines on STAN.

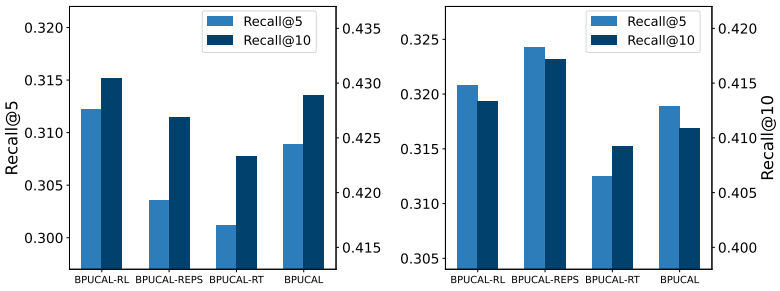
Similarity	Methods	Datasets			
		Foursquare		Weeplace	
		Recall@5	Recall@10	Recall@5	Recall@10
1.00	Originals	0.3398	0.4435	0.3618	0.4553
0.90	AMF-10	0.3305	0.4385	0.3406	0.4324
	Random-10	0.3276	0.4367	**0.3503**	**0.4412**
	BPUCAL-10	**0.3392**	**0.4412**	0.3473	0.4384
0.85	AMF-15	0.3172	0.4336	0.3256	0.4175
	Random-15	0.3105	0.4322	0.3302	0.4257
	BPUCAL-15	**0.3249**	**0.4352**	**0.3313**	**0.4263**
0.80	AMF-20	0.3024	0.4193	0.3115	0.4073
	Random-20	**0.3089**	**0.4286**	**0.3213**	**0.4123**
	BPUCAL-20	0.3045	0.4256	0.3189	0.4109
0.75	AMF-25	0.2885	0.4095	**0.3027**	**0.4061**
	Random-25	**0.2914**	**0.4103**	0.2851	0.3922
	BPUCAL-25	0.2879	0.4089	0.2953	0.4017
0.70	AMF-30	**0.2702**	**0.3912**	0.2616	0.3804
	Random-30	0.2635	0.3948	**0.2736**	**0.3839**
	BPUCAL-30	0.2612	0.3883	0.2672	0.3823
0.08	LSTM-TrajGAN	0.0145	0.0237	-	-
0	LSTM-TrajGAN	-	-	0.0057	0.0082



(a) Foursquare (b) Weeplaces

Fig. 3: Performance comparison of data generated by four variants on MainTUL.

- BPUCAL-RT: Remove the TUL module in our model. This variant is used to verify the effectiveness of the TUL module.



(a) Foursqure (b) Weeplaces

Fig. 4: Performance comparison of data generated by four variants on STAN.

The results are presented in Figure 3 and Figure 4, where Figure 3 shows the performance of the perturbed trajectories generated by the four models on MainTUL, and Figure 4 shows the performance on STAN. Our experiment is based on the perturbed trajectories which have a similarity of 0.8 with the original trajectories. Since the perturbed trajectories produced by the four models, all have a similarity of 0.8 to the original trajectories, their performance on STAN differs very little.

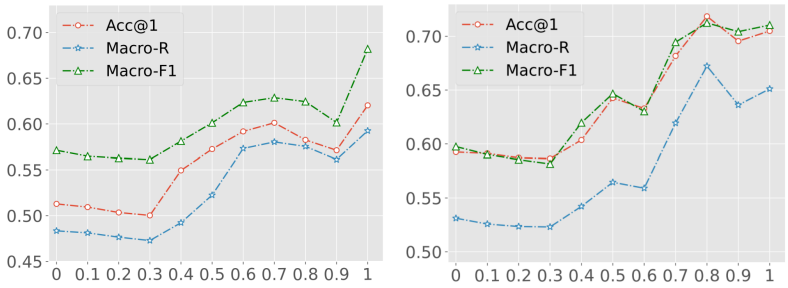
Comparing the performance of the perturbed trajectories generated by four models on MainTUL, one can see that BPUCAL-RT has the worst performance. The reason may be that with the assistance of TUL, the emphasized perturbation generation module can learn which points are more important for TUL, and generate perturbations for disturbing these points. Without the help of the TUL model, it generates perturbation for each point randomly, and the performance is similar to Random.

