Adversarial Reconstruction of Trajectories: Privacy Risks and Attack Models in Trajectory Embedding

Haochen Han
haochen_neyc@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Shuaiyu Yang
yangshuaiyu6791@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Jiaxin Ding *
jiaxinding@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Luoyi Fu
yiluofu@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Xinbing Wang
xwang8@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Chenghu Zhou
zhouch@lreis.ac.cn
Chinese Academy of Sciences
Beijing, China

Abstract

Human trajectories, representing sequences of location points over time, are extensively collected and analyzed for various real-world applications such as urban planning, transportation management, and personalized location-based services. Trajectory embedding transforms raw trajectories into vector representations, capturing the underlying patterns and structures in the data. However, the abstraction provided by vector representations introduces significant security and privacy risks. These embeddings, often shared between entities or organizations, can be exploited by adversaries to reconstruct original trajectories, thereby compromising individual privacy. In this paper, we investigate the privacy issues of trajectory embeddings from an adversary's perspective. We propose two types of attacks to reconstruct original trajectories using road network information, addressing scenarios where the adversary has varying degrees of access to the black-box representation model. The first attack assumes unrestricted access to the model, allowing the adversary to construct a large-scale dataset and train a neural network to predict the road sequence of the trajectories. The second attack considers limited access, where the adversary computes distance coordinates between selected trajectory landmarks and road segments to infer different parts of the trajectory. Our experiments on a real-world dataset demonstrate that the reconstructed trajectories outperform baseline methods, achieving substantially lower reconstruction errors and more accurate alignment with the original trajectories, highlighting the significant vulnerability of trajectory embeddings to privacy breaches. These findings underscore the need for robust privacy-preserving mechanisms in spatio-temporal data analysis.

CCS Concepts

• **Security and privacy**; • **Computing methodologies** → **Artificial intelligence**;

Keywords

location privacy, trajectory embedding, reconstruction attack

ACM Reference Format:
Haochen Han, Shuaiyu Yang, Jiaxin Ding, Luoyi Fu, Xinbing Wang, and Chenghu Zhou. 2024. Adversarial Reconstruction of Trajectories: Privacy Risks and Attack Models in Trajectory Embedding. In *The 32nd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '24), October 29-November 1, 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3678717.3691274

1 Introduction

Human trajectories, representing sequences of location points over time, are extensively collected and analyzed in various real-world applications, ranging from urban planning [43] and transportation management [34] to personalized location-based services [38]. Trajectory embedding learning has emerged as a unified solution to these applications by transforming raw trajectories into vector representations. These embeddings facilitate multiple downstream tasks, such as clustering [15, 44], classification [17], and anomaly detection [22], by capturing the underlying patterns and structures in the trajectory data.

Trajectory embeddings offer numerous advantages over traditional methods. By converting trajectories into fixed-size vector representations, they simplify the complexity of spatio-temporal data and enable the use of powerful machine learning models [17, 20, 40]. These embeddings can capture both spatial and temporal dependencies, making them highly effective for applications such as traffic prediction [33], travel time estimation [21], and location-based recommendations [35, 37]. Moreover, the use of embeddings allows the sharing of trajectory data between entities or organizations without directly exposing sensitive location information [23], thereby promoting collaborative efforts and improving the overall utility of the data. For example, organizations release their trajectory embedding systems, such as LibCity, and pre-trained trajectory embeddings can be published to facilitate downstream tasks [33]. In addition, trajectory embeddings can also be shared among distributed deep learning paradigms that effectively reduce computational load by

*Jiaxin Ding is the corresponding author.

SIGSPATIAL '24, October 29-November 1, 2024, Atlanta, GA, USA

ACM ISBN 979-8-4007-1107-7/24/10
https://doi.org/10.1145/3678717.3691274

adaptively partitioning deep learning inference tasks between mobile devices and the cloud [18]. Trajectory embeddings can be computed locally on mobile devices and then uploaded to the cloud for further analysis, enhancing efficiency.

However, the abstraction provided by vector representations can also introduce significant security and privacy risks. Trajectory data are highly private and can reveal personal information. Previous studies have shown that trajectory data can uncover details like home and work addresses [30], identify social relationships [14], and reveal unique movement patterns [10] that can uniquely identify individuals. As a result, privacy issues associated with location reports and user trajectories have long been recognized as serious concerns [2, 11]. Although trajectory embeddings do not explicitly include human location information, the sharing between different organizations or entities in various applications opens up new vulnerabilities. In graph neural networks (GNNs), graph embeddings can be used to infer edges between nodes [16, 29]. This edge information can be used to infer sensitive relations. In a social network setting, for example, edge information can contain highly private personal relations, and successful link inference can lead to serious privacy consequences. Previous research on natural language processing (NLP) has also shown that sentence embeddings generated by language models for downstream tasks can be used to infer keywords, attributes, and memberships highly associated with original sentences and sensitive individual information, leading to significant privacy concerns [19, 31]. Similarly, these findings can be extended to trajectory embeddings. Adversaries equipped with trajectory embeddings and additional data, such as location embeddings, can potentially reconstruct the original trajectories, thus compromising the privacy of individuals, by exposing their personal movements and sensitive location information [12]. Such privacy risks are particularly concerning and challenging when embeddings are shared or published, however, the potential for reverse-engineering trajectory embeddings and the resulting privacy risks have not been thoroughly studied and understood.

In this paper, we systematically investigate the privacy issues of trajectory embeddings from an adversary's perspective, as shown in Figure 1. We propose two types of attacks aimed at reconstructing the original trajectories using road network information as contextual information to improve reconstruction accuracy and enhance real-world applicability, addressing scenarios where the adversary has varying degrees of access to the black-box representation model. The first attack assumes the adversary's unrestricted access to the model, enabling the adversary to construct a sufficient additional dataset with mappings from trajectories to their embeddings, and train a neural network with the fine-grained road network details to predict the road sequence of the trajectories. The second attack considers the limited access to the representation model, where the adversary only has a limited number of trajectories with embeddings. The adversary uses these trajectories as landmarks, computes distances between road segments and the selected trajectory landmarks as coordinates, along with estimated distances of the embedding by a neural network, to infer the key road segments of the original trajectories with embeddings.

Our study conducts extensive experiments on a real-world dataset to evaluate the effectiveness of these attacks. The results demonstrate that the reconstructed trajectories under adversarial attacks significantly outperform the baseline methods, achieving substantially lower reconstruction errors and more accurate alignment with the original trajectories. We also analyze the characteristics of trajectory embeddings and find that the beginning and end parts of trajectories contribute most in embedding. This observation aligns with the fact that human trajectories exhibit a strong spatial correlation. Furthermore, these points, usually corresponding to the origin and destination, are particularly significant, as they often represent key locations on the trajectory, such as home, work, or places of frequent visit. For adversaries, this also means that the start and end points of trajectories are more susceptible to attacks, potentially revealing visited locations and constituting an invasion of privacy. Our experiment results underscore the need for robust privacy-preserving mechanisms in spatio-temporal data analysis to protect against such adversarial threats. Our contribution can be summarized as follows:

- We propose two adversarial reconstruction attacks on trajectory embeddings under different assumptions. The first attack can reconstruct complex-shaped trajectories that align with the road network through unlimited queries to the target model. This attack leverages the model's full capabilities to accurately predict the sequence of road segments. The second attack infers key road segments by calculating distance coordinates to trajectory landmarks, identifying locations most similar to the estimated coordinates of the original trajectory using a limited amount of data. This approach demonstrates the feasibility of effective reconstruction even with restricted access to the model.
- We conduct extensive experiments to show the effectiveness of the proposed attacks. Experimental results show that current trajectory embedding models are highly vulnerable to our attacks, achieving substantially lower reconstruction errors and more accurate alignment with the original trajectories compared to baseline methods.
- We discuss the characteristic of trajectory embeddings from an adversary's perspective. Our results demonstrate that trajectory embeddings place emphasis on the start and end points, which are crucial for accurately capturing the overall movement patterns. This insight explains the high performance of the attack experiments, as adversaries can exploit these critical points to reconstruct trajectories effectively.

