# Machine Unlearning on Trajectory Data: An Experimental Analysis


Yuanjun Liu
Soochow University
Lingnan University
Suzhou, Jiangsu, China
yjliu1@stu.suda.edu.cn

Jiaxing Shen
School of Data Science
Lingnan University
Hong Kong, China

Weiqi Wang
School of Computing
University of Technology
Sydney
Sydney, Australia

Jingwen Li
School of Computer Scienci
Sichuan Normal University
Chendu, Sichuan, China

Jiajie Xu
School of Computer Sci. and Tech.
Soochow University
Suzhou, Jiangsu, China

Lei Zhao
School of Computer Sci. and Tech.
Soochow University
Suzhou, Jiangsu, China

Haoran Xie
School of Data Science
Lingnan University
Hong Kong, China

An Liu*
School of Computer Sci. and Tech.
Soochow University
Suzhou, Jiangsu, China
anliu@suda.edu.cn



## ABSTRACT

Machine Learning (ML) models have become essential for trajectory data analysis, supporting tasks such as similarity learning, map matching, simplification, and recovery. While learning from trajectory data enables personalized and location-based services, the ability to unlearn specific samples is equally crucial for removing outdated information, mitigating privacy risks, and complying with emerging data regulations. Existing machine unlearning methods, however, have been developed almost exclusively for the Computer Vision (CV) domain and their applicability to trajectory learning remains largely unexplored. In this work, we present the first systematic study of machine unlearning for trajectory data. We investigate (i) whether and how representative unlearning algorithms can be adapted to trajectory learning models, (ii) how to evaluate their effectiveness, and (iii) how unlearning outcomes vary across datasets, tasks, and unlearning scenarios. To this end, we design a comprehensive experimental framework covering eleven unlearning algorithms, four key trajectory learning tasks, and multiple evaluation metrics on three real-world datasets. Our study reveals the unique characteristics and sensitivities of unlearning on trajectory data, provides insights into the limitations of existing approaches, and highlights open challenges for future research. The results establish a foundation for building standardized benchmarks and advancing machine unlearning in trajectory analytics.



## CCS CONCEPTS

• **Information systems** → **Location based services**; • **Security and privacy** → **Security services**; • **Computing methodologies** → **Machine learning**.





*Conference acronym 'XX, June 03–05, 2018, Woodstock, NY*





ACM ISBN 978-1-4503-XXXX-X/2018/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

## KEYWORDS

Trajectory Data Mining, Machine Unlearning, Trajectory Similarity, Trajectory Simplification, Map Matching, Trajectory Recovery

**ACM Reference Format:**
Yuanjun Liu, Jiaxing Shen, Weiqi Wang, Jingwen Li, Jiajie Xu, Lei Zhao, Haoran Xie, and An Liu. 2018. Machine Unlearning on Trajectory Data: An Experimental Analysis . In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX


## 1 INTRODUCTION

With the widespread availability of GPS devices and the increasing use of location-based services [7, 26, 46], massive volumes of trajectory data are being collected, typically represented as sequences of points containing location and timestamps. Consequently, various trajectory learning models have been proposed to support fundamental trajectory tasks, including the TMN [76] and BLUE [86] models for trajectory similarity learning, S3 [20] and MLSimp [65] models for trajectory simplification, GraphMM [47] and MMA [67] models for trajectory map matching, and TERI [10] and TRMMA [67] models for trajectory recovery, among others. For example, a collected trajectory is first matched to the map to reduce GPS noise during collection and to prepare for map-based services, then simplified to reduce storage space, and finally recovered to compute representations via similarity learning to serve $k$-NN queries.

While trajectory models are expected to learn information from data, it is equally important for them to *unlearn* information. On the one hand, trajectory data, collected with spatial and temporal information, lose their utility if they become outdated or pass through a spatial area that has changed its function. On the other hand, trajectory learning models may memorize sensitive information of users [6, 67], raising ethical and security concerns. Thus, legal frameworks, such as the General Data Protection Regulation (GDPR) [18], have been enacted to guarantee users the right to request the removal of their data. Therefore, unlearning information from learned trajectory models is crucial for both providing better services and protecting user privacy.

Machine unlearning, which aims to erase the influence of specified samples from trained ML models, can remove out-of-date data and protect the privacy of specific individuals. Given the original model $\theta$ trained on the original dataset $D$ and the unlearning set $D_u \subset D$, the unlearned model $\theta_u$ is expected to be similar to the model retrained on the remaining set $D_r = D \setminus D_u$. Although retraining unlearns exactly, it is computationally prohibitive. For example, JD.com reportedly collects 1 TB of trajectory data every day [40], retraining models on such a volume of data is impractical. Consequently, numerous unlearning algorithms have been proposed to approximate retraining. These include statistic-based methods [5, 59], adversarial-based methods [14, 66], information matrix based methods [24, 25], teacher-student based methods [15, 36], and perturbation-based methods [23, 90], among others.

However, existing machine unlearning research mainly focuses on the CV field, the machine unlearning on trajectory data remains unexplored. The following questions need to be answered. First, it is unclear whether existing unlearning algorithms can be directly applied to trajectory learning models. If not, what are the reasons, and can we make adaptations? Second, the performance of unlearning algorithms on trajectory data is unknown. How should we evaluate the performance and unlearning efficacy on trajectory data? Do existing unlearning algorithms perform as well as they do in CV tasks, and are they consistent across different trajectory learning tasks and scenarios? Which algorithms should be used for different trajectory learning tasks and scenarios? Third, if the unlearning algorithms do not perform as expected, what are the reasons? The characteristics of trajectory data and different trajectory learning tasks need to be clarified. Do these characteristics affect unlearning and evaluation, and if so, how? Fourth, do existing unlearning algorithms satisfy the demands of unlearning trajectory data? What are the remaining challenges of unlearning on trajectory data?

To answer the above questions, we initiate the study of unlearning trajectory data in learned trajectory models in this paper. Specifically, we study its effect on four key trajectory tasks and inform the community of our findings and open problems regarding unlearning trajectory data. Section 2 discusses related works. Section 3 introduces the definitions of four trajectory learning tasks and the details of selected state-of-the-art (SOTA) trajectory learning models for each task. Section 4 provides details on ten machine unlearning algorithms, the adaptations required for specific algorithms, and the reasons why several unlearning algorithms cannot be applied to trajectory data directly. Moreover, it is challenging to measure how well a model has truly forgotten the deleted cohort of data after an unlearning algorithm is applied, as defining an appropriate set of evaluation metrics is an open problem in and of itself. We elaborate on these metrics in Section 5. Then, we conduct comprehensive experiments on three real-world trajectory datasets, two unlearning scenarios, four trajectory learning tasks and models, and ten unlearning algorithms in Section 6. Experimental results demonstrate the performance of unlearning algorithms on different datasets, metrics, and scenarios. Section 7 discovers the characteristics of trajectory data and learning tasks, as well as their effects on unlearning algorithms. Section 8 further discusses several open problems of unlearning on trajectory data and suggests potential solutions. Finally, we conclude our work in Section 9.

The main contributions of this paper are as follows.

- We initiate the study of unlearning on trajectory data, investigating four trajectory learning tasks with corresponding models and ten unlearning algorithms, detailing how to combine the algorithms with trajectory learning models, and how to set appropriate metrics for measuring performance on both trajectory tasks and unlearning efficacy.
- We conduct comprehensive experiments with three real-world trajectory datasets and two unlearning scenarios, and analyze the results to answer a large number of key questions and summarize the learned lessons and open problems, offering valuable insights to guide future research.
- The open-source code [1] unifies the development process of unlearning on trajectory data, with implemented unlearning algorithms serving as a unified benchmark.

## 2 RELATED WORK

Our study connects trajectory learning and machine unlearning, so we introduce the literature on both of them.

### 2.1 Trajectory Data Mining

**Trajectory Similarity Computation.** Trajectory similarity computation methods can be divided into two groups, rule-based methods, e.g., DTW [1], STS [38], and ITS [83], and learning-based methods, such as BLUE [86], LH-plugin [64], and GPE [48]. Since rule-based methods rely on pre-defined rules, which cannot take advantage of the information inside the data, and are typically time-consuming, learning-based methods have become more popular recently. Learning-based approaches use neural networks, such as RNNs, GNNs, and Transformers, to convert trajectories into representation vectors, subsequently computing the similarity of vectors as the similarity of trajectories. t2vec [41], At2vec [45], NeuTraj [77], TMN [76], Aries [21], and RSTS [12] utilize RNNs or Transformers to represent trajectories. GTS [87], KGTS [13], GRLSTM [88], TrajGAT [78], ST2Vec [19], START [32], and JGRM [52] combine GNNs on road networks with RNNs or Transformers to represent trajectories. MMTEC [44] leverages the CDE network for trajectory representation. GPE [48] and GREEN [89] study better position embedding methods. LH-plugin [64] employs hyperbolic space to address the triangle inequality issues. BLUE [86] proposes a hierarchical architecture to model trajectories.

**Trajectory Simplification.** Trajectory simplification tasks can be categorized into two types, minimizing the number of remaining points given a bounding error, and minimizing the error given a fixed number of remaining points. The perpendicular Euclidean distance (PED) [54], synchronized Euclidean distance (SED) [60], direction-aware distance (DAD) [33], and speed-aware distance (SAD) [57] measure the point-pair-wise errors between the original and simplified trajectories. Given the number of remaining points, DPTS [50] and DOTS [4] minimize the errors. Douglas-Peucker [28] and TDMR [53] are top-down and bottom-up algorithms, respectively, to minimize the number of points with bounded error. Learning-based approaches are typically error-bounded. RLTS [71] and MARL4TS [73] utilize reinforcement learning to choose optimal points to form the simplified trajectory. S3 [20] proposes a Seq2Seq2Seq framework to encode the trajectory and then decode

[1] https://anonymous.4open.science/r/TrajMU-309F/

the simplified trajectory. RL4QDTS [72] and MLSimp [65] optimize simplification based on downstream queries.

**Trajectory Map Matching.** HMM [58] uses the Hidden Markov Model to map the trajectory to the most likely road route. ST-Matching [51] aims to align low sampling-rate trajectories with temporal and speed constraints. OLMM [42] proposes an online algorithm for real-time map matching supported by a group of seed trajectories. SnapNet [55] is an incremental HMM algorithm for low sampling-rate cellular trajectories that filters noise points. DeepMM [22] employs an attention Seq2Seq framework to map trajectories in the latent space. LHMM [63] uses a deep-learning enhanced HMM model to map cellular trajectories using multi-relational graphs based on the co-occurrence relationship between cell towers and roads. GraphMM [47] builds a graph to mine inter-trajectory and trajectory-road correlations as well as correlations between road segments. DMM [62] developed a deep reinforcement learning-based map matching framework for cellular trajectories based on an encoder-decoder RNN network. RLOMM [8] is an online mapping algorithm based on reinforcement and contrastive learning methods alongside graph and recurrent neural networks. MMA [67] classifies trajectory points to the most likely candidate road segments with attention-enhanced point and segment embeddings.

**Trajectory Recovery.** The trajectory recovery methods can be categorized into recovering in free space and recovering on the road network. CACT [31] recovers trajectories based on trajectory clustering, and SimiDTR [85] recovers trajectories based on similar trajectory searching, both focus on free space. TCVD [82] and TrajRecovery [74] are multi-modal models for recovering trajectories on the road network based on traffic camera data. TERI [10] focuses on trajectory recovery with irregular time intervals based on learnable Fourier features. RNTrajRec [11] employs multi-task learning to mine the rich spatial and temporal information of trajectories and recover trajectories on the road network. LightTR [49] recovers trajectories in free space in a federated learning scenario based on a meta-knowledge enhanced local-global training scheme. ProDiff [3] employs a diffusion model to recover trajectories in free space based on prototype guidance. TRMMA [67] employs a Seq2Seq architecture to obtain the embeddings of recovered points and then maps them to the most likely road segments.

