Szilvia Lestyán*, Gergely Ács, and Gergely Biczók

In Search of Lost Utility: Private Location Data

Abstract: The unavailability of training data is a permanent source of much frustration in research, especially when it is due to privacy concerns. This is particularly true for location data since previous techniques all suffer from the inherent sparseness and high dimensionality of location trajectories which render most techniques impractical, resulting in unrealistic traces and non-scalable methods. Moreover, time information of location visits is usually dropped, or its resolution is drastically reduced. In this paper we present a novel technique for privately releasing a composite generative model and whole high-dimensional location datasets with detailed time information. To generate high-fidelity synthetic data, we leverage several peculiarities of vehicular mobility such as its language-like characteristics ("you should know a *location* by the company it keeps") or how humans plan their trips from one point to the other. We model the generator distribution of the dataset by first constructing a variational autoencoder to generate the source and destination locations, and the corresponding timing of trajectories. Next, we compute transition probabilities between locations with a feed forward network, and build a transition graph from the output of this model, which approximates the distribution of all paths between the source and destination (at a given time). Finally, a path is sampled from this distribution with a Markov Chain Monte Carlo method. The generated synthetic dataset is highly realistic, scalable, provides good utility and, nonetheless, provably private. We evaluate our model against two state-of-the-art methods and three real-life datasets demonstrating the benefits of our approach.

Keywords: Location data anonymization, Differential Privacy, Generative Models

DOI 10.56553/popets-2022-0076
Received 2021-11-30; revised 2022-03-15; accepted 2022-03-16.

1 Introduction

Analyzing human mobility patterns has been in the focus of both researchers and practitioners in the last decades [22] [6] [12]. Having created an already $12 billion market, the location data industry is steadily expanding with companies that harvest, sell, or trade in location data. Many of these firms claim that privacy has utmost importance in their businesses and that they never sell personal data.[1]

However, collecting and mining location data come inherently with their own strong privacy and other ethical concerns [5]. Several studies have shown that pseudonymization and quasi-standard de-identification are not sufficient to prevent users from being re-identified in location datasets [11] [37]. This significantly hinders location data sharing and use by researchers, developers and humanitarian workers alike[2].

Although a plethora of different anonymization techniques have been proposed for location data (for a very thorough survey see [19]), they all suffer from either weak utility, or weak privacy guarantees, or they are not scalable to large datasets. Indeed, location data are inherently high-dimensional and often sparse, which makes an individual's location trajectory unique even in very large populations[3]. This has a detrimental effect on privacy, and also on utility due to the curse of dimensionality. Note that aggregation per se does not necessarily prevent such re-identification in practice [31, 36]. The brittleness of privacy guarantees ignited research towards location anonymization with provable privacy guarantees. So far, only a handful of prior works [7, 10, 24, 25] have addressed the off-line anonymization of complete location trajectories with formal privacy guarantees. Most of these approaches use some form of Differential Privacy [16], which has become the *de facto* privacy model recent years [2, 18, 33].

***Corresponding Author: Szilvia Lestyán:** Department of Networked Systems and Services,CrySyS Lab, Budapest University of Technology and Economics, E-mail: lestyan@crysys.hu
Gergely Ács: E-mail: acs@crysys.hu
Gergely Biczók: E-mail: biczok@crysys.hu

1 https://themarkup.org/privacy/2021/09/30/theres-a-multibillion-dollar-market-for-your-phones-location-data
2 https://www.economist.com/leaders/2014/10/23/call-for-help
3 Four data points—approximate places and times where an individual was present—have been demonstrated to be enough to uniquely re-identify 95% of the users in a dataset of 1.5 million users [11]

Unfortunately, off-line location anonymization with Differential Privacy often implies serious accuracy degradation which can make anonymized data useless in practice. In particular, most schemes follow the common three-steps approach to generate privacy preserving synthetic location traces: (1) finding a faithful generative model of the underlying data generating distribution, (2) adding noise to the training of this model in order to provide Differential Privacy, (3) generating synthetic location trajectories from the noisy generative model. Generally, inaccuracy stems from either using a sub-optimal generative model, or a model which is not sufficiently robust against the additional noise needed for Differential Privacy. Although a more complex model is capable of faithfully capturing the peculiarities of location data, it also requires larger perturbation for privacy owing to the increased number of its parameters. Moreover, complex models very often are not scalable to larger datasets with several hundreds of thousands of location trajectories [7, 25]. Finally, to the best of our knowledge, none of these differentially private anonymization approaches release the valuable fine-grained temporal characteristics of location visits.

In this paper, we propose a novel off-line anonymization scheme called DP-Loc for location data with strong Differential Privacy guarantees. Unlike prior works, we tackle high dimensional data modeling with generative neural networks (GNN) which have shown great promise recently. GNNs have the potential to automatically learn the general features of a location dataset including complex regularities such as the subtle and valuable correlation among different location visits as a function of time. The key of our approach is a novel decomposition of the generator model into a sequence of smaller models and a post-processing step, which are robust against perturbation needed for Differential Privacy, and therefore can be used to generate high-fidelity synthetic location trajectories. In particular, we first project all trajectories to a smaller set of frequent locations thereby reducing the dimensionality of the data. Then, a synthetic trace is created by first generating its source and destination along with time information, and then finding a path on a transition graph between these endpoints. Both the endpoint and graph generation are modelled by distinct neural networks scalable for being trained with differentially private gradient descent [1].