2 PRELIMINARIES

2.1 Notations

We denote a trajectory as $t = [p_1, p_2, \ldots, p_n]$, where $p_i = [lon_i, lat_i]$ represents the i-th location of the trajectory in the form of a longitude and latitude pair. A vehicle trajectory, constrained by roads, can be mapped onto the road network. The road network is denoted as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ where $\mathcal{V}$ represents the set of endpoints of road segments, $\mathcal{E}$ represents the set of directed roads and $\mathbf{X}$ is the locations of all elements. A node $v \in \mathcal{V}$ is a point defined by a longitude and latitude pair where two roads interact, and a road segment $r = [p_1, p_2, \ldots, p_n] \in \mathcal{E}$ is essentially a trajectory where vehicles can only travel from the start to the end. We denote a road mapped trajectory as $t^{map} = [v_1, v_2, \ldots, v_{n'}]$, where v_i represents the i-th node visited by the original trajectory. The

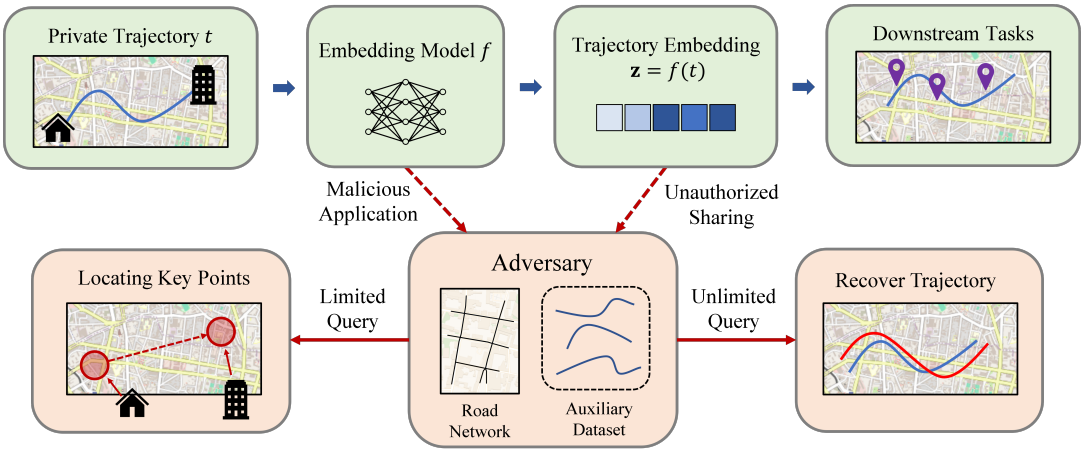


Figure 1: Overview of trajectory reconstruction attack. The embedding z of a private trajectory t, which is shared to third party organizations for downstream tasks, can be obtained by an adversary. The adversary implements attack algorithms to recover the original trajectory t with the assistance of road network information and an auxiliary dataset. Depending on the query capability of the model, the adversary can either reconstruct complex-shaped trajectories or merely locate key points.

Table 1: Notations.

Notation	Description
t	Trajectory
t^{map}	Road mapped trajectory
p	Spatial point
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$	Road network
$r \in \mathcal{E}$	Road segment (also edge of G)
$v \in \mathcal{V}$	Endpoint of road segment (also node of G)
f	Target trajectory embedding model
d	Dimension of trajectory embeddings
$\mathbf{z}$	Trajectory embedding
$\mathcal{A}$	Attack model

frequently used notations introduced here and in the following sections are summarized in Table 1.

2.2 Trajectory Representation Learning

Trajectory data analysis has been extensively studied over a long period. Traditional algorithms, which count on point-to-point comparisons to compute trajectory similarity, often exhibit high time complexity and rely on unique models for specific tasks [1, 3, 5, 42]. To achieve better generality, trajectory representation learning methods transform trajectories into low-dimensional embedded vectors, providing a unified solution. Formally, a trajectory embedding model is defined as $f : t \rightarrow \mathbf{z}$, where $\mathbf{z} \in \mathbb{R}^d$ represents the trajectory embedding. The learned embeddings can be employed for various trajectory analysis tasks, such as similarity computation [40], traffic prediction [33] and location-based recommendation [35].

There exists a large amount of previous work on trajectory representation learning [7, 36]. Early models primarily utilize encoder-decoder architectures to predict surrounding locations. These pioneer approaches such as t2vec [20], traj2vec [41] sample trajectories at low sampling rates and learn trajectory representations by recovering the original trajectories. Supervised learning methods have also been incorporated into trajectory representation learning to enhance capabilities for specific tasks such as similarity computation [40] and anomaly detection [22]. Recent works have turned to graph neural networks to further capture information about the road network [8, 17]. This learning pattern first generates road representations in vector form and then uses a sequence model to compute trajectory embeddings based on these road embeddings. Contrastive learning [6] is a commonly used learning technique in this paradigm, which constructs positive and negative data samples with the aim of reducing the distance between positive pairs and increasing the distance between negative pairs. For both contrastive and non-contrastive representation models, it is assumed that the embeddings of close trajectories should approximately maintain this closeness in the latent space. This property can be leveraged to build attack models for adversaries.

3 PROBLEM FORMULATION

3.1 Motivation

In this paper, we consider a black-box representation model trained by location service providers. Users can access the model to calculate trajectory embeddings for downstream tasks. Specifically, users send a trajectory t to the service provider and obtain a vector $\mathbf{z}$ that embeds the spatial information of t. This kind of vectors can then be shared through online platforms for further analysis in practice since the trajectory information is transformed into a vector form. For example, location service providers can cooperate with other recommendation platforms to utilize multiple data sources and the embedded vectors are regarded as a promising choice to mitigate

privacy issues. However, recovering a trajectory from the vector representation can result in severe privacy concerns.

Previous research in the language and graph domains has demonstrated that embedded vectors can leak information about original data [27, 45]. In these attack studies, the adversary uses auxiliary datasets to access black-box models and obtain training data to decode the embedding vectors. Research in trajectory models follows methods derived from language models, treating trajectories as simple sequences without considering spatial relations among locations [12]. This approach typically divides an area into multiple cells to form a vocabulary, applying language processing methods to trajectories composed of these cells. However, vehicle trajectories can be more accurately represented using road segments. The application of map matching methods can further enhance the trajectory recovery task. From an adversary's perspective, an intuitive way to extract trajectory information from embedded vectors is to leverage a road network, which is the focus of this paper.