The results of BPUCAL-RL show that replacing the MLP layer with LSTM in emphasized perturbation generation module results in worse performance. It may be that each time unit of LSTM shares weights, it is more difficult to generate perturbation for each point based on their importance to TUL.

BPUCAL-REPS shows the effect of the emphasized perturbation selector. Compared to BPUCAL-REPS, BPUCAL can make the accuracy of the MainTUL drop by more than 0.1. The emphasized perturbation selector keeps the largest perturbation on a trajectory and attenuates the rest. It can drive the perturbation generation module to focus more on disturbing the points which are important for TUL.

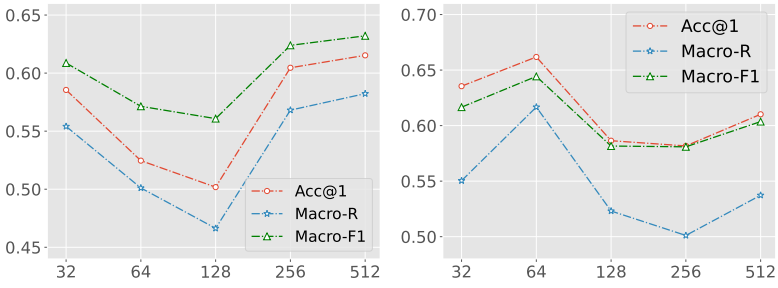
5.5 Parameter Study

In this section, we evaluate the sensitivity of our BPUCAL with different settings of damping decrement β and the check-in embedding d on two datasets. We compare the privacy protection performance of the perturbed trajectories generated by BPUCAL under different parameter settings on MainTUL. The results are shown in Figure 5 and Figure 6. We compare the performance of BPUCAL with parameter β ranging from 0.3 to 1. It shows that although the curve fluctuates, it is generally in an upward trend. This may be because when β is smaller, the perturbation generator G can focus more on perturbing the points which are pivotal for TUL during the confrontation



(a) Foursquare (b) Weeplaces

Fig. 5: Analysis of damping decrement β.



(a) Foursqure (b) Weeplaces

Fig. 6: Analysis of check-in embedding dimension d.

between G and TUL. We then study the impact of check-in embedding dimension d and vary the value in 32, 64, 128, 256, and 512 these five empirical values. One can notice that the privacy protection performance is sensitive to the value of d. But one can also note that the privacy protection performance could gain improvement when d is moderate, so it is vital to choose an appropriate d value.

6 Conclusion

In this paper, we propose a collaborative adversarial learning-based model BPUCAL. We aim to make a minimum variation of the original trajectories to mislead the TUL model for balancing data utility and privacy protection. To achieve this goal, we elaborately design a collaborative adversarial learning module, which generates perturbations to disturb a few points on a trajectory that is pivotal for TUL. Extensive experiments are conducted over two real-world datasets. The results show the effectiveness of our proposal in balancing data privacy and utility for trajectories.

References

[1] Gao Q, Zhou F, Zhang K, et al (2017) Identifying Human Mobility via Trajectory Embeddings. In: IJCAI, pp 1689-1695.

[2] Miao C, Wang J, Yu H, et al (2020) Trajectory-user linking with attentive recurrent network. In: Proceedings of the 19th international conference on autonomous agents and multiagent systems, pp 878-886.

[3] Zhao P, Luo A, Liu Y, et al (2020) Where to go next: A spatio-temporal gated network for next poi recommendation. In: IEEE Transactions on Knowledge and Data Engineering, pp 2512-2524.