## 2.2 Machine Unlearning

Given a trained model on the training set, machine unlearning [2] aims to remove the influence of the unlearning set, which is a subset of the training set. The unlearning algorithm was first introduced to solve statistical queries with an exact algorithm [5], but it cannot handle deep learning models. Retraining on the remaining set, i.e., the set of data in the training set but not in the unlearning set, is the exact algorithm for all models, but it is time-consuming. SISA [34] is an unlearning framework that partitions data and trains multiple models on each partition, enabling exact unlearning by retraining models only on specific data partitions. However, it degrades into retraining on all remaining data when the unlearning data exists in multiple partitions, and training models on small pieces of data may harm the overall performance.

The approximate unlearning algorithms are mainly assembled with basic techniques, e.g., finetuning, distillation, influence estimation, adversarial noise, and parameter perturbation. NegGrad [24] simply fine-tunes on the unlearning data by maximizing the task loss, but this can easily cause catastrophic forgetting (CF), i.e., low performance on the remaining set. BadT [15] and SCRUB [36] are teacher-student frameworks with distillation. TopK and RandomK [90] perturb parameters and then fine-tune the model on both the remaining and unlearning sets. HessianNewton [25] first computes the Hessian matrix of all parameters to estimate the influence of the unlearning set and then updates the parameters based on the Newton method, but the second-order matrix consumes significant space and is only suitable for small models. Fisher [24] estimates the influence of the unlearning set using the first-order Fisher matrix and then perturbs parameters, but simply perturbing based on the unlearning set can easily cause CF on the remaining set. SSD [23] estimates the influences of both the unlearning set and the remaining set, then adjusts the parameters directly. SFRon [30] weights parameters based on the influences of two sets and then fine-tunes the model. RandLabel [29] and NoiseLabel [81] fine-tune the model on the unlearning set with selected wrong labels. UNSIR [66] generates adversarial data for each unlearning class by maximizing the task loss, then impairs the model with adversarial data and repairs the model with a subset of the remaining set. DLFD [14] generates adversarial noise for each unlearning batch by maximizing the task loss and minimizing the optimal transport loss, then fine-tunes the model with adversarial noise on the remaining batches.

Machine unlearning literature mainly focuses on the Computer Vision field, and there are some studies investigating machine unlearning in other fields, such as graph data [9, 84], large language models [39, 79], databases [27, 35], federated learning [70, 75], and recommendation systems[16, 80].

# 3 TRAJECTORY LEARNING TASKS

Our study aims to be general enough so that its conclusions are pertinent to different trajectory learning tasks with different models. Following, we introduce the definitions of trajectory and four trajectory learning tasks with their corresponding SOTA models.

Definition 1 (Trajectory). *A trajectory $T$ is defined as a time-ordered sequence of points, i.e., $T = [p_1, p_2, \ldots, p_n]$, where each point $p_i = (x_i, y_i, t_i)$ records the longitude, latitude, and timestamp.*

*Trajectory Similarity Learning* aims to represent each trajectory as a fixed-dimensional vector, such that the similarity of learned vectors represents the similarity of the trajectories. The SOTA model BLUE [86] is a transformer-based hierarchical encoder to represent trajectories, manipulating hierarchical trajectory patches based on the decimals of coordinates, and trained with a decoder and mean square error (MSE) loss.

*Trajectory Simplification* aims to eliminate points from a trajectory $T$ to obtain a simplified trajectory $T'$, such that $T'$ has fewer points and its error is small. The SOTA model MLSimp [65] is a GNN-based model to simplify trajectories by evaluating the importance of points, and trained with a diffusion-based model via mutual learning. The model also utilizes the spatial and temporal range queries on the dataset to optimize the results. Since the mutual learning involves several rounds and the GNN model is retrained

at each round, we treat preceding rounds as the pre-training of the diffusion model and the final round as the training of the GNN model. Moreover, the grid embeddings are pre-trained based on the original dataset $D$.

*Trajectory Map Matching* aims to transform a trajectory to a sequence of road edges or road nodes, where the road edges and nodes form the road network, and high matching accuracy is demanded. The SOTA model MMA [67] is a NN-enhanced algorithm for map matching. It matches each point with the most likely edge from the top-$k$ candidate road network edges retrieved by the R-tree index [37], and the required sequence is obtained by linking the edges of all points sequentially. The node2vec process that yields the embeddings of road network edges is regarded as pre-training.

*Trajectory Recovery* aims to transform trajectory $T = [p_1, \ldots, p_n]$ to $T' = [p'_1, \ldots, p'_m]$, such that the error is minimized. Generally, $T$ is sparser than $T'$, i.e., $n < m$, or $T$ has missing parts, i.e., $\exists i, j \in [2, n], p_{i+1}.t - p_i.t >> p_{j+1}.t - p_j.t$. The SOTA model TRMMA [67] is a NN-enhanced algorithm for trajectory recovery based on map matching. It computes the optimal road network edge of each recovered point $p'$ among map-matched road network edges of $T$, then recovers the positions. The training of MMA on the original dataset to obtain the embeddings of road segments and map-matching results is regarded as pre-training.

## 4 MACHINE UNLEARNING METHODS

Given a learning algorithm $\mathcal{A}$, the original model $\theta_0$ is trained on the original training set $D$, i.e., $\theta_0 \leftarrow \mathcal{A}(D)$. A well-known learning algorithm is minimizing the task-relevant loss $\mathcal{L}$ by gradient descent, i.e., $\theta_0 \leftarrow \theta_* - \frac{\alpha}{|D|}\sum_{x\in D}\nabla\mathcal{L}(x)$, where $\alpha$ is the learning rate, and $\theta_*$ represents the randomly initialized parameters.

Definition 2 (Machine Unlearning). *Given the unlearning set $D_u \subset D$ and the original model parameters $\theta_0$, machine unlearning aims to design an unlearning algorithm $\mathcal{U}$ such that the unlearned parameters $\theta_u$ are similar to the retrained parameters $\theta_r$, where $\theta_u \leftarrow \mathcal{U}(D_u|\theta_0)$, $\theta_r \leftarrow \mathcal{A}(D_r)$, and $D_r = D \setminus D_u$ is the remaining set.*

Previous studies [23, 36] also describe unlearning the unlearning set $D_u$ as forgetting the forgetting set. Similarly, the remaining set is also referred to as the retain set.

Below, we introduce ten unlearning methods that are universal to different learning tasks. Each of these offers a different procedure based on the remaining set $D_r$ and/or the unlearning set $D_u$, and is mainly assembled with basic techniques, e.g., finetuning, distillation, influence estimation, and parameter perturbation.

**Retrain**, the 'oracle' solution, retrains the model on the remaining set, i.e., $\theta_r \leftarrow \mathcal{A}(D_r)$. However, this is time-consuming, whereas unlearning algorithms are expected to be more efficient.

**FineTune** simply fine-tunes the model on the remaining set $D_r$ for a small number of epochs, steering the model towards forgetting the unlearning set $D_u$, i.e., $\theta_u \leftarrow \theta_0 - \frac{\alpha}{|D_r|}\sum_{x\in D_r}\nabla\mathcal{L}(x)$.

**NegGrad** [24] fine-tunes the model on the remaining set $D_r$ with a negated gradient, attempting to unlearn $D_u$ by maximizing the loss via gradient ascent on that data and 'undoing' the gradient descent process that the network had previously undergone, i.e., $\theta_u \leftarrow \theta_0 + \frac{\alpha}{|D_u|}\sum_{x\in D_u}\nabla\mathcal{L}(x)$.

**BadT** [15] employs the original model as a good teacher and a randomly initialized model as a bad teacher to train a new student model on the remaining set $D_r$. This is done by minimizing the task loss and the distillation loss with the good teacher while maximizing the distillation loss with the bad teacher, i.e., $\theta_u \leftarrow \theta_* - \frac{\alpha}{|D_r|}\sum_{x\in D_r}\nabla(\mathcal{L}(x) + \gamma\mathcal{L}_d(x|\theta_0)) + \frac{\alpha\gamma}{|D_r|}\sum_{x\in D_r}\nabla\mathcal{L}_d(x|\theta_*)$, where $\mathcal{L}_d$ is the distillation loss and $\gamma$ is its weight.

**SCRUB** [36] employs the original model as the teacher model and fine-tunes the model by maximizing the task loss and the distillation loss on the unlearning set $D_u$, and minimizing the two losses on the remaining set $D_r$, iteratively, i.e., $\theta_u \leftarrow \theta_0 + \frac{\alpha}{|D_u|}\sum_{x\in D_u}\nabla(\mathcal{L}(x)+\gamma\mathcal{L}_d(x|\theta_0)) - \frac{\alpha}{|D_r|}\sum_{x\in D_r}\nabla(\mathcal{L}(x)+\gamma\mathcal{L}_d(x|\theta_0))$.

For the BadT and SCRUB methods, the distillation loss is set as the KL loss for the MMA and TRMMA models, since their loss is classification loss, and set as the MSE loss for others.

**GDRGMA** [43] observes the conflict between the gradients of the remaining data and unlearning data, which hinders the unlearning effect. It dynamically modifies the directions and magnitudes of gradients for remaining and unlearning batches, such that the cosine value of two gradients is not less than zero, then updates the model with a gradient composed of $(1-\lambda)\times$ modified gradient of the remaining batch $+\lambda\times$ modified gradient of the unlearning batch, where $\lambda$ is a self-tuning parameter.

**TopK** and **RandomK** [90] perturb a fraction of model parameters, where RandomK picks randomly, while TopK selects most sensitive parameters. Then, they finetune the model by minimizing the task loss on the remaining batches and the JS loss on the unlearning batches with the original model.

The sensitivity of parameters is measured with the Fisher information. Given a dataset $D$ and the task loss function $\mathcal{L}$, the Fisher information value of each parameter $\theta^{(i)} \in \theta$ is computed as

$$F_i(D) = \frac{1}{|D|}\sum_{T\in D}\left(\frac{\partial}{\partial\theta^{(i)}}\mathcal{L}(x \mid \theta)\right)^2$$

**SFRon** [30] first computes parameter weights by measuring the Fisher information on the remaining set $D_r$ and unlearning set $D_u$, and tunes the parameters $\theta'$ having more Fisher information on the unlearning set $D_u$, i.e., $\theta' = \{\theta^{(i)}|F_i(D_u) > F_i(D_r), \theta^{(i)} \in \theta\}$. Then, it iteratively performs gradient ascent adjusted by weights on the unlearning batch and gradient descent on the remaining batch.

**SSD** [23] adjusts the parameters directly by suppressing the parameters that have higher Fisher information on the unlearning set $D_u$, i.e., $\theta^{(i)} = \min(\lambda F_i(D)/F_i(D_u), 1)\theta^{(i)}$ if $F_i(D_u) > \alpha F_i(D)$ else $\theta^{(i)}$, where $\lambda$ and $\alpha$ are hyper-parameters.

Besides the above methods, there is a wide range of unlearning methods designed for specific tasks or requiring particular conditions, that *can hardly be applied to trajectory data directly, and how to migrate them is an open problem.* UNSIR [66], RandLabel [29], and NoiseLabel [81] are suitable for multi-class classification, but the BLUE and MLSimp models do not use classification loss, and the MMA and TRMMA models depend on binary classification, which degrades the core adversarial step to simply maximizing the task loss. DLFD [14] learns adversarial noise, but since trajectory shapes are highly variable, the learned adversarial noise on specific trajectories can hardly be applied to others. Moreover, the trajectory models have complex and non-differentiable processes. Specifically,

the BLUE model builds hierarchical patches for trajectories, the ML-Simp model transforms trajectories into sequences of grids, and the TRMMA model matches trajectories to sequences of road segments. These operations block the gradients to adversarial noise.