3.2 Attack Model

Basically, the adversary obtains an embedded vector $\mathbf{z}$ of a trajectory t. This trajectory and the related vector are termed the target trajectory and the target embedding, respectively. We assume that the adversary has city-level information about the area where the target trajectory is located. Additionally, the adversary can obtain the city's road network from online map resources. The goal of the adversary is to reconstruct the original input trajectory from the given embedding. Achieving this goal can expose private information such as home and workplace locations and even pose a threat to personal security. We follow the commonly studied assumption of the black-box scenario where the target model f (the representation model used to generate the target embedding) is already pre-trained and its parameters are frozen [19, 27, 45]. The adversary can only access the target model without knowing the model architecture and parameters. The adversary keeps an auxiliary trajectory dataset $\mathcal{D}_{aux}$ which contains trajectory data located in the same area as the target trajectory. The auxiliary dataset can be generated in various ways. The adversary can collect real vehicle trajectories recorded by GPS devices. If the adversary has no capability to collect trajectory data, an alternative method is to create a synthetic dataset. For example, the adversary can generate random source-destination pairs and obtain trajectories through navigation services. while individual human movement patterns exhibit significant variation, human vehicle trajectories generally follow similar distributions at a macro level [28]. We assume that the auxiliary dataset has a distribution similar to that of the target trajectory.

In this paper, we consider two types of adversary based on their ability to query the model. The first adversary can access the model without restriction, allowing one to construct a large-scale attack dataset and subsequently train an attack network. The provider of the trajectory representation model is a natural example of an adversary of this type. For the second adversary, we relax the assumption by limiting the number of queries that the adversary can make to the model to a far smaller number. This type of adversary can arise in various scenarios, such as when service providers restrict user access to the model or when the attacker obtains only a small number of samples from a data breach. It is also possible that the adversary is unable to afford the high cost of numerous queries. Since the number of available auxiliary trajectories corresponds to the number of queries, we refer to the size of the auxiliary dataset as the maximum number of queries for simplicity. Formally, the attack is defined as follows.

$$\hat{t} = \mathcal{A}(\mathbf{z}, f, \mathcal{D}_{aux}). \tag{1}$$

4 Adversarial RECONSTRUCTION ATTACK

In this section, we present the proposed adversarial reconstruction attack (ARA) in different settings. Adversary 1 learns a decoder with unlimited queried data to directly recover the trajectories. Adversary 2 searches for roads whose distance coordinates to pre-selected landmarks are most similar to those of the original trajectory and links them to form a reconstruction. We refer to the two attacks as ARA-1 and ARA-2 for simplicity.

4.1 Attack 1: Trajectory Reconstruction Attack with Unlimited Queries

With unlimited queries to the target model, the adversary can fully exploit the information contained within the embedded vectors and thus train an attack neural network to recover the target trajectory. Figure 2 shows the overall process of this attack. To invert a trajectory given its embedding, we propose to use a generative network Φ, computing the next probable point based on the embedding and previous outputs by maximizing the probability of the mapped trajectory $\Pr(t_{aux}^{map} \mid f(t_{aux}))$, where $t_{aux} = [p_1, p_2, \ldots, p_n]$ denotes a trajectory of n points from the auxiliary dataset $\mathcal{D}_{aux}$ and f is the target model. To construct the training set, the adversary sends queries with auxiliary trajectories to the target model and obtains mappings from trajectories to their embeddings. During the training stage, the embedding is used as features. We perform a map matching algorithm [24] on the GPS point trajectory t_{aux} to obtain the mapped one $t_{aux}^{map} = [v_1, v_2, \ldots, v_{n'}]$ which serves as the embedding label. The mapped trajectories need to be set to a uniform length n_{max}, facilitating efficient batch processing. For a trajectory longer than the length, we reserve the source and destination nodes and randomly sample the rest until the length reaches n_{max}. Trajectories shorter than n_{max} are padded to match the uniform length. The reconstruction network Φ is trained by minimizing the cross entropy loss at each step:

$$\mathcal{L}_{\Phi}(t_{aux}^{map}; \theta_{\Phi}) = -\sum_{i=1}^{n_{max}} \log(\Pr(v_i \mid f(t_{aux}), v_1, v_2, \ldots, v_{i-1})), \tag{2}$$

where v_i denotes the i-th node in the mapped trajectory. In the inference stage, the target embedding is input to the reconstruction model and the adversary obtains a node sequence. By mapping the sequence onto the road network, the adversary can further obtain a complex-shaped trajectory that approximates the locations and directions of the target trajectory.

It should be noted that the attack network is essentially a one-to-many decoder based on the frozen target encoder model. Instead of using a sequence as the input to the encoder, the trajectory embeddings are projected to the hidden dimension of the attack model

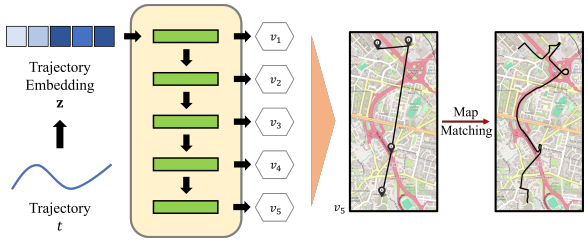


Figure 2: Attack framework of attack 1. The reconstruction network is a decoder which takes a trajectory embedding as input and generates a node point sequence. A map matching algorithm is then applied to the generated sequence to obtain a precise path.

and then fed to the decoder as the initial input. The model Φ receives the embedding as input and then computes the hidden state. The hidden state undergoes two processes: first, it is decoded to generate the output node which is essentially a multi-class classification task; second, it is passed to the next decoding unit. After the final unit, we obtain a sequence of output nodes. Then a map matching algorithm [24] is applied to this sequence to better fit the information from the road network, resulting in the reconstruction $\hat{t}$ of the target trajectory.

4.2 Attack 2: Trajectory Reconstruction Attack with Limited Queries

In the first attack, the adversary needs to send numerous queries to train the attack network. For an adversary with relatively few queries, the trained attack network can yield a catastrophic result. In such situations, we propose to locate a series of key points that the target trajectory is likely to pass through using landmarks. The adversary uses the queried trajectories as landmarks, computing distances between between road segments and the selected trajectory landmarks as coordinates, along with a neural network to estimate the coordinate of the target embedding. By selecting road segments with the most similar coordinates, the adversary can infer the key road segments of the original trajectory. The detailed process is presented in Algorithm 1.

We first select several trajectories from $\mathcal{D}_{aux}$ as landmarks. Here the number of landmarks m is set to be small (less than 5 in this work), following the setting in the previous network alignment studies [26]. We denote $distance(t_1, t_2)$ to represent the distance between trajectory t_1 and t_2. A toy example is shown in Figure 3. We designate the central red point as the target location, and the green trajectories as the landmarks. The blue line segments represent the shortest paths from the target point to the landmarks, with the lengths corresponding to the nearest Euclidean distances between the point and the landmarks. These distances are considered as a coordinate, where nearer points have closer coordinates. We assume that the adversary, lacking precise knowledge of the target point's location, can obtain an approximate distance coordinate of the target point. By repeatedly computing distance coordinates for various locations throughout the area, the adversary can approximate the target point's location by selecting the point with the smallest distance coordinate error. Given that the coordinates are not exact, this approach can be generalized by clustering a set of points with the smallest differences. As Euclidean distances cannot be applied to trajectories of different lengths, we utilize standard trajectory distance metrics to locate a target trajectory, which can be Hausdorff distance [3], Fréchet distance [1], dynamic time warping (DTW) [42], edit distance on real sequences (EDR) [5], etc. These distance metrics are inherently designed to compare two trajectories in both locations and directions, and are thus unsuitable for comparing a single point to a trajectory. Instead of using individual spatial points as in the toy example, we utilize road segments as units to locate the target trajectory.