[4] Vlahogianni E I, Karlaftis M G, Golias J C (2014) Short-term traffic forecasting: Where we are and where we're going. In: Transportation Research Part C: Emerging Technologies, pp 3-19.

[5] Wu H, Xue M, Cao J, et al (2016) Fuzzy trajectory linking. In: IEEE 32nd International Conference on Data Engineering (ICDE), pp 859-870.

[6] Chow C Y, Mokbel M F (2011) Trajectory privacy in location-based services and data publication. In: ACM Sigkdd Explorations Newsletter, pp 19-29.

[7] Liu X, Chen H, Andris C (2018) trajGANs: Using generative adversarial networks for geo-privacy protection of trajectory data (Vision paper). In: Location privacy and security workshop, pp 1-7.

[8] Rao J, Gao S, Kang Y, et al (2020) LSTM-TrajGAN: A deep learning approach to trajectory privacy protection. In: Proceedings of the 11th International Conference on Geographic Information Science.

[9] Chow C Y, Mokbel M F, Liu X (2006) A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In: Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems, pp 171-178.

[10] A. Almusaylim Z, Jhanjhi N Z (2020) Comprehensive review: Privacy protection of user in location-aware services of mobile cloud computing. In: Wireless Personal Communications, pp 541-564.

[11] Cavanillas J M, Curry E, Wahlster W (2016) The big data value opportunity. In: New horizons for a data-driven economy: A roadmap for usage and exploitation of big data in Europe, pp 3-11.

[12] Nergiz M E, Atzori M, Saygin Y (2008) Towards trajectory anonymization: a generalization-based approach. In: Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS, pp 52-61.

[13] Li M, Zhu L, Zhang Z, et al (2017) Achieving differential privacy of trajectory data publishing in participatory sensing. In: Information Sciences, pp 1-13.

[14] Zhou F, Gao Q, Trajcevski G, et al (2018) Trajectory-User Linking via Variational AutoEncoder. In: IJCAI, pp 3212-3218.

[15] Andrés M E, Bordenabe N E, Chatzikokolakis K, et al (2013) Geo-indistinguishability: Differential privacy for location-based systems. In : Proceedings of the 2013 ACM SIGSAC conference on Computer communications security, pp 901-914.

[16] Chen W, Li S, Huang C, et al (2022) Mutual Distillation Learning Network for Trajectory-User Linking. In: IJCAI.

[17] Garfinkel S (2015) De-identification of Personal Information. In: US Department of Commerce, National Institute of Standards and Technology.

[18] De Montjoye Y A, Hidalgo C A, Verleysen M, et al (2013) Unique in the crowd: The privacy bounds of human mobility. In: Scientific reports, pp 1-5.

[19] Chen S, Fu A, Shen J, et al (2020) RNN-DP: A new differential privacy scheme base on Recurrent Neural Network for Dynamic trajectory privacy protection. In: Journal of Network and Computer Applications.

[20] Tu Z, Xu F, Li Y, et al (2018) A new privacy breach: User trajectory recovery from aggregated mobility data. In: IEEE/ACM Transactions on Networking, pp 1446-1459.

[21] Qian T, Liu B, Nguyen Q V H, et al (2019) Spatiotemporal representation learning for translation-based POI recommendation. In: ACM Transactions on Information Systems (TOIS), pp 1-24.

[22] Li Y, Yu R, Shahabi C, et al (2017) Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: International Conference on Learning Representations (ICLR).

[23] Fiore M, Katsikouli P, Zavou E, et al (2020) Privacy in trajectory micro-data publishing: a survey. In: Transactions on Data Privacy, pp 91-149.

[24] Zhao Z, Dua D, Singh S (2017) Generating natural adversarial examples. In: International Conference on Learning Representations (ICLR).

[25] Vermeire T, Brughmans D, Goethals S, et al (2022) Explainable image classification with evidence counterfactual. In: Pattern Analysis and Applications, pp 315-335.

[26] Graves A, Graves A (2012) Long short-term memory. In: Supervised sequence labelling with recurrent neural networks, pp 37-45.