## 5 MEASURING THE IMPACT OF UNLEARNING

As mentioned earlier, evaluating unlearning is an open problem in ML literature. Besides running time, two aspects are mainly measured. The Membership Inference Attack (MIA) evaluates the privacy status of the model, and task metrics measure the performance of the model on downstream tasks.

**MIAs.** The core idea behind MIA is to train a binary classifier to distinguish between members, i.e., the remaining set $D_r$, and non-members, i.e., the unseen validation set $D_v$. Then, the classifier is employed to infer the membership of the unlearning set $D_u$. The MIA value of a trajectory $T$ is 1 if $T$ is classified as a member else 0. The MIA score is computed as the averaged classification result on the unlearning set $D_u$, i.e., $\frac{1}{|D_u|}\sum_{T_i \in D_u} MIA(\theta, T_i)$, where $MIA(\theta, T_i)$ is the classification result of trajectory $T_i$.

We focus on the common black-box attack setting where the attacker observes only the outputs of the model. Thus, we adopt two attacks used in unlearning literature and apply them to trajectory models, the confidence-based attack [23], where the MIA classifier is trained on the output vectors, and the sample-based attack [68], where the MIA classifier is trained on the output trajectories. For the trajectory similarity and map-matching tasks that yield the representations and the probabilities, we use the confidence-based attack. Specifically, for the map-matching task, we take the top-$k$ probabilities of each point and treat each point as an individual sample, where $k$ is set as 10 following [67]. For the trajectory simplification and recovery tasks that output the sampled and recovered trajectories, we use the sample-based attack.

**Task Metrics.** It is worth noting that trajectory learning models are designed with robustness. For the trajectory similarity task, models can infer on unseen datasets directly with a little performance drops [48, 86]. For the trajectory simplification task, models adjust results based on queries on the dataset [65, 72], utilizing information from unseen data. For map-matching and trajectory recovery tasks, models consider nearby candidates [8, 11, 67], ensuring that matching results are not drastically incorrect. Thus, we evaluate downstream tasks on not only unlearning set $D_u$ and remaining set $D_r$, but also an unseen validation set $D_v$.

For the similarity task, the ranking and $k$-nn tasks quantify the quality of similarity. Given the query set $D_q$ and database set $D_d$, the smaller the rank of query trajectory $T_i \in D_q$ in $D_d \cup D'_q$, the better the learned similarity, where $D'_q$ is the noised query set, and the noised trajectory $T'_i$ is regarded as the most similar trajectory of query trajectory $T_i$. Let $s_{ij}$ be the similarity of trajectory $T_i \in D_q$ and $T_j \in D_d$ and $s_{ii'}$ be the similarity of trajectory $T_i$ and its most similar trajectory $T'_i$, $\gamma_i$ be the rank of $s_{ii'}$ in $(s_{i1}, s_{i2}, \ldots, s_{i|D_d|})$. The mean rank (MR) is defined as $\frac{1}{|D_q|}\sum_{i=1}^{|D_q|}\gamma_i$. Given the query set $D_q$, noised query set $D'_q$, and database set $D_d$, the higher hit ratio of $T'_i$ in $knn(T_i, D_d)$ for a query trajectory $T_i \in D_q$, the better the learned similarity. The hit ratio at the top-$k$ results, i.e., HR@$k$, is computed as $\frac{1}{|D_q|}\sum_{i=1}^{|D_q|}\mathbb{1}T'_i \in knn(T_i, D_d)$, where $\mathbb{1}T'_i \in knn(T_i, D_d)$ is 1 if the mose similar trajectory $T'_i$ of $T_i$ is retrieved, else 0. $MR \geq 1$ and HR@$k \in [0, 1]$, and lower $MR$ and higher HR results mean better performance. The query set is set as the unlearning set $D_u$, the remaining set $D_r$, and the validation set $D_v$, respectively.

For the simplification task, the SED evaluates simplification error, and the F1 measures the performance on range query task. Given trajectory $T = [\ldots, p_i, \ldots]$ and its simplified trajectory $T' = [\ldots, p'_j, \ldots]$, SED measures the maximal distance of each point $p_i \in T$ from its synchronized point $q_i$ having the same time and located on $T'$, i.e., $\max_{p_i \in |T|}\sqrt{(p_i.x - q_i.x)^2 + (p_i.y - q_i.y)^2}$, where $q_i.x = p'_j.x + \delta(p'_{j+1}.x - p'_j.x)$, $q_i.y = p'_j.y + \delta(p'_{j+1}.y - p'_j.y)$, $\delta = \frac{p_i.t - p'_j.t}{p'_{j+1}.t - p'_j.t}$, and $p'_j.t \leq p_i.t \leq p'_{j+1}.t$. SED $\geq 0$ and lower SED is better. Given a set of query points, the original trajectory set $D_d$, and the simplified trajectory set $D'_d$, let $R_i$ be the range query results of a query point $p_q$ on the original set $D_d$ and $R'_i$ be the results on the simplified set $D'_d$, the F1 score of the simplification task is computed as $F1(R_q, R'_q)$, where $F1(x, y) = (2PQ)/(P + Q)$, $P = |x \cap y|/|x|$, and $Q = |x \cap y|/|y|$. F1 $\in [0, 1]$ and higher F1 is better. We report the averaged SED and F1 result on the whole set.

For the map matching task, the Acc and F1 metrics are employed. Given a trajectory $T$, let $S$ be the matched road segments of the ground truth and $S'$ be the matched road segments of the model, the F1 score of the map matching task is computed as $F1(S, S')$. And the Acc counts the number of identical segments, i.e., $Acc = (\sum_{1 \leq i \leq |S|}\mathbb{1}S_i = S'_i)/|S|$, where $\mathbb{1}S_i = S'_i$ is 1 if the $i$-th matched segments are the same. $Acc \in [0, 1]$ and higher Acc is better. We report the averaged Acc and F1 result on the whole set.

For the recovery task, the MAE measures recovery error. Given the recovered trajectory $T' = [\ldots, p'_i, \ldots]$ and its ground truth $T = [\ldots, p_i, \ldots]$, MAE is computed as the average distance of all point pairs, i.e., $\frac{1}{|T|}\sum_{p_i \in T}\sqrt{(p_i.x - p'_i.x)^2 + (p_i.y - p'_i.y)^2}$. MAE $\geq 0$ and lower MAE is better. Moreover, the TRMMA model first matches the to-be-recovered point to the candidate road segments, so the Acc evaluates the accuracy of this process. We report the averaged MAE and Acc result on the whole set.

## 6 EXPERIMENTAL EVALUATION

We empirically evaluate and analyze the aforementioned ten unlearning algorithms on three real-world datasets, in the context of several trajectory tasks and evaluation metrics.

### 6.1 Experimental Setup

**Dataset.** We experiment on three real-world trajectory datasets. Porto [56] contains over one million trajectories collected in Porto, Portugal, from 1 July 2013 to 30 June 2014 by 442 drivers, BJ [61] contains over one million trajectories collected in Beijing, China, from 2 May 2009 to 25 May 2009 by 15,920 users, and Xian [17] contrains over three million trajectories collected in Xian, China, from 1 Oct. 2016 to 31 Oct. 2016 by 535,828 users. All datasets can be downloaded from our code repository. We randomly select 100,000 trajectories as the training set to train the original model and two sets for testing and validating during training, each of size 20,000. Then, the two sets are combined as the unseen validation set $D_v$ to execute the MIA, and the training set is split into two distinct sets, i.e., the remaining set $D_r$ and the unlearning set $D_u$. Specifically,

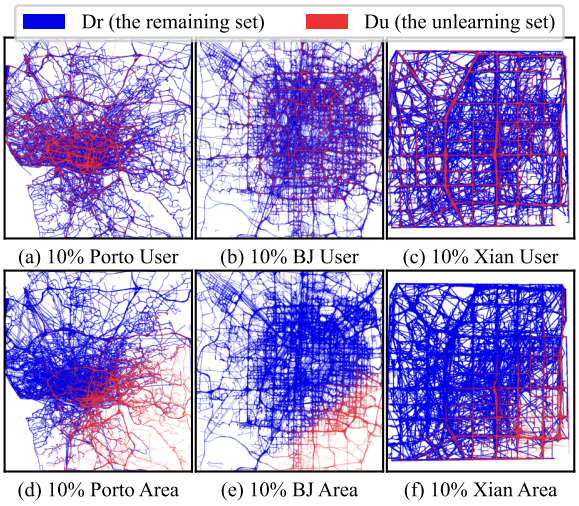


**Figure 1: Dataset Visualization of the unlearning set $D_u$ and the remaining set $D_r$ when unlearning 5% data. (a) the User scenario of the Porto dataset, (d) the Area scenario of the Porto dataset, (b) the User scenario of the BJ dataset, (e) the Area scenario of the BJ dataset, (c) the User scenario of the Xian dataset, (f) the Area scenario of the Xian dataset.**

for MLSimp, the size of the training set is set to 1,000 following [65], and the sizes of other sets are reduced proportionately.

**Unlearning Scenarios.** We consider two unlearning scenarios: Unlearn User, where a group of users requests to unlearn all their trajectories, and Unlearn Area, where a group of trajectories closest to a certain area are considered outdated and unlearned. Each scenario contains three unlearning ratios: 10% 20% and 30%, and we report the averaged results on three ratios.

Given the unlearning ratio $r_u$, the training set $D$, and the target area $\Phi$, the Area scenario unlearns trajectories $D_u$ that are closest to the area, i.e., $dis(T_i, \Phi) \geq dis(T_j, \Phi), T_r \in D_r, T_j \in D_u, |D_u| = |D|r_u, D_r = D \setminus D_u, 0 \leq r_u \leq 1$. This scenario simulates a situation where a certain area changes its function or the trajectory patterns passing through it change, so the learned model is expected to unlearn the outdated information. $dis(T, \Phi)$ computes the distance of trajectory $T$ to the area $\Phi$, and we take the averaged distance of all points to the area center. Given the unlearning ratio $r_u$, the training set $D$ and the user set $\Psi$ with their trajectories $\Upsilon_i$, i.e., $\Upsilon_i \subset D, i \in \Psi$, the User scenario unlearns trajectories $D_u$ belonging to $\Psi_u$, i.e., $D_u = \bigcup_{i \in \Psi_u} \Upsilon_i, |\Psi_u| = |\Psi|r_u, D_r = D \setminus D_u, 0 \leq r_u \leq 1$. This scenario models privacy-driven deletion requests, where users exercise their right to remove their data. Specifically, a fraction $r_u$ of users request the deletion of all their trajectories.

We visualize the remaining set $D_r$ and the unlearning set $D_u$ in Figure 1. It is obvious that the data distributions of the two unlearning scenarios, i.e., the User and Area, are distinct. The User scenario is more random than the Area unlearning, as users are dispersed throughout the city. In contrast, the $D_u$ of the Area scenario is more centralized, and we chose areas at the edge.

**Compared Methods.** We compare the aforementioned ten unlearning methods based on the BLUE, MLSimp, MMA, and TRMMA models for trajectory similarity computation, simplification, map matching, and recovery tasks, respectively. For the unlearning methods, the parameters are set as their default, and the epochs are set as the one where the unlearning metric results do not change anymore. For the BadT and SCRUB methods, the distillation loss is set as the KL loss for the MMA and TRMMA models, since their loss is classification loss, and set as the MSE loss for others. For the TopK and RandomK algorithms, the JS loss here is replaced with the MSE loss for BLUE and MLSimp models since they do not rely on classification loss. Following [90], the weight of the distilling loss is set to 0.1. For the trajectory learning models, they are trained with default configurations, including training epochs, learning rate, batch size, etc. The minimal length of MLSimp is set to 500 for Porto, 110 for BJ, and 400 to Xian to ensure adequate trajectory numbers, and the simplification rate is set to 1% for Porto, 5% for BJ, and 1% for Xian to ensure the simplified trajectories are valid.