DP-Loc has several advantages. First, projecting traces to a smaller set of locations helps generalization and also increases robustness against the added noise. Second, separating endpoint from path generation preserves the length of trajectories more accurately compared to related work. Third, recently proposed advanced composition theorems of Differential Privacy [1] enable us to use differentially private neural networks with better utility instead of simple Markovian models [10, 24], and quantify the privacy guarantee of DP-Loc more accurately. Finally, neural networks are sufficiently flexible to model transition probabilities depending on the destination and time, which facilitates the generation of more realistic location traces. Figure 2 illustrates the power of DP-Loc, where the density of complete synthetic location trajectories produced by DP-Loc are compared to the original data and a state-of-the-art solution from [24].

Our specific contributions are as follows:

1. We propose a novel generative model, DP-Loc to generate realistic location trajectories with time information. A Variational Auto-Encoder (VAE) is used to generate the source and destination of a trace along with a single timestamp of the trace. Then, a feed-forward neural network is built to compute the transition probabilities between any two locations depending on the destination and time, which implicitly defines the distribution of all paths between the source and the destination at a given time. Finally, a realistic path is generated by sampling from this distribution with a Markov Chain Monte Carlo (MCMC) method.
2. Our composite generative model can be trained with differentially private gradient descent [1] on sensitive location data, and, therefore, can be used to synthesize the training data with formally proven privacy guarantees. Such synthetic training data can be shared for any purposes without violating individuals' privacy.
3. We evaluate our model on three real-life public mobility datasets, and demonstrate that the generated private synthetic data has higher utility compared to previous works.

2 Related Work

A prominent line of anonymization research uses the notion of Differential Privacy [14] which gives a privacy guarantee based on rigorous mathematical proofs. Some proposals based on synthetic data generation via machine learning, i.e., modeling the dataset through the underlying distributions of generating variables, do apply Differential Privacy specifically for location data [10, 25], but with significant shortcomings. He et

al.[25] (DPT) discretize raw GPS trajectories using hierarchical reference systems to capture individual movements at different speeds. They then propose an adaptive mechanism to select a small set of reference systems to construct prefix tree counts. Lastly, a direction-weighted sampling is applied to improve utility.

Chen et al. designed Ngram [10], a variable-length n-gram model that makes use of an exploration tree structure and Markovian assumptions, with Differential Privacy. Gursoy et al. [24] designed AdaTrace, a generative model with a four-phase synthesis process consisting of feature extraction, synopsis learning, differentially private noise injection and synthetic trace generation. Additionally, they provided defense against three ad-hoc privacy attacks. Bindschaedler and Shokri [7] (SGLT) enforce plausible deniability to generate privacy-preserving synthetic traces. SGLT first recommends trace similarity and intersection functions that map a synthetic trace to an original one under similarity and intersection constraints. Then, it generates one synthetic trace using one real trace as its seed. If the synthetic trace satisfies plausible deniability, i.e., there exist k other real traces that can be mapped to the synthetic trace, then it preserves the privacy of the seed trace. DPT and Ngram generate synthetic traces as random walks that do not incorporate destinations. However, a realistic trajectory heavily depends on the destination, and humans rarely visit a location following a *"let's go to a place where people usually go from here"* policy. In addition, a random walk can possibly generate synthetic traces with unrealistic length (see details in Section 5). Moreover, time-of-day is also left out from the models (Ngram, DPT, AdaTrace); clearly, this reduces the descriptive power of the generative model as human mobility does show strong time-of-day patterns [22]. In fact, time-of-day even influences trip destinations. (Just consider how your destination varies from 8am to 8pm.) Simply dropping the timestamps averages out the visit frequencies between night and day, morning and afternoon etc. SGLT applies large bins in order to incorporate time (morning, afternoon, evening and night). However, SGLT is mainly designed for CDR (Call Detail Record) datasets, and it utilizes the peculiarities of the implied mobility patterns, such as semantic similarity (SGLT) between locations. This approach can hardly generate valuable synthetic data when it comes to short term, dense movements, such as GPS trajectories of vehicles between start and end locations. Borrowing the example from [7], consider Alice and Bob spending all day at their respective work locations w_A and w_B, and all night at their respective home locations h_A and h_B. Obviously, their mobility models are semantically very similar, although it might be the case that $h_A \neq h_B$ and $w_A \neq w_B$. In this example, the best semantic mapping between locations will be $w_A \leftrightarrow w_B$ and $h_A \leftrightarrow h_B$. This example clearly introduces the type of mobility traces generated by this model. In our experiments, we applied datasets of vehicle trajectories where the above illustrated semantic similarity is not applicable at all times. Furthermore, in CDR-like data, we can mostly observe the locations where people reside for longer periods (such as work and home), while DP-Loc models individuals' movements from one of these locations to another, where the turns and stops depend on the hour of the day (e.g., owing to traffic jams). For example, oftentimes we take a different route from home to work depending on the traffic conditions. More formally, in the case of CDR (or similar) datasets, the transition probabilities between locations are closer to uniform; however, when we increase the sampling rate of locations along a route, this does not hold anymore (particularly when we also condition on time and destination), thus allowing us to build better models. Our model is best suited for trips that have a start and destination that can be fit in the same time-slot. For different datasets different time granularity can be used based on the underlying application. However, for CDR-like data, we suggest two possible alterations: (1) setting one time-slot to one day, or (2) remove all time information. The evaluation of these scenarios go beyond the scope of this paper, hence we leave them for future research.