Because this landmark method can only locate one position at a time, we further divide the trajectory into multiple segments and reconstruct the entire trajectory by locating the endpoints of each segment. Specifically, we divide the trajectory into k equal segments and consider the points near each division point as a sub-trajectory. This means that a trajectory divided into k parts has $k+1$ subtrajectories, with the first part and the last part shorter than the others. By locating different subtrajetories using landmarks, we can link the located positions to form a rough directed trajectory. For instance, we construct a source-destination pair when k equals 1, and add a midpoint when k equals 2. We denote the i-th ($i = 0, 1, \ldots, k$) subtrajectory of a trajectory $t = [p_1, p_2, \ldots, p_n]$ as t^i that includes the points whose indices are around $\lfloor \frac{n \times i}{k} \rfloor$.

We use road segments as the basic trajectory unit to infer the probable location of a subtrajectory. For the i-th subtrajectory of the landmarks, the w-th ($w = 1, 2, \ldots, |\mathcal{E}|$) road segment r has a coordinate with respect to the subtrajectory as $x_w^i = [(x_w^i)_1, (x_w^i)_2, \ldots, (x_w^i)_m]$,

Algorithm 1 Attack with Limited Queries

Input: Target embedding $\mathbf{z}$, target model f, road segment set $\mathcal{E}$, auxiliary dataset $\mathcal{D}_{aux}$, landmark number m, trajectory part number k, most similar road number q
Output: The reconstructed trajectory $\hat{t}$

$\hat{t} \leftarrow \emptyset$
Send queries: $\mathcal{Z}_{aux} \leftarrow f(\mathcal{D}_{aux})$
Train estimation networks $\{\Psi^i\}_{i=1}^n$ with $(\mathcal{D}_{aux}, \mathcal{Z}_{aux})$
Select landmarks $l_1, l_2, \ldots, l_m \in \mathcal{D}_{aux}$
for $i = 0, 1, \ldots, k$-th trajectory part **do**
  Target coordinate $y^i \leftarrow \emptyset$
  Road coordinate $x_1^i, x_2^i, \ldots, x_{|\mathcal{E}|}^i \leftarrow \emptyset$
  for $j = 1, 2, \ldots, m$-th landmark l_j **do**
    $l \leftarrow l_j$
    $y^i \leftarrow y^i \cup \Psi^i(\mathbf{z}, f(l))$
    for $w = 1, 2, \ldots, |\mathcal{E}|$-th road $r_w \in \mathcal{E}$ **do**
      $r \leftarrow r_w$
      $x_w^i \leftarrow x_w^i \cup distance(r, l^i)$
    end for
  end for
  Sort $r_1, r_2, \ldots, r_{|\mathcal{E}|}$ on $\|x_w^i - y^i\|$ in an ascending order
  Selected road set $\mathcal{R} \leftarrow \{r_1, r_2, \ldots, r_q\}$
  Find center $p_c = \sum_{r \in \mathcal{R}} r/|R|$
  $\hat{t} \leftarrow \hat{t} \cup p_c$
end for
return $\hat{t}$

where $(x_w^i)_j = distance(r, l^i)$ denotes the distance between the w-th road r_w and that part of j-th ($j = 1, 2, \ldots, m$) landmark l.

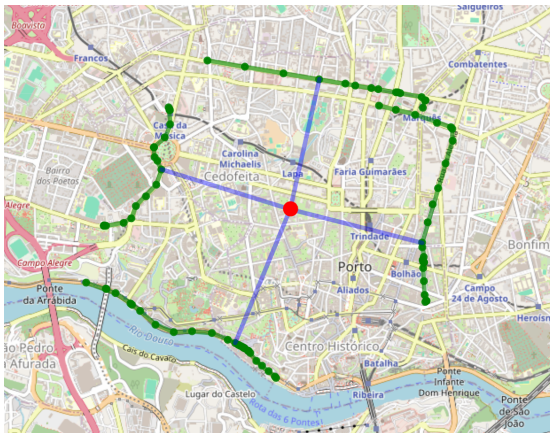


Figure 3: A toy example illustrating the distance coordinate of trajectory landmarks. The green trajectories are pre-defined landmarks and the red point is the target location. The distances between the given point and these landmarks are computed as the lengths of the blue line segments. These distances serve as a coordinate in a multi-dimensional space. By comparing an approximate coordinate with the best-matched coordinates across the area, an approximate location can be identified.

The next is to compute the distance coordinate of the target trajectory with the target embedding. We propose to train a distance estimate network Ψ^i which takes in two embeddings as input and outputs the distance between the certain i-th subtrajectory of the two original trajectories:

$$\Psi^i(f(t_1), f(t_2)) = distance(t_1^i, t_2^i). \tag{3}$$

In this way, we can estimate the distance between the target trajectory and the landmarks by feeding them into Ψ and then obtain an approximate distance coordinate of the target trajectory. This network can be trained on a small auxiliary data set. For any trajectory pair from the auxiliary trajectories, the network Ψ minimizes the squared error between the output and the distance of the two trajectories. To construct the features of the inputs, we concatenate the two trajectory embeddings, as a result of which the output is not symmetric for different input orders. Consequently, the distance needs to be computed as the average of the inputs of the two orders. Essentially, this distance computing method extends the size of the auxiliary dataset to a second order $O(|\mathcal{D}_{aux}|^2)$, which allows for training with fewer queries. The target coordinate $y^i = [y_1^i, y_2^i, \ldots, y_m^i]$ can be computed by:

$$y_j^i = \frac{\Psi^i(\mathbf{z}, f(l)) + \Psi^i(f(l), \mathbf{z})}{2}, \tag{4}$$

where t is the target trajectory and l is the j-th trajectory landmark from $\mathcal{D}_{aux}$. We implement clustering on the q roads with coordinates most similar to the target trajectory by computing the average location of the selected roads. By repeatedly computing the $k+1$ subtrajectories using landmark locating, the adversary can obtain a $(k+1)$-point trajectory, representing an inversion of the target embedding.

5 EVALUATION

5.1 Experimental Setup

Dataset. Our experiments are conducted on the real-world dataset Porto [25]. Porto is an widely used open-source dataset released in a Kaggle competition which consists of over 1.7 million taxi trajectories from 2013 to 2014. We randomly select 10,000 trajectories from it as the target trajectories, which are embedded into vector representations by the target model. We download the road map of Porto from OpenStreetMap[1] and build the road network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ as defined in Section 2.1 for attack purposes. We implement a map matching algorithm [24] to obtain mapped trajectories and retain only trajectories with a Hausdorff distance of less than 500 meters from the original trajectory as training data for the adversary.

Trajectory Embedding Models. We use four trajectory embedding models to perform our attack.

- **Transformer** [32] is an embedding model for sequential data by leveraging self-attention mechanisms to capture contextual relationships within the input sequence. We treat trajectories as cell sequences and use a masked prediction task to train the model.
- **T2vec** [20] converts trajectories to cell sequences and learn a sequence-to-sequence representation model with reconstruction loss.
- **NeuTraj** [40] utilizes a metric learning method in a recurrent neural network (RNN) architecture for fast similarity computation.
- **START** [17] uses a graph attention network to generate road representations and leverages a Transformer architecture to learn trajectory representations based on masked prediction and contrastive learning.