**Evaluations.** The running time evaluates the efficiency of unlearning methods. For MIA, the similarity and map matching tasks use the confidence-based MIA, while the simplification and recovery tasks utilize the sample-based method. For task metrics, we employ two groups of metrics. The default metrics are MR for similarity, SED for simplification, F1 for map matching, and MAE for recovery. Another group of metrics are HR@10 for similarity, F1 for simplification, Acc for map matching, and Acc for recovery.

We treat the results of Retrain as the golden standard for task metrics and MIAs, and shorter running time is preferred. Since the metrics have different ranges and preference, e.g., MR$\geq 0$ and lower MR is better, and $F1 \in [0, 1]$ and higher F1 is better, we employ the SimScore to analyze the similarity of between the results of Retrain $x_{gt} \geq 0$ and the results of another unlearning method $x_u \geq 0$. The SimScore of two results $x_u, x_{gt}$ is calculated as $\text{SimScore}(x_u, x_{gt}) = \frac{\min(x_u, x_{gt})}{\max(x_u, x_{gt})}$, SimScore $\in [0, 1]$, and higher score indicates that the unlearning method is better aligned with the Retrain.

**Table 1: Summary Results on All Tasks**

| Method | Ranks of Tasks | | | | Sum | Runtime Speed Up | | | | Sum |
|---|---|---|---|---|---|---|---|---|---|---|
| | Sim. | Simp. | Map. | Rec. | Rank | Sim. | Simp. | Map. | Rec. | Rank |
| FineTune | 3 | 5 | **1** | 5 | 3 | 11 | 114 | 9 | 15 | 5 |
| NegGrad | 9 | 4 | 9 | 9 | 9 | 25 | 288 | 23 | 37 | 3 |
| BadT | 6 | 7 | 2 | 4 | 5 | 2 | 28 | 3 | 2 | 9 |
| SCRUB | 5 | 8 | 8 | 8 | 7 | 3 | 28 | 3 | 2 | 8 |
| GDRGMA | **1** | 6 | 5 | **1** | 2 | 29 | **460** | **38** | **53** | **1** |
| TopK | 8 | 2 | 3 | 6 | 6 | 9 | 98 | 7 | 4 | 6 |
| RandomK | 2 | **1** | 4 | 3 | **1** | 23 | 248 | 16 | 5 | 4 |
| SFRon | 7 | 9 | 7 | 7 | 8 | 7 | 77 | 6 | 10 | 7 |
| SSD | 4 | 3 | 6 | 2 | 4 | **30** | 322 | 27 | 44 | 2 |

## 6.2 Experimental Results

Limited by space, the detail results of four tasks are displayed at Section A. Below, we show the summarized tables.

**Summary Results on Four Tasks** Table 1 displays the ranks of all unlearning algorithms on four trajectory learning tasks, and the summary rank on all tasks. The GDRGMA is good at the similarity and recovery tasks, RandomK suits the simplification, and FineTune is the best for the map matching task. RandomK, GDRGMA, FineTune, and SSD have better overall performance, and NegGrad,

**Table 2: Task Results among Datasets and Metrics**

| Method | Default Metric Group | | | | | | | | | | | | | | | Another Metric Group | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Porto | | | | | BJ | | | | | Xian | | | | | Porto | BJ | Xian |
| | Sim. | Simp. | Map. | Rec. | Rank | Sim. | Simp. | Map. | Rec. | Rank | Sim. | Simp. | Map. | Rec. | Rank | Rank | Rank | Rank |
| Retrain | 1.875 | 0.038 | 0.870 | 100.197 | - | 8.027 | 0.013 | 0.827 | 254.936 | - | 4.206 | 0.019 | 0.956 | 78.145 | - | - | - | - |
| FineTune | 1.633 | 0.039 | **0.886** | 115.246 | 4 | 36.074 | 0.013 | 0.834 | **254.700** | 4 | 2.447 | **0.019** | 0.968 | 107.322 | 4 | 4 | 5 | 3 |
| NegGrad | 5791.420 | 0.039 | 0.110 | 402.384 | 9 | 5087.995 | 0.013 | 0.504 | 713.602 | 9 | 10450.453 | 0.019 | 0.810 | 367.706 | 9 | 9 | 9 | 9 |
| BadT | 4.735 | 0.039 | 0.887 | 115.523 | 5 | 40.597 | **0.013** | 0.836 | 256.179 | 5 | 6.432 | 0.019 | 0.967 | 110.233 | 3 | 5 | 4 | 4 |
| SCRUB | 2.867 | 0.038 | 0.685 | 391.792 | 7 | 74.563 | 0.013 | 0.631 | 659.600 | 8 | 5.991 | 0.019 | 0.850 | 336.765 | 6 | 7 | 7 | 6 |
| GDRGMA | 1.798 | **0.038** | 0.907 | **99.014** | **1** | 20.227 | 0.013 | 0.830 | 255.583 | **1** | 2.849 | 0.019 | 0.971 | **76.860** | **1** | **1** | 2 | **1** |
| TopK | 4566.998 | 0.038 | 0.887 | 111.704 | 6 | 4086.008 | 0.013 | 0.832 | 264.653 | 6 | 4139.831 | 0.019 | 0.967 | 111.608 | 7 | 8 | 8 | 8 |
| RandomK | 2.031 | 0.039 | 0.893 | 104.794 | 3 | 20.909 | 0.013 | 0.833 | 259.473 | 3 | **3.128** | 0.019 | 0.970 | 95.684 | 2 | 3 | **1** | 2 |
| SFRon | 70.556 | 0.039 | 0.759 | 306.531 | 8 | 22.333 | 0.013 | 0.661 | 609.812 | 7 | 41.235 | 0.019 | 0.865 | 292.606 | 8 | 6 | 6 | 7 |
| SSD | **1.820** | 0.039 | 0.902 | 103.376 | 2 | **20.125** | 0.013 | **0.829** | 258.654 | 2 | 346.596 | 0.019 | **0.960** | 87.869 | 5 | 2 | 3 | 5 |

SFRon, and SCRUB have lower rank. The key is that the previous methods do not change parameters significantly. Compared to SSD and SFRon that both utilize the Fisher information values to adjust the parameters, SSD computes the threshold value carefully and modifies parameters with bounding, yet SFRon simply modifies all parameters having more information on the unlearning set $D_u$. Similarly, both TopK and RandomK perturb parameters then fine-tune, but TopK modifies important parameters which ruins the performance. Both NegGrad and GDRGMA tune on the unlearning set $D_u$, NegGrad simply updates the model with opposite gradient and GDRGMA takes the gradient of a remaining batch as reference, so GDRGMA performs better. The SCRUB, SFRon, and NegGrad methods all utilize gradient ascent on the unlearning data, resulting in poor performance, indicating that maximizing the loss on the unlearning data is not practical for trajectory learning. Though RandomK, SSD and FineTune have the best overall performance, they are not good at all tasks respectively.

**Analysis on Efficiency.** Unlearning algorithms are motivated by the desire to avoid the high costs of retraining. We thus evaluate the efficiency of the approximate unlearning algorithms studied. We record the speed-up of each algorithm over Retrain for our four trajectory learning tasks and their corresponding models, and report the averaged speed-up results over unlearning scenarios, datasets, and unlearning ratios. Table 1 shows the results of unlearning efficiency. GDRGMA has the highest speedup, which tunes on a small fraction of data and runs a few epochs. SSD requires no tuning and is more efficient than others. NegGrad tunes on the unlearning set merely and is faster than following methods that tune on the remaining set. Compared to RandomK, TopK needs more time to find sensitive parameters. The BadT algorithm is the most time-consuming unlearning algorithm, since it contains three models, i.e., the good teacher model, the bad teacher model, and the student model, and finetunes on the full remaining set.

**Analysis on Datasets.** Table 2 shows the task results among three datasets and ranks by the SimScore. Regarding datasets, Porto reports better similarity results, BJ has simplification results, and Xian is better for the map matching and recovery task. For the trajectory similarity task, the road network of Porto gives trajectories diverse shapes, making them more distinguishable for answering similar trajectory search queries. For the trajectory simplification task, the trajectories of the BJ dataset are more regular, restricted by the road network, and can be simplified easily. The trajectory map matching and recovery tasks are based on the road network, and the sparser road network of Porto and the regular road network of Xian benefit map matching and recovery, as shown in Figure 1. Moreover, unlearning methods have similar ranks on different datasets, indicating the coincident performance across datasets.

**Analysis on Metrics.** Table 2 displays the results of both the default metric group (i.e., MR for similarity, SED for simplification, F1 for map matching, and MAE for recovery) and another metric group (i.e., HR@10 for similarity, F1 for simplification, Acc for map matching, and Acc for recovery), to verify the performance on different metrics. On the whole, the methods have similar ranks on different metrics, indicating that different task metrics represent the performance with one accord. Specifically, most methods achieve identical ranks across both metric groups, and the rank differences of the remaining methods are all not more than two.

**Table 3: MIA Results among Unlearning Scenarios**

| Method | MIA Results of User | | | | | MIA Results of Area | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Sim. | Simp. | Map. | Rec. | Rank | Sim. | Simp. | Map. | Rec. | Rank |
| Retrain | 0.502 | 0.659 | 0.499 | 0.492 | - | 0.102 | 0.409 | 0.499 | 0.091 | - |
| FineTune | 0.500 | 0.651 | 0.500 | 0.507 | 4 | 0.126 | 0.390 | 0.502 | 0.088 | 2 |
| NegGrad | 0.498 | 0.672 | 0.501 | 0.489 | **1** | 0.497 | 0.431 | 0.500 | 0.127 | 9 |
| BadT | **0.503** | **0.660** | 0.499 | 0.511 | 2 | **0.104** | 0.487 | 0.499 | **0.088** | **1** |
| SCRUB | 0.499 | 0.742 | 0.500 | 0.516 | 9 | 0.093 | 0.510 | 0.477 | 0.099 | 6 |
| GDRGMA | 0.500 | 0.716 | 0.502 | 0.496 | 8 | 0.092 | 0.483 | 0.500 | 0.101 | 4 |
| TopK | 0.499 | 0.635 | 0.501 | 0.498 | 5 | 0.376 | 0.393 | **0.499** | 0.087 | 8 |
| RandomK | 0.500 | 0.709 | **0.499** | 0.501 | 7 | 0.084 | 0.456 | 0.499 | 0.084 | 5 |
| SFRon | 0.499 | 0.668 | 0.501 | 0.501 | 3 | 0.077 | 0.487 | 0.430 | 0.118 | 7 |
| SSD | 0.496 | 0.689 | 0.501 | **0.489** | 6 | 0.118 | **0.416** | 0.482 | 0.078 | 3 |

**Analysis on MIA.** Table 3 presents the averaged MIA results on four tasks among two unlearning scenarios, and ranks by the SimScore. As shown in Figure 1, the unlearning set of the Area scenario is more centralized and distinct from the remaining set, so the MIA values of the Area scenario are much lower. Conversely, the distributions of the unlearning and remaining sets of the User scenario are similar, the remaining set contains the information of the unlearning set, so the MIA values of the User scenario are higher. While BadT and FineTune are good at both scenarios, others perform distinctly on the two scenarios. NegGrad is the best on the User scenario but the worst on the Area scenario, TopK and SFRon are better on the User scenario, and others prefer the Area scenario.