Ngram, DPT and AdaTrace apply the Laplace mechanism to preserve Differential Privacy, where the noise is added to the Markovian probabilities. In contrast to these, we apply the Moments Accountant [1] method with the Gaussian mechanism that utilizes an advanced composition theorem by taking into account the exact noise distribution. Finally, SGLT has been shown [24] to be very slow, while DPT requires large computational power (256GB RAM and 48 cores for $50,000$ traces, whereas we generate $400,000$ traces), thus these models are not scalable for large datasets.

Most privacy-preserving training algorithms for neural networks are based on modifying the gradient information generated during backpropagation. The modification involves clipping the gradients (to bound the influence of any single record on the released model parameters) and adding calibrated random noise [1, 8]. Some works propose to use generative adversarial networks (GANs) [29] or mixture models [3] to directly generate privacy-preserving synthetic data. Differentially Private GAN [34, 35, 38] has been applied to gener-

ate image and Electronic Health Record data. The approach in [20] aims to generate time series with LSTM (Long-Short Term Memory Networks) and GAN (Generative Adversarial Networks) with DP guarantees, as well as multi-variate tabular data. GS-WGAN [9] uses gradient-sanitized Wasserstein GAN to generate synthetic data and has also been demonstrated on image data. Another DP-GAN architecture was proposed in [4] to release patient-level clinical trial data with Differential Privacy. All these techniques use some variant of DP-SGD (Differentially Private Stochastic Gradient Descent) [1] for training the discriminator of GAN. By contrast, PATE-GAN [26] was proposed to generate synthetic multi-variate tabular data using PATE [28]. Nonetheless, none of these generative models are specific to location data generation.

3 Preliminaries

3.1 Location Data

In general, location data is geographical information about a specific object's whereabouts associated to a time identifier. Formally, let $\mathbf{L} = \{L_1, L_2, \ldots, L_{|\mathbf{L}|}\}$ be the universe of locations, where $|\mathbf{L}|$ is the size of the universe. We assume that the whole universe is represented as a grid, and each location corresponds to a cell in the grid. Each record in a location database is a sequence of timestamped location visits drawn from the universe. Specifically, a sequence S of length $|S|$ is an ordered list of items $S = (L_{\ell_1}, t_1) \rightarrow (L_{\ell_2}, t_2) \rightarrow \ldots \rightarrow (L_{\ell_{|S|}}, t_{|S|})$, where $\forall 1 \leq i \leq |S|$, $L_{\ell_i} \in \mathbf{L}$. A location may occur multiple times in S. A location database D is composed of a multiset of sequences $D = \{S_1, S_2, \ldots, S_{|D|}\}$, where $|D| = N$ denotes the number of traces in D.

3.2 Differential Privacy

Differential Privacy [16] (DP) ensures that the outcome of any computation on a database is insensitive to the change of a single record. It follows that any information that can be learned from the database with a record can also be learned from the one without that particular record. In our case, DP guarantees that our generative model is not affected by any single original trajectory beyond the privacy budget measured by ε and δ, which can be computed as follows.

Definition 3.1 (**(ϵ, δ)-Differential Privacy** [15])**.** A privacy mechanism $\mathcal{M}$ gives (ϵ, δ)-Differential Privacy if for any database D_1 and D_2 differing on at most one record and for any subset of outputs $S \subseteq \mathit{Range}(\mathcal{M})$:

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D_2) \in S] + \delta$$

Intuitively, a privacy mechanism $\mathcal{M}$ satisfying Definition 3.1 does not release any information that is specific to any single record in dataset D up to ε and δ, where δ is preferably smaller than $1/|D|$ [15].

A fundamental concept for achieving Differential Privacy is the global sensitivity of a function [16]:

Definition 3.2 (**Global L_p-Sensitivity** [17])**.** For any function $f : \mathcal{D} \rightarrow R^d$, the L_p-sensitivity of f is $\Delta f = \max_{\mathcal{D}_1, \mathcal{D}_2} ||f(D_1) - f(D_2)||_p$ for all D_1, D_2 differing in at most one record, where $|| \cdot ||_p$ denotes the L_p-norm.