[27] Maouche M, Ben Mokhtar S, Bouchenak S (2018) Hmc: Robust privacy protection of mobility data against multiple re-identification attacks. In: Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, pp 1-25.

[28] Gao S, Ma J, Shi W, et al (2013) TrPF: A trajectory privacy-preserving framework for participatory sensing. In: IEEE Transactions on Information Forensics and Security, pp 874-887.

[29] Wang Y, Xu D, He X, et al (2012) L2P2: Location-aware location privacy protection for location-based services. In: Proceedings IEEE INFOCOM, pp 1996-2004.

[30] Jiang J, Han G, Wang H, et al (2019) A survey on location privacy protection in wireless sensor networks. In: Journal of Network and Computer Applications, pp 93-114.

[31] Yin C, Xi J, Sun R, et al (2017) Location privacy protection based on differential privacy strategy for big data in industrial internet of things. In: IEEE Transactions on Industrial Informatics, pp 3628-3636.

[32] Li M, Zhu L, Zhang Z, et al (2017) Achieving differential privacy of trajectory data publishing in participatory sensing. In: Information Sciences, pp 1-13.

[33] Zhao X, Dong Y, Pi D (2019) Novel trajectory data publishing method under differential privacy. In: Expert Systems with Applications.

[34] Yang D, Zhang D, Zheng V W, et al (2014) Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. In: IEEE Transactions on Systems, Man, and Cybernetics: Systems, pp 129-142.

[35] Liu Y, Wei W, Sun A, et al (2014) Exploiting geographical neighborhood characteristics for location recommendation. In: Proceedings of the 23rd ACM international conference on conference on information and knowledge management, pp 739-748.

[36] Luo Y, Liu Q, Liu Z (2021) Stan: Spatio-temporal attention network for next location recommendation. In: Proceedings of the Web Conference , pp 2177-2185.

[37] Ngo H, Kim J (2015) Location privacy via differential private perturbation of cloaking area. In: 2015 IEEE 28th computer security foundations symposium, pp 63-74.

[38] Niu B, Li Q, Zhu X, et al (2014) Achieving k-anonymity in privacy-aware location-based services. In: IEEE INFOCOM 2014-IEEE conference on computer communications, pp 754-762.

[39] Dobson J E, Fisher P F (2003) Geoslavery. In: IEEE Technology and Society Magazine, pp 47-52.

[40] Xing L, Jia X, Gao J, et al (2021) A location privacy protection algorithm based on double k-anonymity in the social internet of vehicles. In: IEEE Communications Letters, pp 3199-3203.

[41] Ying B, Makrakis D, Mouftah H T (2013) Dynamic mix-zone for location privacy in vehicular networks. In: IEEE Communications Letters, pp 1524-1527.

[42] Ma C, Li J, Ding M, et al (2020) On safeguarding privacy and security in the framework of federated learning. In: IEEE network, pp 242-248.

[43] Deng L, Sun H, Zhao Y, et al (2023) S2TUL: A Semi-Supervised Framework for Trajectory-User Linking. In: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, pp 375-383.

[44] Jiang R, Song X, Huang D, et al (2019) Deepurbanevent: A system for predicting citywide crowd dynamics at big events. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery data mining, pp 2114-2122.

[45] Jiang R, Yin D, Wang Z, et al (2021) Dl-traff: Survey and benchmark of deep learning models for urban traffic prediction. In: Proceedings of the 30th ACM international conference on information knowledge management, pp 4515-4525.

[46] Jiang R, Song X, Fan Z, et al (2018) Deepurbanmomentum: An online deep-learning system for short-term urban mobility prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence.

[47] Fan Z, Song X, Jiang R, et al (2019) Decentralized attention-based personalized human mobility prediction. In: 2019 Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, pp 1-26.

[48] Yu Y, Tang H, Wang F, et al (2020) TULSN: siamese network for trajectory-user linking. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp 1-8.