## 7 LEARNED LESSONS

While most questions have been answered in specific sections, several questions need to be reinforced.

**Data Distributions and Their Effects.** We focus on two scenarios, the User scenario, which unlearns all trajectories belonging to a group of users, and the Area scenario, which unlearns a group of trajectories near a specific area. Previous analyses have shown that these two scenarios have different data distributions and produce disparate results. *For the User scenario, the distributions of the remaining set $D_r$ and the unlearning set $D_u$ are similar,* as users live and act all over the city, leading to spatial distributions similar to selecting and unlearning trajectories randomly. Consequently, the remaining set contains the information about the unlearning set, and the MIA results are generally around 0.5. In contrast, *the Area scenario, where trajectories are separated according to spatial location, has more distinguishable distributions between the remaining and unlearning sets.* The MIA results saw larger gaps. What's more, *trajectory data share the same space.* Generally, the longitude and latitude ranges and the road network of the unlearning set $D_u$ and remaining set $D_r$ intersect. It is possible that a point $p$ exists in both datasets, i.e., $p \in T_u, p \in T_r, T_u \in D_u, T_r \in D_r$. Thus, even if the model has not seen a trajectory $T_u$, the spatio-temporal information of the trajectory may be contained in $D_r$ and learned by the model.

Therefore, *trajectory data prefer gentler unlearning algorithms to align similar distributions.* NegGrad and SFRon algorithms employ gradient ascent on the unlearning set, and TopK modifies most sensitive parameters based on the unlearning set, these operations damage parameters dramatically, so they generally underperform in Table 2. RandomK modifies less sensitive parameters and is better than TopK, GDRGMA reduces the gradients conflict, and FineTune simply tunes on the remaining set, they are less aggressive and obtained moderate task ranks.

Moreover, *the road networks also bring shifts in data distribution.* As illustrated in Figure 1 and mentioned in Table 2, the road networks of BJ and Xian are more regular than that of Porto, so their simplification errors are lower. Also, the road networks of Porto and Xian are sparser, so their map matching accuracies are higher, and the recovery errors are lower.

**Characteristics of Different Learning Tasks.** Distinct characteristics complicate unlearning and evaluation. For the trajectory similarity task, models are designed with robustness [48, 86], they can provide high-quality results on unseen datasets directly, and naturally on the unlearning set. For the trajectory simplification task, the model can employ queries to adjust simplification results, and the pre-trained mutual learning model and grid embeddings contain the information about the unlearning set.

The trajectory map matching and trajectory recovery tasks rely on the road network and match points to candidate segments. First, the embeddings of the segments of the road network are pre-trained and fixed, and matching is enhanced by the R-tree index to improve the probability of choosing the truth, so models are likely to maintain performance since the road network remains the same. Second, models built on map road networks inherit the characteristics of unlearning on graphs [9, 84], e.g., dependence on topological structure. Third, recovery is based on matching points to segments, linking segments with their shortest path. If one matched road segment changes, the shortest paths with its adjacent segments change, then the recovered trajectory changes intensely. Thus, recovered trajectories may change significantly if even one segment is mismatched.

## 8 OPEN PROBLEMS

There are several unsolved problems and research directions for unlearning on trajectory data.

**How to migrate other unlearning algorithms?** As mentioned before, there is a group of unlearning algorithms that cannot be applied to trajectory models directly. Adversarial-based algorithms [14, 66] train noises or samples to antagonize the unlearning set $D_u$, then train the model with the noises or samples to stimulate unlearning $D_u$. However, on the one hand, trained noise with a fixed shape is not practical for trajectories with various shapes. On the other hand, trajectory models process trajectories complexly. Specifically, the BLUE model builds hierarchical patches, the MLSimp model gridifies trajectories, and the TRMMA model matches trajectories to the map. These operations block the gradients during the training of adversaries.

**Could we borrow stones from other hills?** There are some questions that are also open or solved problems for other unlearning fields, so we could learn from other fields to design better unlearning algorithms for trajectory data. The MLSimp algorithm utilizes mutual learning to optimize two models iteratively, which is similar to semantic communications [69] that optimize two models simultaneously. The map matching task is performed on the road network, unlearning the map matching model inherits the problems of unlearning on graphs [9, 84]. The unlearning effect of the User scenario is not as obvious as that of the Area scenario since the trajectories of the unlearning set of the User scenario are like randomly selected ones. This phenomenon is also demonstrated in literature on image classification tasks [23], i.e., the results of unlearning images in each class are not as impressive as unlearning the whole class. Therefore, it is possible to handle these problems by consulting unlearning algorithms designed for other fields.

## 9 CONCLUSION

We have presented the first study of machine unlearning on trajectory data. This is a crucial ingredient for providing updated services and protecting user privacy. Our investigation covered four key trajectory learning tasks, i.e., similarity learning, simplification, map matching, and recovery, each employing a SOTA model, across three real-world datasets and two unlearning scenarios. It studied different unlearning methods, including simple baselines, such as Finetune and NegGrad, teacher-student based methods, such as BadT and SCRUB, and information matrix based methods, such as SSD and SFRon, among others, adapted from the machine unlearning literature. Our investigation studied appropriate metrics including task and MIA metrics in order to substantiate and interpret results. Our results answer a large number of related key questions and also point out the characteristics of trajectory data and learning tasks. The work puts forth the basic skeleton, for instance, unlearning algorithms, performance metrics, downstream tasks, and unlearning scenarios, as well as related findings en route to a much-needed benchmark for unlearning on trajectory data.

# REFERENCES


[1] Donald J. Berndt and James Clifford. 1994. Using Dynamic Time Warping to Find Patterns in Time Series. In *KDD*. 359–370.

[2] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *IEEE S&P*. 141–159.

[3] Tianci Bu, Le Zhou, Wenchuan Yang, Jianhong Mou, Kang Yang, Suoyi Tan, Feng Yao, Jingyuan Wang, and Xin Lu. 2025. ProDiff: Prototype-Guided Diffusion for Minimal Information Trajectory Imputation. In *ICML*. 18.

[4] Weiquan Cao and Yunzhao Li. 2017. DOTS: An online and near-optimal trajectory simplification algorithm. *Journal of Systems and Software* 126 (2017), 34–44.

[5] Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *IEEE S&P*. 463–480.

[6] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: evaluating and testing unintended memorization in neural networks. In *USENIX Security*. 267–284.

[7] Bing-Jyue Chen, Chiok Yew Ho, and De-Nian Yang. 2024. AFTER: Adaptive Friend Discovery for Temporal-Spatial and Social-Aware XR. In *ICDE*. 2639–2652.

[8] Minxiao Chen, Haitao Yuan, Nan Jiang, Zhihan Zheng, Sai Wu, Ao Zhou, and Shangguang Wang. 2025. RLOMM: An Efficient and Robust Online Map Matching Framework with Reinforcement Learning. *Proc. ACM Manag. Data* 3, 3, Article 209 (2025), 26 pages.

[9] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2022. Graph Unlearning. In *CCS*. 499–513.

[10] Yile Chen, Gao Cong, and Cuauhtemoc Anda. 2023. TERI: An Effective Framework for Trajectory Recovery with Irregular Time Intervals. *Proc. VLDB Endow.* 17, 3 (2023), 414–426.

[11] Yuqi Chen, Hanyuan Zhang, Weiwei Sun, and Baihua Zheng. 2023. RNTrajRec: Road Network Enhanced Trajectory Recovery with Spatial-Temporal Transformer. In *ICDE*. 829–842.

[12] Ziwen Chen, Ke Li, Silin Zhou, Lisi Chen, and Shuo Shang. 2022. Towards robust trajectory similarity computation: Representation-based spatio-temporal similarity quantification. *WWW* 26, 4 (2022), 1271–1294.

[13] Zhen Chen, Dalin Zhang, Shanshan Feng, Kaixuan Chen, Lisi Chen, Peng Han, and Shuo Shang. 2024. KGTS: Contrastive Trajectory Similarity Learning over Prompt Knowledge Graph Embedding. *AAAI* 38, 8 (2024), 8311–8319.

[14] Dasol Choi and Dongbin Na. 2025. Distribution-level feature distancing for machine unlearning: towards a better trade-off between model utility and forgetting. In *AAAI*. 2536–2544.

[15] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. 2023. Can Bad Teaching Induce Forgetting? Unlearning in Deep Networks Using an Incompetent Teacher. *AAAI* 37, 6 (2023), 7210–7217.

[16] Yizhou Dang, Yuting Liu, Enneng Yang, Guibing Guo, Linying Jiang, Jianzhe Zhao, and Xingwei Wang. 2025. Efficient and Adaptive Recommendation Unlearning: A Guided Filtering Framework to Erase Outdated Preferences. *ACM Trans. Inf. Syst.* 43, 2, Article 49 (2025), 25 pages.

[17] DiDi. 2016. https://outreach.didichuxing.com/en/

[18] European Parliament and Council of the European Union. [n.d.]. *Regulation (EU) 2016/679 of the European Parliament and of the Council.* https://data.europa.eu/eli/reg/2016/679/oj

[19] Ziquan Fang, Yuntao Du, Xinjun Zhu, Danlei Hu, Lu Chen, Yunjun Gao, and Christian S. Jensen. 2022. Spatio-Temporal Trajectory Similarity Learning in Road Networks. In *KDD*. 347–356.

[20] Ziquan Fang, Changhao He, Lu Chen, Danlei Hu, Qichen Sun, Linsen Li, and Yunjun Gao. 2023. A Lightweight Framework for Fast Trajectory Simplification. In *ICDE*. 2386–2399.

[21] Chunhui Feng, Zhicheng Pan, Junhua Fang, Jiajie Xu, Pengpeng Zhao, and Lei Zhao. 2022. Aries: Accurate Metric-based Representation Learning for Fast Top-k Trajectory Similarity Query. In *CIKM*. 499–508.

[22] Jie Feng, Yong Li, Kai Zhao, Zhao Xu, Tong Xia, Jinglin Zhang, and Depeng Jin. 2022. DeepMM: Deep Learning Based Map Matching With Data Augmentation. *TMC* 21, 7 (2022), 2372–2384.

[23] Jack Foster, Stefan Schoepf, and Alexandra Brintrup. 2024. Fast Machine Unlearning without Retraining through Selective Synaptic Dampening. *AAAI* 38 (2024), 12043–12051.

[24] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2020. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *CVPR*. 9304–9312.

[25] Chuan Guo, Tom Goldstein, Awni Y. Hannun, and Laurens van der Maaten. 2020. Certified Data Removal from Machine Learning Models. In *ICML*, Vol. 119. 3832–3842.

[26] Yang Guo, Zhiqi Wang, Jin Xue, and Zili Shao. 2024. A Spatio-Temporal Series Data Model with Efficient Indexing and Layout for Cloud-Based Trajectory Data Management. In *ICDE*. 1171–1184.

[27] Chaowei He, Yuanjun Liu, Qingzhi Ma, Shenyuan Ren, Xizhao Luo, Lei Zhao, and An Liu. 2026. Forgetting by Pruning: Data Deletion in Join Cardinality Estimation. In *AAAI*. 9. https://arxiv.org/abs/2511.20293

[28] John Hershberger and Jack Snoeyink. 1992. Speeding Up the Douglas-Peucker Line-Simplification Algorithm. In *SDH*. University of British Columbia, 134–143.

[29] Mark He Huang, Lin Geng Foo, and Jun Liu. 2024. Learning to Unlearn for Robust Machine Unlearning. In *ECCV*. 202–219.

[30] Zhehao Huang, Xinwen Cheng, JingHao Zheng, Haoran Wang, Zhengbao He, Tao Li, and Xiaolin Huang. 2024. Unified Gradient-Based Machine Unlearning with Remain Geometry Enhancement. In *NIPS*, Vol. 37. 26377–26414.