Differential Privacy maintains composition; the privacy guarantee of the k-fold adaptive composition of any mechanism can be computed using the moments accountant method [1]. The moments accountant generalizes the regular approach of keeping track of (ϵ, δ) using an advanced composition theorem by taking into account the exact noise distribution.

Definition 3.3 (**Privacy Loss** [1])**.** Let $\mathcal{M}$ be a privacy mechanism which assigns a value $O \in \mathit{Range}(\mathcal{M})$ to a dataset D. The privacy loss of $\mathcal{M}$ with datasets D_1 and D_2 and auxiliary input aux at output O is a random variable:

$$\mathcal{L}(O; D_1, D_2, \mathcal{M}, aux) = \log \frac{P[\mathcal{M}(D_1, aux) = O]}{P[\mathcal{M}(D_2, aux) = O]}$$

Definition 3.4. (**Log of the Moment Generating Function**) For a given mechanism $\mathcal{M}$, the log of the moment generating function evaluated at λ is: $\alpha_{\mathcal{M}}(\lambda; aux, D_1, D_2) = \log \mathbb{E}_{O \sim \mathcal{M}(D_1, aux)}[\exp(\lambda \mathcal{L}(O; D_1, D_2, \mathcal{M}, aux))]$.

Theorem 3.1. (**Moments Accountant** [1]) Let $\alpha_{\mathcal{M}}(\lambda) = \max_{aux, D_1, D_2} \alpha_{\mathcal{M}}(\lambda; aux, D_1, D_2)$ be defined as above. Let $\mathcal{M}_{1:k}$ be the k-fold adaptive composition of $\mathcal{M}_1, \ldots, \mathcal{M}_k$. Then:

1. *Composability:* $\alpha_{\mathcal{M}_{1:k}}(\lambda) \leq \sum_{i=1}^{k} \alpha_{\mathcal{M}_i}(\lambda)$
2. *Tail bound:* For any $\varepsilon > 0$, the mechanism $\mathcal{M}_{1:k}$ is (ϵ, δ)-differentially private for $\delta = \min_{\lambda} \exp(\alpha_{\mathcal{M}_{1:k}}(\lambda) - \lambda\epsilon)$.

The *Gaussian Mechanism* [17] consists of adding Gaussian noise to the true output of a function. In particular, for any function $f : D \to R^n$, the Gaussian mechanism is defined as adding i.i.d. Gaussian noise with variance $(\Delta_2 f \cdot \sigma)$ and zero mean to each coordinate value of $f(D)$. In fact, the Gaussian mechanism draws vector values from a multivariate isotropic Gaussian distribution which is described by random variable $\mathcal{G}(f(D), \Delta_2 f \cdot \sigma \mathbf{I}_n)$.

4 Model

4.1 Overview

Our goal is to generate private synthetic location traces. In particular, having a location dataset D with a multiset of trajectories, our goal is to build a generative model which approximates the true generator distribution of D, where every trajectory in D is a sample from this distribution. The model is built using the privacy-sensitive data D, and, hence, the training process of this model must guarantee Differential Privacy for any user/trajectory in D. Due to the large complexity of this model, we decompose it into four main parts as follows:

- **Dimensionality Reduction:** We identify the K most frequently visited cells on the map and work only with these locations afterwards by projecting all trajectories to these cells.
- **Trajectory Initialization:** A generative model, called *Trajectory Initializer (TI)*, learns the underlying joint distribution of the starting and ending locations and time variable of all trajectories, i.e., their very first (source) and very last (destination) location visits along with the single timestamp of the whole trace.
- **Transition Probability Generation:** A classification model, called *Transition Probability Generator (TPG)*, learns the transition probability distribution between any two consecutive locations, i.e., it outputs the probability distribution for the next hop in a trace, conditioned on the current location, the destination and time. Both of these models are trained with Differential Privacy guarantees on a potentially sensitive training dataset.
- **Trace Generation:** Sampling a source L_{src} and destination L_{dst} along with the time t from the output distribution of TI, and using the transition probabilities between any locations generated by TPG, the trace generator (TG) non-deterministically reconstructs a trajectory between source L_{src} and destination L_{dst} at time t combining Dijkstra's shortest path and the Metropolis-Hastings algorithm. As this process only uses the output of TI and TPG, and some public information about locations, the whole generation process becomes differentially private as the first two are already differentially private.

Algorithm 1: Differentially Private Synthetic Trace Generator (DP-Loc)