The chosen models each represent a typical category of trajectory embedding models. Transformer is a pure sequence model without consideration of spatial information. T2vec belongs to the encoder-decoder architecture with a reconstruction loss. NeuTraj is designed specifically for similarity computation in a supervised learning pattern. START stands for the GNN plus a sequence model type. All the embedding models are frozen before the attack experiments. We set the dimension d of the output trajectory embeddings to be 64, 128 and 256, which are commonly used in the real-world applications.

Implementations. All experiments are conducted on Ubuntu 20.04 with an NVIDIA GeForce 3090 GPU. We use PyTorch 1.12.1 to implement all the embedding models and attack models.

5.2 Performance of Attack 1

Experimental Settings. For adversary 1, an RNN model with a hidden dimension of $2d$ and an output dimension of $|\mathcal{E}|$ is trained as the reconstruction network Φ. We utilize the gate mechanism [9] to capture data dependencies and use Softmax as the activation function of the output layers. The maximum length of the output n_{max} is set to be 20. We randomly select 100,000 trajectories from Porto

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

Table 2: Comparison of the performance of ARA-1 and baselines with embedding dimensions at 128. Absolute distances HD and FD are measured in meter and normalized distanced NHD and NFD are relative to the diameter of the trajectory.

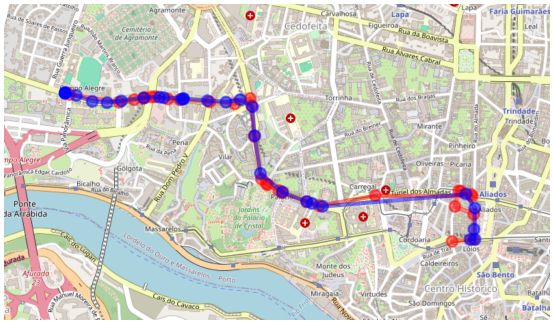
Model	Method	Precision↑	Recall↑	F1↑	HD↓	NHD↓	FD↓	NFD↓
Transformer	MLC	0.7348	0.5172	0.5312	-	-	-	-
	EIA	0.6642	0.1875	0.2895	408.24	**0.1450**	602.16	0.2085
	ARA-1	**0.8358**	**0.7784**	**0.7984**	**388.19**	0.1497	**452.15**	**0.1761**
t2vec	MLC	0.6687	0.2817	0.3633	-	-	-	-
	EIA	0.5796	0.1976	0.2887	528.27	0.1748	617.71	0.2037
	ARA-1	**0.8938**	**0.8447**	**0.8642**	**243.91**	**0.0948**	**290.36**	**0.1131**
NeuTraj	MLC	0.7687	0.4855	0.5245	-	-	-	-
	EIA	0.5893	0.1988	0.2928	539.67	0.1809	595.32	0.1991
	ARA-1	**0.8631**	**0.8067**	**0.8281**	**271.45**	**0.1061**	**313.50**	**0.1226**
START	MLC	0.7688	0.4529	0.4866	-	-	-	-
	EIA	0.5895	0.2289	0.3231	613.45	0.2027	734.08	0.2399
	ARA-1	**0.9262**	**0.8834**	**0.9012**	**272.55**	**0.1031**	**333.30**	**0.1268**

that do not overlap with the target trajectories as the adversary's auxiliary data, where the ratio of the training set to the validation set is set to 9:1. We compare our method with two baseline attacks to demonstrate the efficiency of the proposed attack.

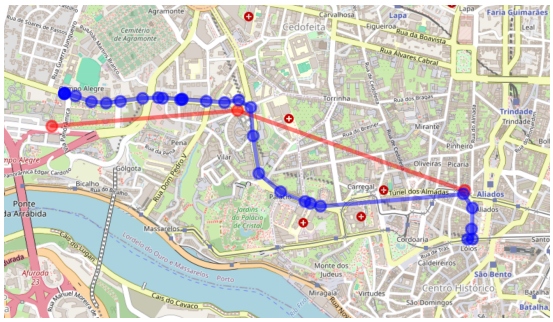
- **Multi-label classification (MLC)** [31] ignores the spatial dependencies of points in a trajectory. The adversary trains a Multi-Layer Perceptron (MLP) with binary cross entropy loss to predict nodes of the original trajectory t given the trajectory embedding z. We set two hidden layers with $2d$ dimension.
- **Embedding inversion attack (EIA)** [12] ignores the interaction between the vehicle trajectory and the road network and removes the map matching stage of our proposed attack. By dividing the whole area into cellular grids, the attacker aims at predicting a cell sequence of the original trajectory.

Evaluation Metrics. We consider two types of metrics: classification metrics and geometry metrics. The classification metrics include node-level precision, recall and F1 score for adversary 1 in the reconstruction network. This type of metrics can evaluate the performance of the proposed method in recovering the original trajectory nodes. The geometry metrics measure the difference between the original trajectory and the reconstructed trajectory. We select Hausdorff distance (HD) and Fréchet distance (FD) as the distance metrics. Hausdorff distance measures the worst-case mismatch between two sets, which is highly sensitive to outliers, while Fréchet distance focuses on the continuous alignment between two trajectories, which is sensitive to the order of points. It is noted that the trajectories vary in lengths, as a result of which an absolute distance cannot fully capture the reconstruction error. We normalize these distances by dividing the distances by the diameter of the minimum enclosing circle of the original trajectory, denoted as normalized Hausdorff distance (NHD) and normalized Fréchet distance (NFD).

Comparison with Baselines. A reconstruction example by ARA-1 is shown in Figure 4(a), where the blue line represents the original trajectory and the red line represents the reconstruction from its embedding. The reconstructed trajectory by the adversary almost coincides with the original trajectory and can even fill in gaps at sparsely sampled locations. We compare our method with the baselines (MLC and EIA), and the experimental results are displayed in Table 2, where the trajectory embedding dimension is set to 128. As MLC only predicts a node set without sequence, we do



(a) ARA-1



(b) ARA-2

Figure 4: Examples of the reconstructed trajectories of the two attacks with ARA-1 on (a) and ARA-2 on (b). The blue trajectory is the original trajectory and the red trajectory is the reconstructed trajectory.

not compute the geometry metrics for it. It can be seen that our proposed ARA-1 method outperforms the two baselines on all four embedding models. For the classification metrics, the reconstruction attack achieves a dominating F1 score of almost 0.8 on all models, indicating the vast majority of points in the original trajectories are recovered, while the other two methods has F1 scores of no more than 0.6. This result reveals the high efficiency of leveraging road matching in the attack works. Compared with EIA, trajectories reconstructed by ARA-1 has smaller values in all four geometric metrics, which demonstrates that the reconstructed trajectories have positions closer to and larger directional similarities with the original trajectories. This is consistent with the classification metric results, since better classification implies closer distances.

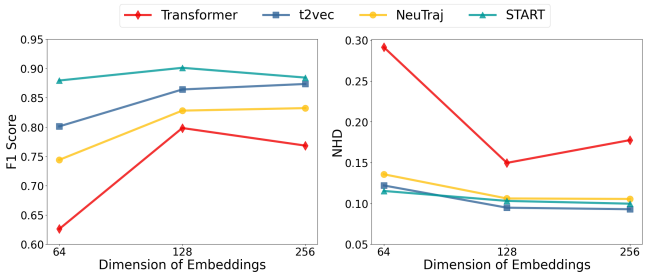


Figure 5: F1 scores and normalized Hausdorff distances (NHD) of our reconstruction attack on all target models given different trajectory embedding sizes, i.e. 64, 128 and 256.