[31] Chih-Chieh Hung, Wen-Chih Peng, and Wang-Chien Lee. 2015. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. *The VLDB Journal* 24, 2 (2015), 169–192.

[32] 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 *ICDE*. IEEE.

[33] Bingqing Ke, Jie Shao, Yi Zhang, Dongxiang Zhang, and Yang Yang. 2016. An Online Approach for Direction-Based Trajectory Compression with Error Bound Guarantee. In *APWeb*. 79–91.

[34] Korbinian Koch and Marcus Soll. 2023. No matter how you slice it: Machine unlearning with sisa comes at the expense of minority classes. In *SaTML*. IEEE, 622–637.

[35] Meghdad Kurmanji, Eleni Triantafillou, and Peter Triantafillou. 2024. Machine Unlearning in Learned Databases: An Experimental Analysis. *Proc. ACM Manag. Data* 2, 1, Article 49 (2024), 26 pages.

[36] Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. 2023. Towards Unbounded Machine Unlearning. In *NIPS*, Vol. 36. 1957–1987.

[37] Scott T. Leutenegger, Jeffrey M. Edgington, and Mario A. Lopez. 1997. STR: a simple and efficient algorithm for R-tree packing. In *ICDE*. 497–506.

[38] Guanyao Li, Chih-Chieh Hung, Mengyun Liu, Linfei Pan, Wen-Chih Peng, and S.-H. Gary Chan. 2021. Spatial-Temporal Similarity for Trajectories with Location Noise and Sporadic Sampling. In *ICDE*. 1224–1235.

[39] Jiaqi Li, Qianshan Wei, Chuanyi Zhang, Guilin Qi, Miaozeng Du, Yongrui Chen, Sheng Bi, and Fan Liu. 2024. Single image unlearning: efficient machine unlearning in multimodal large language models. In *NIPS*. Article 1116, 40 pages.

[40] Ruiyuan Li, Huajun He, Rubin Wang, Yuchuan Huang, Junwen Liu, Sijie Ruan, Tianfu He, Jie Bao, and Yu Zheng. 2020. JUST: JD Urban Spatio-Temporal Data Engine. In *ICDE*. 1558–1569.

[41] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. 2018. Deep Representation Learning for Trajectory Similarity Computation. In *ICDE*. 617–628.

[42] Biwei Liang, Tengjiao Wang, Shun Li, Wei Chen, Hongyan Li, and Kai Lei. 2016. Online Learning for Accurate Real-Time Map Matching. In *PAKDD*. 67–78.

[43] Shen Lin, Xiaoyu Zhang, Willy Susilo, Xiaofeng Chen, and Jun Liu. 2024. GDR-GMA: Machine Unlearning via Direction-Rectified and Magnitude-Adjusted Gradients. In *MM*. 9087–9095.

[44] Yan Lin, Huaiyu Wan, Shengnan Guo, Jilin Hu, Christian S. Jensen, and Youfang Lin. 2024. Pre-Training General Trajectory Embeddings With Maximum Multi-View Entropy Coding. *TKDE* 36, 12 (2024), 9037–9050.

[45] An Liu, Yifan Zhang, Xiangliang Zhang, Guanfeng Liu, Yanan Zhang, Zhixu Li, Lei Zhao, Qing Li, and Xiaofang Zhou. 2022. Representation Learning With Multi-Level Attention for Activity Trajectory Similarity Computation. *TKDE* 34, 5 (2022), 2387–2400.

[46] Yu Liu, Qian Ge, Wei Luo, Qiang Huang, Lei Zou, Haixu Wang, Xin Li, and Chang Liu. 2024. GraphMM: Graph-Based Vehicular Map Matching by Leveraging Trajectory and Road Correlations. *TKDE* 36, 1 (2024), 184–198.

[47] Yu Liu, Qian Ge, Wei Luo, Qiang Huang, Lei Zou, Haixu Wang, Xin Li, and Chang Liu. 2024. GraphMM: Graph-Based Vehicular Map Matching by Leveraging Trajectory and Road Correlations. *TKDE* 36, 1 (2024), 184–198.

[48] Yuanjun Liu, Guanfeng Liu, Qingzhi Ma, Zhixu Li, Lei Zhao, and An Liu. 2025. GPE: Global Position Embedding for Trajectory Similarity Computation. In *KDD*. 1927–1938.

[49] Ziqiao Liu, Hao Miao, Yan Zhao, Chenxi Liu, Kai Zheng, and Huan Li. 2024. LightTR: A Lightweight Framework for Federated Trajectory Recovery. In *ICDE*. 4422–4434.

[50] Cheng Long, Raymond Chi-Wing Wong, and H. V. Jagadish. 2014. Trajectory simplification: on minimizing the direction-based error. *Proc. VLDB Endow.* 8, 1 (Sept. 2014), 49–60.

[51] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *GIS*. 352–361.

[52] Zhipeng Ma, Zheyan Tu, Xinhai Chen, Yan Zhang, Deguo Xia, Guyue Zhou, Yilun Chen, Yu Zheng, and Jiangtao Gong. 2024. More Than Routing: Joint GPS and Route Modeling for Refine Trajectory Representation Learning. In *WWW*. 3064–3075.

[53] P. F. Marteau and G. Ménier. 2009. Speeding up simplification of polygonal curves using nested approximations. *Pattern Analysis and Applications* 12, 4 (2009), 367–375.

[54] Nirvana Meratnia and Rolf A. de By. 2004. Spatiotemporal Compression Techniques for Moving Point Objects. In *EDBT*. 765–782.

[55] Reham Mohamed, Heba Aly, and Moustafa Youssef. 2017. Accurate Real-time Map Matching for Challenging Environments. *TITS* 18, 4 (2017), 847–857.

[56] Luis Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luis Damas. 2013. Predicting Taxi–Passenger Demand Using Streaming Data.

*TITS* 14, 3 (2013), 1393–1402.
[57] Jonathan Muckell, Paul W. Olsen Jr., Jeong-Hyon Hwang, Catherine T. Lawson, and S. S. Ravi. 2014. Compression of trajectory data: a comprehensive evaluation and new approach. *GeoInformatica* 18, 3 (2014), 435–460.
[58] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *GIS*. 336–343.
[59] Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. 2020. Variational bayesian unlearning. *Advances in Neural Information Processing Systems* 33 (2020), 16025–16036.
[60] M. Potamias, K. Patroumpas, and T. Sellis. 2006. Sampling Trajectory Streams with Spatiotemporal Criteria. In *SSDBM*. 275–284.
[61] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. DITA: Distributed In-Memory Trajectory Analytics. In *SIGMOD*. 725–740.
[62] Zhihao Shen, Kang Yang, Xi Zhao, Jianhua Zou, Wan Du, and Junjie Wu. 2024. DMM: A Deep Reinforcement Learning Based Map Matching Framework for Cellular Data. *TKDE* 36, 10 (2024), 5120–5137.
[63] Weijie Shi, Jiajie Xu, Junhua Fang, Pingfu Chao, An Liu, and Xiaofang Zhou. 2023. LHMM: A Learning Enhanced HMM Model for Cellular Trajectory Map Matching. In *ICDE*. 2429–2442.
[64] Jianing Si, Haitao Yuan, Nan Jiang, Minxiao Chen, Xiao Ma, and Shangguang Wang. 2025. Towards Robust Trajectory Embedding for Similarity Computation: When Triangle Inequality Violations in Distance Metrics Matter. In *ICDE*. 1–14.
[65] Yumeng Song, Yu Gu, Tianyi Li, Yushuai Li, Christian S. Jensen, and Ge Yu. 2024. Quantifying Point Contributions: A Lightweight Framework for Efficient and Effective Query-Driven Trajectory Simplification. *Proc. VLDB Endow.* 18, 2 (2024), 453–465.
[66] Ayush K. Tarun, Vikram S. Chundawat, Murari Mandal, and Mohan Kankanhalli. 2024. Fast Yet Effective Machine Unlearning. *TNNLS* 35, 9 (2024), 13046–13055.
[67] Wei Tian, Jieming Shi, and Man Lung Yiu. 2025. Efficient Methods for Accurate Sparse Trajectory Recovery and Map Matching. In *ICDE*. 363–375.
[68] Cheng-Long Wang, Qi Li, Zihang Xiang, Yinzhi Cao, and Di Wang. 2025. *Towards lifecycle unlearning commitment management: measuring sample-level unlearning completeness.*
[69] Weiqi Wang, Zhiyi Tian, Chenhan Zhang, and Shui Yu. 2025. SCU: An Efficient Machine Unlearning Scheme for Deep Learning Enabled Semantic Communications. *TIFS* 20 (2025), 547–558.
[70] Weiqi Wang, Chenhan Zhang, Zhiyi Tian, and Shui Yu. 2025. FedU: Federated Unlearning via User-Side Influence Approximation Forgetting. *TDSC* 22, 3 (2025), 2550–2562.
[71] Zheng Wang, Cheng Long, and Gao Cong. 2021. Trajectory Simplification with Reinforcement Learning. In *ICDE*. 684–695.
[72] Zheng Wang, Cheng Long, Gao Cong, and Christian S. Jensen. 2024. Collectively Simplifying Trajectories in a Database: A Query Accuracy Driven Approach. In *ICDE*. 4383–4395.
[73] Zheng Wang, Cheng Long, Gao Cong, and Qianru Zhang. 2021. Error-Bounded Online Trajectory Simplification with Multi-Agent Reinforcement Learning. In *KDD*. 1758–1768.
[74] Dongen Wu, Ziquan Fang, Qichen Sun, Lu Chen, Haiyang Hu, Fei Wang, and Yunjun Gao. 2024. TrajRecovery: An Efficient Vehicle Trajectory Recovery Framework based on Urban-Scale Traffic Camera Records. In *KDD*. 5979–5990.
[75] Leijie Wu, Song Guo, Junxiao Wang, Zicong Hong, Jie Zhang, and Yaohong Ding. 2022. Federated Unlearning: Guarantee the Right of Clients to Forget. *IEEE Network* 36, 5 (2022), 129–135.
[76] Peilun Yang, Hanchen Wang, Defu Lian, Ying Zhang, Lu Qin, and Wenjie Zhang. 2022. TMN: Trajectory Matching Networks for Predicting Similarity. In *ICDE*. 1700–1713.
[77] 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 *ICDE*. 1358–1369.
[78] Di Yao, Haonan Hu, Lun Du, Gao Cong, Shi Han, and Jingping Bi. 2022. TrajGAT: A Graph-based Long-term Dependency Modeling Approach for Trajectory Similarity Computation. In *KDD*. 2275–2285.
[79] Jin Yao, Eli Chien, Minxin Du, Xinyao Niu, Tianhao Wang, Zezhou Cheng, and Xiang Yue. 2024. Machine Unlearning of Pre-trained Large Language Models. In *ACL*. 8403–8419.
[80] Shanshan Ye and Jie Lu. 2024. Sequence Unlearning for Sequential Recommender Systems. In *Advances in Artificial Intelligence*. 403–415.
[81] Shanshan Ye, Jie Lu, and Guangquan Zhang. 2025. Towards Safe Machine Unlearning: A Paradigm that Mitigates Performance Degradation. In *WWW*. 4635–4652.
[82] Fudan Yu, Wenxuan Ao, Huan Yan, Guozhen Zhang, Wei Wu, and Yong Li. 2022. Spatio-Temporal Vehicle Trajectory Recovery on Road Network Based on Traffic Camera Video Data. In *KDD*. 4413–4421.
[83] Liu Yuanjun, Liu An, Liu Guanfeng, Li Zhixu, and Zhao Lei. 2023. Towards Effective Trajectory Similarity Measure in Linear Time. In *DASFAA*. 283–299.
[84] Chenhan Zhang, Weiqi Wang, Zhiyi Tian, and Shui Yu. 2024. Forgetting and Remembering Are Both You Need: Balanced Graph Structure Unlearning. *TIFS* 19 (2024), 6751–6763.
[85] Yupu Zhang, Liwei Deng, Yan Zhao, Jin Chen, Jiandong Xie, and Kai Zheng. 2023. SimiDTR: Deep Trajectory Recovery with Enhanced Trajectory Similarity. In *DASFAA*. 431–447.
[86] Silin Zhou, Yao Chen, Shuo Shang, Lisi Chen, Bingsheng He, and Ryosuke Shibasaki. 2025. Blurred Encoding for Trajectory Representation Learning. In *KDD*. 4132–4143.
[87] Silin Zhou, Peng Han, Di Yao, Lisi Chen, and Xiangliang Zhang. 2023. Spatial-temporal fusion graph framework for trajectory similarity computation. *WWW* 26, 4 (2023), 1501–1523.
[88] Silin Zhou, Jing Li, Hao Wang, Shuo Shang, and Peng Han. 2023. GRLSTM: trajectory similarity computation with graph-based residual LSTM. In *AAAI*, Vol. 37. 4972–4980.
[89] Silin Zhou, Shuo Shang, Lisi Chen, Peng Han, and Christian S. Jensen. 2025. Grid and Road Expressions Are Complementary for Trajectory Representation Learning. In *KDD*. 2135–2146.
[90] Zhiwei Zuo, Zhuo Tang, Kenli Li, and Anwitaman Datta. 2025. Machine Unlearning Through Fine-Grained Model Parameters Perturbation. *TKDE* 37, 4 (2025), 1975–1988.