Input: Private Dataset D
Dimensionality Reduction:
Choose K most frequent locations TOP-K: $L_{\ell_1}, ...L_{\ell_K}$, $\forall L_{\ell_i} \in \mathbf{L}$ with Gaussian Mechanism
Project each trace $p \in D$ to TOP-K
Model Construction:
Train the Trace Initialization Model TI_{θ_1} on D with DP-SGD [1]
Train the Transition Probability Generation Model TPG_{θ_2} on D with DP-SGD [1]
Trajectory Reconstruction:
for $i \in [1, \ldots, |D|]$ **do**
  Sample $(L_{src}, L_{dst}, t) \sim TI_{\theta_1}$
  Build a routing graph $G(V, E)$, where $V = \mathbf{L}$ and $weight((L_x, L_y)) = -\log TPG_{\theta_2}[L_y|t, L_x, L_{dst}]$, where $(L_x, L_y) \in E$
  Find the path p between L_{src} and L_{dst} with the minimal total weight in G
  for $m \in [1, \ldots, 10]$ **do**
    Run Metropolis-Hastings on p: $p = MH(p)$
  Generate repetitions in the sampled path $s = (L_{\ell_1}, \ldots, L_{\ell_n})$:
  for $L_{\ell_j} \in s$ **do**
    $\eta \sim Geom(1 - TPG_{\theta_2}[L_{\ell_j}|t, L_{\ell_{j-1}}, L_{\ell_n}])$
    $s' = (L_{\ell_1}, \ldots, L_{\ell_{j-1}}, \underbrace{L_{\ell_j}, \ldots, L_{\ell_j}}_{\eta \text{ times}}, L_{\ell_{j+1}}, \ldots, L_{\ell_n})$
  $D' = D' \cup \{(s', t)\}$
Output: Synthetic dataset D'

Assumptions: Time is divided into equally sized slots which are sufficiently large to include whole trajectories. Each trajectory is assigned to a single time slot t, and all location visits of a trajectory take place within this slot t.

4.2 Model Description

In the remainder of this section, we describe our approach called DP-Loc (Differentially Private Synthetic Trace Generator) in more details that is also summarized in Alg. 1.

4.2.1 Dimensionality Reduction

We project all locations of every trajectory to the K most frequently visited locations (cells) on the map called as TOP-K locations. In particular, each location is mapped to the closest TOP-K location if there is any within a distance of 1000 meters. Otherwise, the whole trace is dropped. The value K refers to the number of most frequently visited cells where 95% of all visits occur; thus, it differs among datasets, and must be chosen in a differentially private fashion (see Section 5.6 and Section 4.3). (We made an exception and lowered the threshold percentage in case of GeoLife-250 dataset to 80% because of the low amount of datapoints in many cells. The same preprocessing was done for Ngram and AdaTrace as well.) The purpose of this dimensionality reduction is to increase model accuracy and also the speed of training. Indeed, if there are many cells on the map, the number of visits per cell typically has a power-law distribution: most cells are never or rarely visited. Using all grid cells would largely increase model complexity and hence training time. Moreover, it also degrades model quality due to the larger perturbation needed by DP (see Section 4.3). On the other hand, dimensionality reduction helps the model preserve large-scale mobility patterns more accurately (with high support) at the expense of losing fine-grained patterns (with low support). This trade-off can be dynamically chosen per application; we believe that retaining 95% of all visits provides reasonably high fidelity.

4.2.2 Trajectory Initialization (TI)

In order to sample a starting location $L_{src} \in \mathbf{L}$, a destination $L_{dst} \in \mathbf{L}$ and time $t \in \mathbf{T}$ for a synthetic trajectory, we build a differentially private Variational Autoencoder (VAE) (see Figure 3a in Appendix A for illustration) that is capable of approximating the joint probability distribution $Pr(L_{src}, L_{dst}, t)$. The model parameters θ_1 are learnt from a sensitive location dataset D, and hence training is performed with DP-SGD [1] (see Section 4.3 for details).

The output of TI_{θ_1} is a 3-dimensional vector $[L_{src}, L_{dst}, t]$ (recall that each trace has a single timestamp in our model). Notice that learning this distribution privately is challenging due to its high dimensionality; the domain of the joint probability distribution is $|\mathbf{L}| \times |\mathbf{L}| \times |\mathbf{T}|$, where $|\mathbf{T}|$ is the number of all possible time slots.

We one-hot encode the input, thus it has a dimension of $2 \times |\mathbf{L}| + 24$, where the size of $|\mathbf{L}|$ depends on the coarseness of the grid, and 24 is the number of hours in a day. Our encoder has two hidden *dense* layers $(100, 100)$ with ReLU and linear activation functions, respectively. The encoder outputs the parameters of the learned normal distribution $\mathcal{N}(\mu, \sigma)$; values drawn from this distribution by the decoder comprise the latent vectors of size 50. The decoder has to transform this latent variable to an actual sample. The decoder has only 1 hidden layer with size 100 and with ReLU activation. Finally, there are three parallel output layers with softmax activation, corresponding to a single output variable (location, destination, time). VAEs have their own specific loss functions; we applied the original one from [13].

4.2.3 Transition Probability Generation (TPG)

Our classifier TPG_{θ_2} is a feed forward network (FFN) endowed with word embedding. It is illustrated in Figure 3b in the Appendix. It approximates the true transition distribution $Pr[L_x \to L_y | t, L_{dst}]$ for any frequent (see details below) location L_x and L_y for every possible time slot $t \in \mathbf{T}$ and destination $L_{dst} \in \mathbf{L}$. That is, the probability that an individual at location L_x moves to location L_y towards destination L_{dst} at time t.