Impact of Trajectory Embedding Dimension. To explore the influence of trajectory embedding dimension, we conduct our attack on different sizes: 64, 128 and 256. Figure 5 displays the experimental results. F1 score and NHD are used to represent the classification and geometry metrics, respectively. We observe a significant decrease in the performance of Transformer embeddings at an embedding dimension of 64, while the attacks perform stably for the other three models across different embedding sizes. From the representation perspective, this result partially demonstrates the embedding capability of the representation model. Increasing the embedding dimension from 64 to 128 yields a significant improvement, whereas increasing it from 128 to 256 does not store useful information. This finding is consistent with the studies on the embedding methods [17, 20, 40]. As Transformer is not designed specifically for spatial data, it needs a larger dimension to fully exploit the trajectory information.

Impact of Training Size. It is important to verify the necessity and practicality of our attack. In our attack 1, we reconstruct trajectories from vector representations with a neural network. This supervised learning task is highly dependent on the size of the training set. If the adversary does not have enough trajectories or is difficult to access the representation model in a black-box manner to obtain a sufficient amount of representation data, the performance of this attack can be significantly compromised. We illustrate how the attack effectiveness changes as the size of the training set varies, as shown in Figure 6.

As we can see, the performance of reconstruction attack with the RNN model remains satisfactory when the size of the training set exceeds 1×10^4. When the size of the training set is reduced to 1×10^3 or less, both the Hausdorff distance and the normalized distance relative to the diameter of the trajectory increase significantly. At the size of the training size reaches 1×10^2, the performance becomes highly unstable. The average Hausdorff distance to the original trajectories exceeds 2,000 meters for most of the target trajectories, and the normalized Hausdorff distance is also greater than 1.0. This indicates that the predicted trajectory is probably outside the minimum enclosing circle of the trajectory. These experimental results demonstrate that while our attack ARA-1 performs well with sufficiently large datasets, it cannot cope with small training sets.

5.3 Performance of Attack 2

Experimental Settings. For adversary 2, we train an MLP as the distance estimation network Ψ. The MLP consists of an input layer with dimension of $2d$, two hidden layers with dimension of $2d$, and an output layer with dimension of 1. We use Relu as the activation function of the hidden layers. The predicted distance is the Hausdorff distance of the input embedding pair which is linearly scaled to the range of 0-1, and Sigmoid is chosen to be the activation function of the output layer. We set the landmark number m to be 4 and the most similar road number q to be 10. Considering that the errors of the reconstructed trajectories accumulate with the trajectory part number k, we set different values of k, i.e. $k \in \{1, 2, 4\}$, to compare the experimental results. We fix the query number to 100 as ARA-1 has a significant performance decline when the data volume reaches 100. Due to the fact that we set the trajectory data queries to be a small number, learning method cannot provide a reliable result as shown in Section 5.2. We do not compare with other baselines and validate our attack scheme based on a small number of queried data.

Evaluation Metrics. We skip the classification metrics as there exist no classification tasks; we only compute geometry metrics defined in Section 5.2. In addition to the distance between the original trajectory and the reconstructed trajectory, we take the points of the reconstructed trajectory and compute the minimum distance between the points and the original trajectory. We denote the absolute distance as PD and the distance normalized by the diameter as NPD.

Impact of Subtrajectory Number k. The subtrajectory number k plays a vital role in the performance of attack 2. A small k implies simple reconstruction, while a large k means complex but also accumulates errors. Table 3 shows the geometry metrics of ARA-2 with respect to different subtrajectory number k, where the embedding dimension of the trajectory is set to be 128. Compared to the results in Figure 6, we observe that ARA-2 performs better on small-scale datasets, with all geometry metrics approximately half of those of ARA-1. For instance, we can achieve 1140.72 meters Hausdorff distance on NeuTraj model. An example with $k = 2$ is shown in Figure 4(b), where the blue line represents a original trajectory and the red line represents the reconstruction from its embedding. Although the distance is larger compared to that with unlimited queries, it can be seen that the reconstructed trajectory can still capture the approximate location and direction of the original trajectory. In addition, we observe that all distance metrics are largest

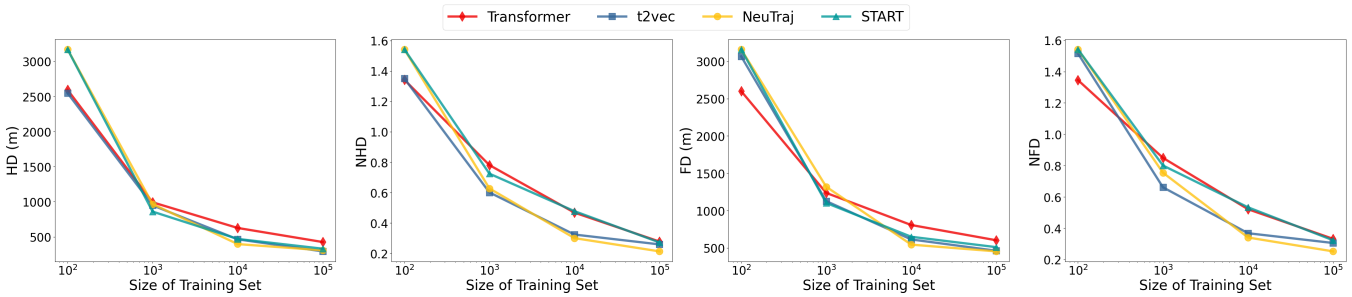


Figure 6: Distances of ARA-1 on different sizes of the training set. Absolute distances HD and FD are measured in meter and normalized distances NHD and NFD are relative to the diameter of the trajectory.

when k increases to 4. NeuTraj performs best when k equals 1 and has the smallest distances among the four models, with all the other models best at $k = 2$. As NeuTraj is the only model trained with a supervised distance computation label, it is reasonable to expect that our attack by locating with landmarks on this model can be more precise, thereby providing higher-level information about the trajectory when k increases to 2. For the other models, increasing the points indicates the accumulation of errors, and thus the best results are achieved when k equals 1.

Table 3: The performance of ARA-2 with 100 queries. Absolute distances HD and FD are measured in meter and normalized distances NHD and NFD are relative to the diameter of the trajectory.

Model	k	HD↓	NHD↓	FD↓	NFD↓
Transformer	1	**1522.25**	**0.6123**	**1873.38**	**0.7530**
	2	1643.22	0.6804	1941.05	0.7964
	4	1677.53	0.7039	2032.00	0.8358
t2vec	1	**1182.95**	**0.4811**	1570.30	**0.6197**
	2	1279.15	0.5277	**1533.11**	0.6275
	4	1477.45	0.6153	1799.74	0.7325
NeuTraj	1	1188.88	0.4816	1614.53	0.6427
	2	**1140.72**	**0.4720**	**1441.40**	**0.5951**
	4	1180.69	0.5002	1520.02	0.6288
START	1	1658.40	**0.6570**	**2024.63**	**0.7970**
	2	**1634.68**	0.6736	2080.06	0.8252
	4	1718.49	0.7233	2253.15	0.8981

Performance of Segments at Different Positions. The experiments show the efficiency of our attack 2, the complete trajectory path can be reconstructed from the representation vector with only a few trajectories. We find that different subtrajectory segments exhibit varying levels of error. Figure 7 shows the distance between the reconstructed points of different positions and the original trajectories. We observed that overall the distances at the start and end points are smaller than those in the middle sections for all models. This result indicates that the distance between the start and end points of trajectories are easier to predict compared with the middle parts.