# A EXPERIMENTAL RESULTS ON FOUR TRAJECTORY LEARNING TASKS

## A.1 Results of Similarity Task

Table 4 shows the results of similarity tasks. GDRGMA and RandomK are optimal, and NegGrad and TopK algorithms are the worst. The NegGrad algorithm simply maximizes the loss on the unlearning set, which leads to catastrophic forgetting. The TopK algorithm first perturbs parameters important to the unlearning set, ignoring that those parameters may also be important to the remaining set.

Most algorithms have similar MIA results to Retrain in the Area scenario, demonstrating that those unlearning algorithms succeeded. Nevertheless, all algorithms have similar MIA results in the User Scenario. Since the distributions of the remaining set $D_r$ and the unlearning set $D_u$ are similar, $D_r$ contains the information about $D_u$, differentiating it from the Area scenario.

## A.2 Results of Simplification Task

Table 5 shows the results of the simplification task. RandomK and TopK perform best, and SCRUB and SFRon are the worst.

The algorithms show lower SED results on the BJ and Xian datasets than the Porto dataset, indicating that the trajectories of the both datasets are easier to simplify. The reason is that the shapes of the road networks of them are more regular than Porto, making the trajectories of them simpler to simplify. For the Porto dataset, the Area scenario saw lower SED values on the unlearning set $D_u$ than the User scenario, and higher SED values on the remaining set $D_r$ than the User scenario. This is because the trajectories of the unlearning set are at the edge of Porto, where the road network has fewer corners, making them simpler to simplify than the trajectories of the unlearning set in the User scenario. On the contrary, the left trajectories of the remaining set of the Area scenario are harder to simplify than those of the User scenario. Similarly, for the BJ dataset, the road network at the center is more regular, making trajectories at the center simpler to simplify than those at the edge. And the left trajectories of the remaining set of the Area scenario are easier to simplify than those of the User scenario. Thus, the User scenario saw lower SED values on the unlearning set $D_u$ than the Area scenario, and higher SED values on the remaining set $D_r$ than the Area scenario.

The MIA results of the simplification task are much higher than those of other tasks, i.e., MIAs of other tasks are generally less than 0.6. The reason is that the MLSimp model employs mutual

**Table 4: MR and MIA Results of Similarity Tasks**

| Method | Unlearn User on Porto Dataset | | | | Unlearn Area on Porto Dataset | | | | Unlearn User on BJ Dataset | | | | Unlearn Area on BJ Dataset | | | | Unlearn User on Xian Dataset | | | | Unlearn Area on Xian Dataset | | | | Rank by SimScore |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | |
| Retrain | 1.717 | 1.751 | 2.196 | 0.503 | 1.518 | 1.581 | 2.490 | 0.150 | 7.589 | 12.365 | 9.105 | 0.502 | 3.753 | 8.556 | 6.793 | 0.066 | 4.007 | 4.952 | 4.817 | 0.501 | 3.085 | 4.281 | 4.096 | 0.090 | - |
| FineTune | 1.343 | 1.609 | 1.905 | 0.500 | **1.613** | 1.444 | 1.886 | **0.151** | 21.618 | 36.758 | 35.256 | 0.500 | 28.372 | 49.129 | 45.310 | 0.148 | 2.162 | 2.531 | 2.550 | **0.502** | 2.240 | 2.648 | 2.550 | 0.081 | 3 |
| NegGrad | 4542.661 | 6516.504 | 6580.751 | 0.498 | 4822.099 | 5986.638 | 6299.869 | 0.497 | 3891.439 | 5745.600 | 5700.235 | 0.498 | 3953.140 | 5587.173 | 5650.383 | 0.498 | 8290.668 | 11953.552 | 11882.935 | 0.498 | 8742.459 | 10773.216 | 11059.889 | 0.497 | 9 |
| BadT | 3.710 | 4.783 | 5.862 | **0.504** | 3.949 | 4.013 | 6.095 | 0.142 | 28.777 | 47.973 | 43.370 | **0.503** | 30.214 | 48.968 | 44.278 | 0.088 | 5.453 | 6.764 | 6.950 | 0.503 | 5.350 | 7.176 | 6.900 | 0.083 | 6 |
| SCRUB | 2.170 | 2.780 | 3.396 | 0.499 | 2.311 | 2.410 | 4.134 | 0.136 | 35.657 | 60.850 | 57.691 | 0.499 | 122.533 | 74.900 | 95.748 | 0.043 | **5.312** | **6.414** | **6.702** | 0.499 | 6.167 | **5.530** | 5.822 | 0.100 | 5 |
| GDRGMA | 1.401 | 1.719 | 2.096 | 0.499 | 1.780 | **1.578** | 2.216 | 0.145 | 15.203 | 23.790 | 20.636 | 0.500 | **15.242** | **24.873** | **21.616** | 0.050 | 2.526 | 2.957 | 2.991 | 0.500 | 2.573 | 3.085 | 2.963 | 0.079 | **1** |
| TopK | 3372.848 | 4867.141 | 4906.504 | 0.499 | 3592.467 | 5268.983 | 5394.047 | 0.457 | 4227.956 | 6204.303 | 6191.668 | 0.498 | 1998.442 | 2912.781 | 2980.895 | 0.382 | 3097.600 | 4375.700 | 4360.883 | 0.499 | 3974.128 | 4365.109 | 4665.564 | 0.289 | 8 |
| RandomK | **1.544** | 1.885 | 2.468 | 0.499 | 2.007 | 1.711 | **2.572** | 0.120 | 16.281 | 24.027 | 21.233 | 0.500 | 15.460 | 26.460 | 21.992 | 0.039 | 2.793 | 3.253 | 3.245 | 0.499 | **3.108** | 3.221 | **3.151** | 0.094 | 2 |
| SFRon | 2.333 | 3.134 | 3.837 | 0.499 | 207.843 | 71.638 | 134.550 | 0.118 | 15.310 | 24.003 | 20.773 | 0.499 | 17.804 | 30.174 | 25.935 | **0.062** | 12.043 | 15.609 | 15.937 | 0.499 | 135.508 | 26.542 | 41.769 | 0.052 | 7 |
| SSD | 1.450 | **1.745** | **2.147** | 0.492 | 1.797 | 1.556 | 2.223 | 0.145 | **14.983** | **23.335** | **20.278** | 0.498 | 15.403 | 25.014 | 21.734 | 0.122 | 2.529 | 2.961 | 2.997 | 0.498 | 457.399 | 848.894 | 764.796 | **0.087** | 4 |

**Table 5: SED and MIA Results of Simplification Tasks**

| Method | Unlearn User on Porto Dataset | | | | Unlearn Area on Porto Dataset | | | | Unlearn User on BJ Dataset | | | | Unlearn Area on BJ Dataset | | | | Unlearn User on Xian Dataset | | | | Unlearn Area on Xian Dataset | | | | Rank by SimScore |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | |
| Retrain | 0.037 | 0.039 | 0.037 | 0.762 | 0.037 | 0.040 | 0.038 | 0.613 | 0.009 | 0.014 | 0.014 | 0.683 | 0.012 | 0.013 | 0.014 | 0.242 | 0.018 | 0.018 | 0.020 | 0.533 | 0.016 | 0.019 | 0.021 | 0.371 | - |
| FineTune | 0.039 | 0.039 | **0.037** | 0.849 | 0.037 | 0.041 | 0.038 | 0.595 | 0.009 | 0.014 | 0.014 | 0.502 | 0.012 | 0.013 | 0.014 | **0.288** | 0.018 | 0.019 | 0.021 | **0.602** | **0.016** | 0.019 | 0.021 | 0.286 | 5 |
| NegGrad | 0.038 | 0.039 | 0.038 | 0.787 | 0.037 | 0.041 | 0.038 | 0.596 | 0.008 | 0.014 | 0.014 | 0.556 | **0.012** | **0.013** | 0.014 | 0.344 | 0.018 | 0.019 | 0.021 | 0.673 | 0.017 | 0.019 | 0.021 | 0.353 | 4 |
| BadT | 0.038 | 0.040 | 0.038 | 0.805 | 0.037 | 0.041 | 0.038 | 0.627 | 0.009 | 0.014 | **0.014** | 0.551 | 0.012 | 0.013 | 0.014 | 0.380 | 0.018 | 0.019 | 0.021 | 0.623 | 0.017 | 0.019 | 0.021 | 0.454 | 7 |
| SCRUB | 0.038 | 0.039 | 0.037 | 0.825 | 0.037 | 0.041 | 0.038 | 0.664 | **0.009** | 0.014 | 0.014 | 0.660 | 0.012 | 0.013 | 0.014 | 0.356 | **0.018** | 0.018 | 0.021 | 0.742 | 0.017 | 0.019 | 0.021 | 0.510 | 8 |
| GDRGMA | 0.038 | 0.039 | 0.038 | 0.864 | **0.037** | 0.041 | 0.037 | 0.724 | 0.009 | 0.014 | 0.014 | 0.606 | 0.012 | 0.013 | 0.014 | 0.325 | 0.017 | 0.019 | 0.021 | 0.679 | 0.017 | 0.019 | 0.021 | 0.401 | 6 |
| TopK | **0.038** | **0.039** | 0.038 | 0.755 | 0.037 | 0.040 | 0.038 | 0.522 | 0.009 | 0.014 | 0.014 | 0.529 | 0.012 | 0.013 | 0.014 | 0.323 | 0.018 | 0.018 | 0.021 | 0.622 | 0.016 | 0.019 | **0.021** | 0.334 | 2 |
| RandomK | 0.038 | 0.039 | 0.038 | **0.767** | 0.037 | 0.041 | 0.038 | **0.606** | 0.009 | 0.014 | 0.014 | **0.674** | 0.012 | 0.013 | 0.014 | 0.294 | 0.018 | 0.018 | 0.021 | 0.685 | 0.017 | **0.019** | 0.021 | 0.467 | **1** |
| SFRon | 0.038 | 0.039 | 0.038 | 0.821 | 0.037 | **0.040** | 0.038 | 0.721 | 0.009 | 0.014 | 0.014 | 0.487 | 0.012 | 0.013 | 0.014 | 0.367 | 0.018 | **0.018** | **0.021** | 0.697 | 0.017 | 0.019 | 0.021 | **0.374** | 9 |
| SSD | 0.038 | 0.039 | 0.038 | 0.743 | 0.037 | 0.041 | **0.038** | 0.575 | 0.009 | **0.014** | 0.014 | 0.600 | 0.012 | 0.013 | **0.014** | 0.345 | 0.018 | 0.018 | 0.021 | 0.725 | 0.017 | 0.019 | 0.021 | 0.329 | 3 |