The input is (L_c, L_{dst}, t) (current location, destination, time), and the output is the probability distribution on the next hop. The two location coordinates of the input vector are fed into an embedding layer where they are embedded separately into the same 50-dimensional vector space[4]. Next, we concatenate these vectors with the time coordinate, resulting in a 101-dimensional vector. The next dense layer has a size of 200 with ReLU activation, and the output layer has softmax. We trained the network with *sparse categorical cross-entropy* and the *SGD* optimizer. As only the K most frequently visited cells are considered, the number of output classes is also K. We use DP-SGD [1] to train TPG and therefore the released model parameters θ_2 are differentially private (see Section 4.3 for details). Besides the current time and destination, the prediction of the next location depends only on the current location and not on the earlier location visits. That is, when the next location is predicted, we do not take into account how the current location is reached. This is not

4 The embedding layer is part of the network, thus trained together with the rest of the layers.

a far-fetched simplification; several studies have shown that 1 or at most 2-order Markov chains provide a sufficiently accurate estimation of the next location visit [21].

4.2.4 Remarks

The choice of an embedding layer, and an FFN instead of a recurrent neural network (such as LSTM) deserves more explanation. As locations exhibit similar characteristics to words, we can rely on the distributional hypothesis[5]. In case of locations, this means that if they follow each other in a given trajectory, then they are also close in the geographical space, and, therefore, will have similar representations in the embedded space. This implies that the model can automatically learn which locations are geographically close to each other.

Although LSTM has an implicit capability to handle temporal and sequential data, its differentially private training with SGD takes approximately 6-8 times longer than without Differential Privacy. However, training simple feed forward networks is considerably faster, almost as fast as the non-private model. Furthermore, our FFN has less parameters than the simplest but still well-performing LSTM, and having less parameters results in lower noise injection, and thus, higher utility. Our solution has an accuracy only 1-2% less than the LSTM layer on the considered datasets.

4.2.5 Trace Generation (TG)

When a trajectory is generated, we first sample a pair of source L_{src} and destination L_{dst} locations along with the time slot t from the output distribution of TI_{θ_1}. Then, a weighted directed routing graph $G(V, E)$ is built, where the edge weights are the transition probabilities between any two location points generated by TPG_{θ_2} (i.e., V is composed of locations in TOP-K, and $weight((L_x, L_y)) = -\log Pr[L_x \rightarrow L_y|t, L_{dst}]$ for any $(L_x, L_y) \in E$, that is the negative logarithm of the transition probability from L_x to L_y conditioned on destination D and time t). Note that G is complete and specific to a given destination L_{dst} and time slot t, hence different graphs are constructed for trajectories differing in their destination or time. The routing graph defines a distribution of paths between any location and destination L_{dst} at time t, and our task is to draw a path from this distribution in order to generate a trajectory.

To do so, the most probable trajectory is first constructed from graph G by applying Dijkstra's shortest path algorithm, and then the Metropolis-Hastings MCMC algorithm to the resulting shortest path, thus we generate one of the most probable paths between L_{src} and L_{dst}. As $-\log TPG_{\theta_2}[L_y|t, L_x, L_{dst}]$ is always non-negative, Dijsktra's shortest path algorithm finds the path with the minimum total weight between two vertices, which is equivalent to the most probable path between L_{src} and L_{dst} at time t in our case. Indeed, let P denote the set of all paths between L_{src} and L_{dst}. Then, the most probable path between L_{src} and L_{dst} is

$$\begin{aligned}
&\min_{p \in P} \sum_{(L_x \rightarrow L_y) \in p} -\log(TPG_{\theta_2}[L_y|t, L_x, L_{dst}]) \\
&= \min_{p \in P} -\log(\prod_{(L_x \rightarrow L_y) \in p} TPG_{\theta_2}[L_y|t, L_x, L_{dst}]) \\
&\approx \min_{p \in P} -\log(\prod_{(L_x \rightarrow L_y) \in p} Pr[L_x \rightarrow L_y|t, L_{dst}]) \\
&= \max_{p \in P} \prod_{(L_x \rightarrow L_y) \in p} Pr[L_x \rightarrow L_y|t, L_{dst}]
\end{aligned}$$

due to the monotonicity property of the logarithm.

This path finding algorithm is deterministic on its own, however this would not account for real life scenarios. Two vehicles can take different routes between identical starting and ending locations (depending on random environmental factors such as traffic, weather, road blocks, etc.). Therefore, we introduce randomness into our trace generation by applying the *Metropolis–Hastings algorithm* (MH) to the shortest path. Specifically, we have a target stationary distribution over all paths, where the probability of a path is computed as above from the routing graph. Sampling directly from this distribution is hard due to its finite but exponentially large domain, therefore, we rely on MCMC methods. Our MH algorithm applied to shortest paths is described in Algorithm 2. Markov chain theory says that we need multiple state transitions to have a "good enough" sample (that comes from a distribution close enough to the target), we perform 10 transitions. We set this value based on our *Route Distribution metric* (see Section 5.3.5) that measures the distance between original and synthetic routes taken between source-destination pairs. Our experiments in Section 5 show that 10 iterations are sufficient for larger datasets, however, smaller databases benefit from 100 and even 150 iterations.