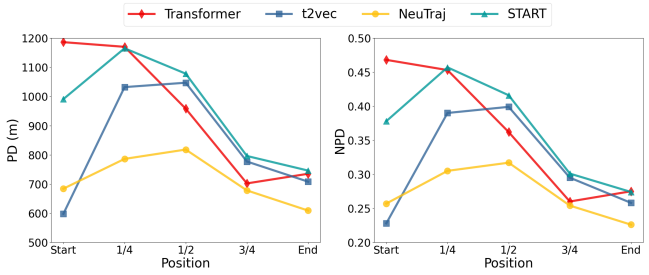


Figure 7: The distances of different positions of the reconstructed points. Absolute distance PD is measured in meter and normalized distance NPD is relative to the diameter of the trajectory.

5.4 Characterising Trajectory Embeddings

In this section, we analyze the performance of attack 2, specifically why the predictions at the start and end points perform better than those at other positions of the trajectory. By deleting portions of trajectory data at different positions, we explore the varying attention that trajectory representation models give to different parts of the trajectory. Specifically, we uniformly divide the trajectory into ten parts, sequentially delete one part at a time, and directly connect the remaining parts to form ten new reduced trajectories. These reduced trajectories are input into the representation model to generate embeddings, and we compute the similarity between the embeddings of the reduced trajectories and the embedding of the original trajectory using cosine similarity and Euclidean distance as metrics. Due to the different similarity metric ranges of various models, directly comparing the similarity scores can obscure comparative results. We normalize these similarity scores by linearly scaling them to the 0-1 range, denoted as normalized similarity and normalized distance. It is important to note that this normalization can result in a normalized similarity score of 0, which does not indicate that the embedding of this reduced trajectory is orthogonal to that of the original trajectory, but that it has the lowest similarity to the original trajectory embedding among all ten reduced trajectories. From the perspective of representation learning, the selection of the nearest neighbor trajectory is based on choosing the one with the highest embedding similarity, rather

than selecting trajectories based on an absolute similarity threshold. Normalizing the similarity scores does not alter their relative magnitudes, and thus does not affect the performance of the vector representations.

The experimental results are shown in Figure 8, where the x-axis represents the position of the starting point of the reduced segment. For instance, 0 indicates that the first one-tenth part (0%-10%) of the trajectory is reduced. It can be observed that reducings at the start and end points result in more significant changes for all four models. This indicates that the start and end points receive more attention, explaining why we achieve more effective performance at the start and end points in Attack 2.

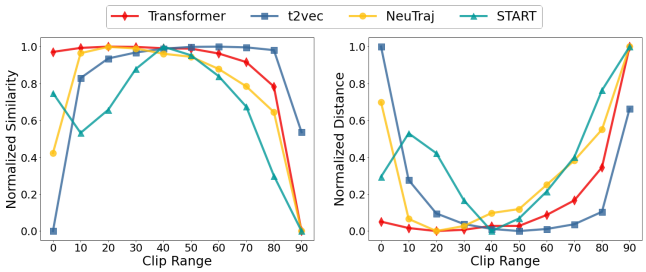


Figure 8: Normalized similarity of the embeddings of the 10 reduced trajectories. The x-axis represents the position of the starting point of the reduced segment. For instance, 0 indicates that the first one-tenth part (0%-10%) of the trajectory is reduced.

This result differs from findings in the natural language processing field. In sentence embeddings, named entities with clear semantics typically carry more information and thus hold greater significance within the sentence [19]. The positional order of words is less important because function words and other semantically insignificant words can appear in various positions within the sentence. For spatial trajectories, due to geographical constraints, human beings cannot quickly transition from one location to a distant one. The positions at the start and end can approximately determine the positions in between. Our result is consistent with this spatial attribute of trajectories.

6 RELATED WORK

Trajectory Representation Models. Trajectory data is typically a sequence of points, with the common structures used to embed trajectories into vector representations to be recurrent neural networks (RNN) and transformer architectures [32]. We refer the readers to [7, 36] for comprehensive overview of different trajectory representation learning models, and introduce the target models used in our attacks here. t2vec [20] utilizes an encoder-decoder framework to learn trajectory representations, trained by recovering trajectories based on masked ones. NeuTraj [40] is another RNN-based trajectory embedding model that leverages a deep metric learning approach for computing trajectory similarity in linear time. START [17] utilizes the typical two-stage learning pattern: first, training a graph neural network over the road network, and second, learning trajectory representations through road embeddings by combining masked prediction and contrastive learning. It integrates spatio-temporal information and leverages the more efficient Transformer framework, achieving state-of-the-art results. These models encompass typical methods utilized in the field of trajectory representation learning. Our attacks can be extended to other models in the same manner.

Embedding Inversion Attack. Although embedding models provide a generalized solution for various downstream tasks, the potential of information leakage needs to be addressed. The adversary can learn sensitive attributes of data embeddings from target models. Embedding inversion has been extensively studied in the graph domain. Chanpuriya et al. [4] propose an optimization algorithm to recover a graph from its node embeddings in a white-box setting. For the black-box setting, there exist multiple attack studies [13, 16, 39] focusing on the link identification task. These studies use auxiliary data and train a shadow model, which requires numerous interactions with the embedding model, which is similar to the setting of our attack 1. Zhang et al. [45] proposes a graph reconstruction attack with auxiliary graph-level embeddings in a black-box setting. MNEMON [29] is a model-agnostic graph recovery attack framework that can recover graph edges only through access to node embeddings without interacting with the target model. This setting is similar to that of our attack 2. In the area of natural language processing (NLP), Song et al. [31] investigate the embedding inversion attack in both white-box and black-box scenarios. They treat sentences as non-sequential data, and only provide a word set that the original sentence may contain. GEIA [19] addresses this by using a transformer architecture to sequentially generate predicted words to form a sentence. The only work on trajectory embedding inversion [12] follows the traditional black-box setting. It directly adopts methods derived in NLP and neglects the spatial relation of trajectories. Our work improves on this by incorporating the road network into the attack and also provides a solution for scenarios with few interactions.

7 CONCLUSION

In this paper, we investigate the privacy risks associated with trajectory embeddings from an adversarial perspective, proposing two effective attack models. Our extensive experiments demonstrated that these attacks significantly outperform baseline methods, revealing substantial vulnerabilities. We found that trajectory embeddings emphasize the start and end points, making them particularly susceptible to privacy breaches. Our work highlights the urgent need for robust privacy-preserving mechanisms in spatio-temporal data embedding-based analysis. Our work informs the development of more secure trajectory embedding methods, ensuring that the benefits of trajectory analysis can be realized without compromising individual privacy, and advancing the balance between data utility and privacy in spatio-temporal data mining.

Acknowledgments

This work is supported by NSF China under Grant No. 61960206002, 62202299, 62020106005, the National Key Research and Development Plan No. 2022YFB3904204, Shanghai Natural Science Foundation No. 22ZR1429100.

References

[1] Helmut Alt and Michael Godau. 1995. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications* 5, 01n02 (1995), 75–91.

[2] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*. 901–914.

[3] E Belogay, C Cabrelli, U Molter, and R Shonkwiler. 1997. Calculating the Hausdorff distance between curves. *Inform. Process. Lett.* 64, 1 (1997).

[4] Sudhanshu Chanpuriya, Cameron Musco, Konstantinos Sotiropoulos, and Charalampos Tsourakakis. 2021. Deepwalking backwards: from embeddings back to graphs. In *Proceedings of International Conference on Machine Learning*. 1473–1483.