**Table 6: F1 and MIA Results of Map Matching Tasks**

| Method | Unlearn User on Porto Dataset | | | | Unlearn Area on Porto Dataset | | | | Unlearn User on BJ Dataset | | | | Unlearn Area on BJ Dataset | | | | Unlearn User on Xian Dataset | | | | Unlearn Area on Xian Dataset | | | | Rank by SimScore |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | |
| Retrain | 0.869 | 0.877 | 0.870 | 0.502 | 0.855 | 0.881 | 0.869 | 0.499 | 0.825 | 0.844 | 0.826 | 0.498 | 0.800 | 0.844 | 0.824 | 0.497 | 0.962 | 0.964 | 0.961 | 0.498 | 0.931 | 0.965 | 0.956 | 0.500 | - |
| FineTune | **0.884** | 0.888 | **0.885** | 0.501 | 0.884 | 0.888 | **0.885** | 0.505 | 0.833 | **0.841** | 0.833 | 0.500 | 0.824 | **0.841** | 0.833 | 0.501 | 0.971 | 0.972 | 0.968 | **0.499** | 0.961 | 0.970 | 0.967 | 0.499 | **1** |
| NegGrad | 0.109 | 0.114 | 0.112 | 0.501 | 0.109 | 0.109 | 0.109 | 0.501 | 0.707 | 0.705 | 0.703 | 0.502 | 0.306 | 0.300 | 0.301 | 0.502 | 0.810 | 0.809 | 0.809 | 0.500 | 0.813 | 0.809 | 0.809 | 0.498 | 9 |
| BadT | 0.885 | 0.893 | 0.886 | 0.498 | **0.882** | 0.893 | 0.886 | 0.497 | 0.833 | 0.851 | 0.834 | 0.499 | 0.820 | 0.849 | 0.832 | 0.500 | 0.970 | 0.972 | 0.967 | 0.501 | 0.956 | 0.971 | 0.966 | 0.501 | 2 |
| SCRUB | 0.678 | 0.682 | 0.680 | **0.502** | 0.613 | 0.734 | 0.722 | 0.497 | 0.530 | 0.535 | 0.532 | **0.499** | 0.707 | 0.745 | 0.739 | 0.461 | 0.859 | 0.859 | 0.859 | 0.499 | 0.816 | 0.855 | 0.852 | 0.472 | 8 |
| GDRGMA | 0.907 | 0.908 | 0.907 | 0.501 | 0.908 | 0.907 | 0.907 | **0.499** | 0.831 | 0.832 | 0.830 | 0.501 | 0.823 | 0.832 | 0.830 | 0.497 | 0.976 | 0.975 | 0.969 | 0.503 | 0.967 | 0.970 | 0.969 | 0.503 | 5 |
| TopK | 0.885 | **0.887** | 0.886 | 0.501 | 0.893 | **0.887** | 0.887 | 0.498 | 0.832 | 0.834 | 0.832 | 0.502 | 0.826 | 0.835 | 0.832 | 0.502 | **0.970** | **0.969** | **0.966** | 0.499 | 0.962 | 0.967 | 0.966 | 0.497 | 3 |
| RandomK | 0.890 | 0.892 | 0.891 | 0.499 | 0.898 | 0.892 | 0.893 | 0.501 | 0.834 | 0.836 | 0.833 | 0.500 | 0.828 | 0.836 | 0.834 | **0.497** | 0.974 | 0.973 | 0.969 | 0.499 | 0.966 | 0.970 | 0.969 | **0.500** | 4 |
| SFRon | 0.763 | 0.779 | 0.775 | 0.501 | 0.625 | 0.823 | 0.789 | 0.465 | 0.586 | 0.614 | 0.606 | 0.502 | 0.553 | 0.825 | 0.784 | 0.386 | 0.839 | 0.841 | 0.841 | 0.499 | 0.821 | 0.935 | 0.913 | 0.439 | 7 |
| SSD | 0.907 | 0.908 | 0.906 | 0.500 | 0.885 | 0.906 | 0.902 | 0.496 | **0.831** | 0.832 | **0.830** | 0.501 | **0.819** | 0.831 | **0.829** | 0.487 | 0.975 | 0.974 | 0.968 | 0.501 | **0.916** | **0.966** | **0.958** | 0.463 | 6 |

**Table 7: MAE and MIA Results of Recovery Tasks**

| Method | Unlearn User on Porto Dataset | | | | Unlearn Area on Porto Dataset | | | | Unlearn User on BJ Dataset | | | | Unlearn Area on BJ Dataset | | | | Unlearn User on Xian Dataset | | | | Unlearn Area on Xian Dataset | | | | Rank by SimScore |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | Du | Dr | Dv | MIA | |
| Retrain | 103.388 | 99.794 | 101.868 | 0.484 | 89.386 | 104.466 | 102.280 | 0.111 | 258.491 | 235.162 | 256.887 | 0.507 | 277.150 | 240.946 | 260.983 | 0.067 | 77.747 | 76.905 | 77.658 | 0.486 | 81.056 | 77.460 | 78.047 | 0.096 | - |
| FineTune | 121.615 | 118.513 | 119.811 | 0.497 | 98.051 | 118.671 | 114.815 | 0.153 | **258.069** | 249.156 | **256.011** | 0.504 | 261.315 | 248.881 | 254.769 | 0.045 | 110.241 | 110.365 | 110.522 | 0.519 | 101.991 | 106.030 | 104.784 | 0.065 | 5 |
| NegGrad | 449.620 | 437.665 | 440.007 | 0.457 | 327.696 | 386.406 | 372.912 | 0.155 | 641.259 | 637.107 | 642.472 | 0.495 | 796.075 | 778.378 | 786.320 | 0.121 | 354.962 | 356.511 | 355.017 | 0.516 | 366.596 | 389.652 | 383.500 | 0.105 | 9 |
| BadT | 119.339 | 115.986 | 117.464 | 0.507 | 101.436 | 121.156 | 117.757 | 0.099 | 259.922 | **243.800** | 257.990 | 0.523 | 268.619 | **247.311** | **259.431** | **0.073** | 107.926 | 107.773 | 108.119 | 0.503 | 109.375 | 114.737 | 113.468 | **0.092** | 4 |
| SCRUB | 378.778 | 368.279 | 371.362 | 0.492 | 363.217 | 444.302 | 424.813 | **0.110** | 640.977 | 636.875 | 642.410 | 0.513 | 695.108 | 666.477 | 675.755 | 0.108 | 332.111 | 331.242 | 330.343 | 0.544 | 337.108 | 348.069 | 341.715 | 0.078 | 8 |
| GDRGMA | 102.437 | **99.922** | 100.781 | 0.499 | 85.659 | **104.501** | **100.781** | 0.195 | 256.079 | 253.800 | 254.577 | **0.505** | 260.100 | 254.366 | 254.577 | 0.045 | 76.680 | 76.351 | 76.964 | **0.482** | **76.565** | **77.639** | **76.964** | 0.063 | **1** |
| TopK | 115.444 | 112.280 | 113.235 | **0.491** | 99.264 | 116.774 | 113.229 | 0.151 | 264.814 | 261.254 | 263.323 | 0.509 | 270.359 | 263.336 | 264.833 | 0.045 | 109.205 | 109.516 | 109.590 | 0.494 | 112.179 | 115.239 | 113.918 | 0.065 | 6 |
| RandomK | 109.098 | 106.371 | 107.112 | 0.514 | **90.633** | 109.704 | 105.848 | 0.081 | 260.560 | 257.319 | 259.066 | 0.492 | 264.314 | 257.210 | 258.368 | 0.087 | 94.748 | 94.643 | 95.031 | 0.496 | 94.253 | 98.437 | 96.994 | 0.085 | 3 |
| SFRon | 292.686 | 283.964 | 286.992 | 0.493 | 311.795 | 337.352 | 326.397 | 0.144 | 480.472 | 474.611 | 480.026 | 0.492 | 729.290 | 748.792 | 745.682 | 0.125 | 322.194 | 322.660 | 321.198 | 0.518 | 281.899 | 253.473 | 254.211 | 0.084 | 7 |
| SSD | **102.519** | 100.003 | **100.855** | 0.496 | 107.652 | 105.166 | 104.064 | 0.124 | 256.291 | 253.995 | 254.965 | 0.475 | **274.558** | 255.104 | 257.014 | 0.055 | **76.934** | **76.489** | **77.138** | 0.497 | 129.369 | 79.105 | 88.178 | 0.057 | 2 |

learning with multiple rounds. Since the GNN model is retrained in the last round, we treat preceding rounds as the pre-training of the diffusion model and the last round as the training of the GNN model. We executed unlearning on the GNN model, but the pre-trained diffusion model still contains the information about the unlearning set. Moreover, the grid embeddings pre-trained on the original dataset also contain the information about the unlearning set. However, unlearning algorithms are generally designed for a single training process, whether and how to unlearn pre-trained and mutual models remain open problems for the community.

## A.3 Results of Map Matching Task

Table 6 shows the results of the map-matching task. FineTune and BadT are good at this task, and NegGrad and SCRUB algorithms underperform obviously. The F1 values on the unlearning set $D_u$ of the Area scenario are lower than those of the User scenario, demonstrating that the Area scenario is easier to unlearn thanks to its distinguishable data distribution. The F1 values of the Porto and Xian datasets are higher than those of the BJ dataset, since the road networks of both are sparser and the distances between points with wrong segments are larger, which benefits map matching. The MIA results of all scenarios are around 0.5, indicating the difficulty of measuring the influence of data in the trajectory map matching task.

The characteristics of the map matching task can be attributed to the following reasons. First, the MMA model matches a point by choosing the optimal surrounding road segment whose embedding has the highest dot product value with the point embedding. The

road segment embeddings generated by the Node2Vec method are fixed and not unlearned, so the MMA model could maintain performance if the road network remains the same. Second, the MMA model is enhanced by the R-tree index to improve the probability of choosing the truth, so the unlearning effect is not as obvious as in other tasks. Third, while previous tasks focus on the whole trajectory, the map matching task pays more attention to individual points, thus restraining influence to individuals. Fourth, while both the image classification task and the map matching task use classification loss, the former classifies images into fixed labels, whereas MMA matches points to selected dynamic segments, i.e., different points have different nearby segments provided by the R-tree index, making unlearning inefficient.

## A.4 Results of Recovery Task

Table 7 shows the results of the recovery task. NegGrad and SCRUB perform worse, and GDRGMA and SSD are better. The MAE values of the Porto and Xian datasets are lower than BJ because the map matching accuracies of both are higher, and higher accuracy benefits recovery. Some algorithms saw dramatic increases in MAE values since recovery is based on linking matched segments with their shortest path on the road network. The trajectory changes intensely even if one segment is mismatched.

The MIA results are diverse in the User and Area scenarios. First, as mentioned before, the distributions of the remaining and unlearning sets in the User scenario are similar, the remaining set contains the information about the unlearning set, so the MIA values are about 0.5. However, the distributions are distinct in the Area scenario, so the MIAs of the Area scenario are lower than those of the User scenario. Second, the trajectory recovery task outputs the whole trajectory, which provides adequate information for MIA, while map matching focuses on specific points and surrounding segments, lacking the global view of the trajectory.