The final step in our TG algorithm is *looping*, where we aim to approximate the time a vehicle stays in one

5 Words that co-occur in the same contexts tend to have similar meanings

Algorithm 2: Metropolis-Hastings Algorithm for TG

Input: Most probable path
$p = (L_{src} = L_{\ell_1}, L_{\ell_2}, \dots L_{\ell_n} = L_{dst})$

1 Choose uniformly random node
$L_{\ell_i} \sim U(p \setminus \{L_{\ell_1}, L_{\ell_n}\})$

2 Choose uniformly random neighbor $L^n_{\ell_i}$ of L_{ℓ_i}

3 Candidate path
$p_c = (L_{\ell_1}, \dots L_{\ell_{i-1}}, L^n_{\ell_i}, L_{\ell_{i+1}}, \dots L_{\ell_n})$

4 Let $\gamma = \dfrac{\prod_{(L_x \to L_y) \in p_c} TPG_{\theta_2}[L_y|t, L_x, L_{dst}]}{\prod_{(L_x \to L_y) \in p} TPG_{\theta_2}[L_y|t, L_x, L_{dst}]}$

5 $p = \begin{cases} p_c & \text{with probability } \min(1, \gamma) \\ p & \text{otherwise} \end{cases}$

Output: p

cell, that is, the number of repetitions of a single location in a trajectory. Looping allows to capture some traffic patterns more faithfully such as rush hours or traffic jams. We model this by generating the repetition number η of a location L_x from a geometric distribution $\eta \sim Geom(1 - TPG_{\theta_2}[L_x|t, L_x, L_{dst}])$, where $TPG_{\theta_2}[L_x|t, L_x, L_{dst}]$ is the probability output of TPG for staying at location L_x.

4.2.6 Remarks

Feeding the destination and time as an input to our transition probability generator enhances model accuracy by a large margin (in certain cases with more than 20%). The rationale behind this is that the probability of the next-hop location is heavily influenced by the direction of movement, i.e., the specific destination where the individual is heading for. Similarly, time also impacts the direction of movement towards a specific destination, especially in vehicular transport, where the route of a vehicle is largely influenced by the traffic, i.e., ultimately time dependent. This is in sharp contrast to earlier works [10] which solely used the last visited locations to predict the next location of a trajectory.

4.3 Privacy Analysis

In this section, we quantify the privacy guarantee of DP-Loc by using the moments accountant described in Section 3.2. Recall that we use the Gaussian Mechanism to provide DP for the dimensionality reduction (Section 4.2.1), and DP-SGD [1] for TI (Section 4.2.2) and TPG (Section 4.2.3). DP-Loc is the adaptive composition of these three mechanisms plus the trace generation (TG) described in Section 4.2.5.

Dimensionality Reduction: To select the most frequent K (top-K) cells, we employ the Gaussian mechanism (see Section 3.2) and add i.i.d Gaussian noise $\mathcal{G}(L_{max}\sigma_K)$ to the visit counts of *all* cells, where L_{max} is an upper bound on the trace length, and return the cells as top-K which have the K largest noisy counts. Both K and L are chosen based on the threshold where 95% of the visits is included in our calculations.

TI and TPG Models: To train the TI and the TPG models, we use the Differentially Private Stochastic Gradient Descent (DP-SGD) by Abadi et al. [1]. This method is independent of the chosen loss function and model, and it adds noise to the clipped gradients. In particular, the gradients of all model parameters in every model update are clipped to have a bounded L_2-norm with value C, and then Gaussian noise with variance $C^2_{TI}\sigma^2_{TI}$ (for TI) and $C^2_{TPG}\sigma^2_{TPG}$ (for TPG) is added to the clipped gradients before updating the parameters. The output of DP-SGD are the parameters θ_1 and θ_2 of TI and TPG, respectively. The sampling probability q in DP-SGD is calculated as follows. Our aim is to provide user-level (or in our case trajectory-level) Differential Privacy. However, recall that trajectories have different lengths, and we divided them into 1-grams: thus there are variable number of training examples belonging to a single trajectory for TPG. In the case of our TI model, we only have one sample per trajectory, thus the sampling probability of a single trajectory here is at most $|B|/|D|$, where $|D|$ is the total number of trajectories and $|B|$ is the size of batch B. However, for TPG, a single trajectory can have multiple samples, therefore, we sample batches differently. We first sample a trajectory from the dataset uniformly at random, and then a 2-gram out of this trajectory also uniformly at random. We repeat this experiment until a batch B of grams (training samples for TPG) is collected. This sampling mechanism ensures that any trajectory is equally probable to participate in an update (batch), and hence the sampling probability becomes $q = \frac{|B|}{|D|}$.

Trace Generation (TG): The trace generation uses only the differentially private models TI_{θ_1} and TPG_{θ_2} as input, and does not access private data. Therefore, the generated traces are also differentially private due to the post-processing property of DP.