[5] Lei Chen, M Tamer Özsu, and Vincent Oria. 2005. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*. 491–502.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of International Conference on Machine Learning*. 1597–1607.

[7] Wei Chen, Yuxuan Liang, Yuanshao Zhu, Yanchuan Chang, Kang Luo, Haomin Wen, Lei Li, Yanwei Yu, Qingsong Wen, Chao Chen, et al. 2024. Deep learning for trajectory data management and mining: A survey and beyond. *arXiv preprint arXiv:2403.14151* (2024).

[8] Yile Chen, Xiucheng Li, Gao Cong, Zhifeng Bao, Cheng Long, Yiding Liu, Arun Kumar Chandran, and Richard Ellison. 2021. Robust road network representation learning: When traffic patterns meet traveling semantics. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 211–220.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[10] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports* 3, 1 (2013), 1–5.

[11] Jiaxin Ding, Chien-Chun Ni, and Jie Gao. 2017. Fighting statistical Re-Identification in human trajectory publication. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 1–4.

[12] Jiaxin Ding, Shichuan Xi, Kailong Wu, Pan Liu, Xinbing Wang, and Chenghu Zhou. 2022. Analyzing sensitive information leakage in trajectory embedding models. In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*. 1–10.

[13] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. 2020. Quantifying privacy leakage in graph embedding. In *Proceedings of MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. 76–85.

[14] Nathan Eagle, Alex Pentland, and David Lazer. 2009. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences* 106, 36 (2009), 15274–15278.

[15] Ziquan Fang, Yuntao Du, Lu Chen, Yujia Hu, Yunjun Gao, and Gang Chen. 2021. E2dtc: An end to end deep trajectory clustering framework via self-training. In *Proceedings of the 2021 IEEE 37th International Conference on Data Engineering*. 696–707.

[16] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. 2021. Stealing links from graph neural networks. In *Proceedings of the 30th USENIX Security Symposium*. 2669–2686.

[17] Jiawei Jiang, Dayan Pan, Houxing Ren, Xiaohan Jiang, Chao Li, and Jingyuan Wang. 2023. Self-supervised trajectory representation learning with temporal regularities and travel semantics. In *Proceedings of the 2023 IEEE 39th International Conference on Data Engineering*. 843–855.

[18] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. 2017. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News* 45, 1 (2017), 615–629.

[19] Haoran Li, Mingshi Xu, and Yangqiu Song. 2023. Sentence Embedding Leaks More Information than You Expect: Generative Embedding Inversion Attack to Recover the Whole Sentence. In *Proceedings of Findings of the Association for Computational Linguistics*. 14022–14040.

[20] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *Proceedings of the 2018 IEEE 34th International Conference on Data Engineering*. 617–628.

[21] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task representation learning for travel time estimation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1695–1704.

[22] Yiding Liu, Kaiqi Zhao, Gao Cong, and Zhifeng Bao. 2020. Online anomalous trajectory detection with deep generative sequence modeling. In *Proceedings of the 2020 IEEE 36th International Conference on Data Engineering*. 949–960.

[23] Mohammad Malekzadeh, Anastasia Borovykh, and Deniz Gündüz. 2021. Honest-but-curious nets: Sensitive attributes of private inputs can be secretly coded into the classifiers' outputs. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 825–844.

[24] Wannes Meert and Mathias Verbeke. 2018. HMM with non-emitting states for Map Matching. In *Proceedings of European Conference on Data Analysis*.

[25] Luís Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luis Damas. 2016. Time-evolving OD matrix estimation using high-speed GPS data streams. *Expert systems with Applications* 44 (2016), 275–288.

[26] Chien-Chun Ni, Yu-Yao Lin, Jie Gao, and Xianfeng Gu. 2018. Network alignment by discrete ollivier-ricci flow. In *Proceedings of International Symposium on Graph Drawing and Network Visualization*. 447–462.

[27] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy risks of general-purpose language models. In *Proceedings of the 2020 IEEE Symposium on Security and Privacy*. 1314–1331.

[28] Farjana Shatu, Tan Yigitcanlar, and Jonathan Bunker. 2019. Shortest path distance vs. least directional change: Empirical testing of space syntax and geographic theories concerning pedestrian route choice behaviour. *Journal of Transport Geography* 74 (2019), 37–52.

[29] Yun Shen, Yufei Han, Zhikun Zhang, Min Chen, Ting Yu, Michael Backes, Yang Zhang, and Gianluca Stringhini. 2022. Finding mnemon: Reviving memories of node embeddings. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2643–2657.

[30] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. 2010. Limits of predictability in human mobility. *Science* 327, 5968 (2010), 1018–1021.

[31] Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 377–390.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017).

[33] Jingyuan Wang, Jiawei Jiang, Wenjun Jiang, Chao Li, and Wayne Xin Zhao. 2021. Libcity: An open library for traffic prediction. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*. 145–148.

[34] Jingyuan Wang, Xin Lin, Yuan Zuo, and Junjie Wu. 2021. Dgeye: Probabilistic risk perception and prediction for urban dangerous goods management. *ACM Transactions on Information Systems* 39, 3 (2021), 1–30.

[35] Jingyuan Wang, Ning Wu, Wayne Xin Zhao, Fanzhang Peng, and Xin Lin. 2019. Empowering A* search algorithms with neural networks for personalized route recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 539–547.

[36] Sheng Wang, Zhifeng Bao, J Shane Culpepper, and Gao Cong. 2021. A survey on trajectory data management, analytics, and learning. *Comput. Surveys* 54, 2 (2021), 1–36.

[37] Sheng Wang, Mingzhao Li, Yipeng Zhang, Zhifeng Bao, David Alexander Tedjopurnomo, and Xiaolin Qin. 2018. Trip planning by an integrated search paradigm. In *Proceedings of the 2018 International Conference on Management of Data*. 1673–1676.

[38] Yong Wang, Guoliang Li, and Nan Tang. 2019. Querying shortest paths on time dependent road networks. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1249–1261.

[39] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. 2022. Linkteller: Recovering private edges from graph neural networks via influence analysis. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy*. 2005–2024.

[40] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering*. 1358–1369.

[41] Di Yao, Chao Zhang, Zhihua Zhu, Jianhui Huang, and Jingping Bi. 2017. Trajectory clustering via deep representation learning. In *Proceedings of the 2017 International Joint Conference on Neural Networks*. 3880–3887.

[42] Byoung-Kee Yi, Hosagrahar V Jagadish, and Christos Faloutsos. 1998. Efficient retrieval of similar time sequences under time warping. In *Proceedings of the 14th International Conference on Data Engineering*. 201–208.

[43] Nicholas Jing Yuan, Yu Zheng, Xing Xie, Yingzi Wang, Kai Zheng, and Hui Xiong. 2014. Discovering urban functional zones using latent activity trajectories. *IEEE Transactions on Knowledge and Data Engineering* 27, 3 (2014), 712–725.

[44] Mingxuan Yue, Yaguang Li, Haoze Yang, Ritesh Ahuja, Yao-Yi Chiang, and Cyrus Shahabi. 2019. Detect: Deep trajectory clustering for mobility-behavior analysis. In *Proceedings of the 2019 IEEE International Conference on Big Data*. 988–997.

[45] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. 2022. Inference attacks against graph neural networks. In *Proceedings of the 31st USENIX Security Symposium*. 4543–4560.