Let $\eta_0(x|\xi) = \mathsf{pdf}_{\mathcal{G}(0,\xi)}(x)$ and $\eta_1(x|\xi) = (1 - q)\mathsf{pdf}_{\mathcal{G}(0,\xi)}(x) + q\mathsf{pdf}_{\mathcal{G}(1,\xi)}(x)$ where q is the sampling probability of a single trace in a single round. Let

$$\alpha_K(\lambda) = (\lambda^2 + \lambda)/4\sigma_K^2 \quad (1)$$

$$\alpha_{TI}(\lambda) = \log\max(E_1(\lambda, \sigma_{TI}), E_2(\lambda, \sigma_{TI})) \quad (2)$$

$$\alpha_{TG}(\lambda) = \log\max(E_1(\lambda, \sigma_{TPG}), E_2(\lambda, \sigma_{TPG})) \quad (3)$$

where $E_1(\lambda,\xi) = \int_{\mathbb{R}} \eta_0(x|\xi) \cdot \left(\frac{\eta_0(x|\xi)}{\eta_1(x|\xi)}\right)^{\lambda} dx$, $E_2(\lambda,\xi) = \int_{\mathbb{R}} \eta_1(x|\xi) \cdot \left(\frac{\eta_1(x|\xi)}{\eta_0(x|\xi)}\right)^{\lambda} dx$, and $\alpha_K(\lambda)$, $\alpha_{TI}(\lambda)$, $\alpha_{TG}(\lambda)$ are the log moments (see Definition 3.4) of the TOP-K identification, TI, and TPG, respectively. $\alpha_K(\lambda)$ in Eq. (1) follows from Lemma 1 in [3].

Theorem 4.1 (**Privacy of DP-Loc**). Let e_{TI} and e_{TG} denote the number of epochs for TI and TPG, and $|B|/|D|$ is the number of SGD iterations per epoch. DP-Loc is (ϵ, δ)-DP, where $\varepsilon = \min_\lambda \frac{1}{\lambda}\left(\alpha_K(\lambda) + \frac{|B|}{|D|}e_{TI}\alpha_{TI}(\lambda) + \frac{|B|}{|D|}e_{TG}\alpha_{TG}(\lambda) - \log(\delta)\right)$

Proof. Let $\alpha_{DPLoc}(\lambda)(\lambda)$ denote the log moment DP-Loc in Alg. 1. It follows from the second part of Theorem 3.1 that $\varepsilon = \min_\lambda \frac{1}{\lambda}(\alpha_{DPLoc}(\lambda) - \log\delta)$. Since dimensionality reduction is only applied once at the very beginning of DP-Loc, and there are $\frac{|B|}{|D|}e_{TI}$ SGD iterations in TI and $\frac{|B|}{|D|}e_{TG}$ SGD iterations in TG in total, it also follows from the first part of Theorem 3.1 that $\alpha_{DPLoc}(\lambda) \leq \alpha_K(\lambda) + \frac{|B|}{|D|}e_{TI}\alpha_{TI}(\lambda) + \frac{|B|}{|D|}e_{TG}\alpha_{TG}(\lambda)$ which concludes the proof. □

Given a fixed value of δ, ε is computed numerically as in [1].

5 Experimental Evaluation

In this section, we empirically evaluate DP-Loc on three publicly available datasets, where two contain the GPS trajectories of different taxi trips from San Francisco [30] and Porto [27], and one has miscellaneous trajectories (e.g. walking, driving, public transport, cycling) (mostly) from Beijing, China (*GeoLife* dataset [39–41]). We show that the synthetic trajectories generated by our model are close to the original trajectories according to four different utility metrics, and apply a fifth utility metric to demonstrate the usefulness of the Metropolis-Hastings algorithm. The architecture described in Section 4.2 is fixed for all datasets. Although it has been shown that the optimization of hyperparameters (including the architecture of DP-Loc) is possible with DP guarantees [1, 23], yet this requires a larger privacy budget. For simplicity and owing to the similarity of the three datasets evaluated, we do not apply this technique here.

5.1 Data

San Francisco Taxi: The original San Francisco (SF) taxi dataset contains a set of GPS trajectories with timestamps recorded by 536 taxis. They were collected over 30 days in the San Francisco Bay Area of the USA in 2009. The trajectories cover the region of San Francisco within the bounding box of (37.6017N, 122.5158W) and (37.8112N, 122.3527W) – approximately 340 km^2. The original sampling rate of these trajectories is roughly 1 per minute. We used all the 276744 trajectories from the SF dataset.

Porto Taxi: The original Porto dataset contains 1.7 million GPS trajectories with timestamps of 441 taxis. This dataset was acquired in the metropolitan area of Porto, Portugal, over a period of nine months in 2012. The sampling rate varies, but it is approximately 1 per 15 seconds. We considered taxi trips only within a bounding box of (41.00456N, -8.7368W) and (41.2728N, -8.4412W) around the city, with a size of approximately 600 km^2. We used a random sub-sample containing 450000 trips for our evaluation.

GeoLife: The original GeoLife dataset is a collection of 17,621 daily trajectories by 182 users in a period of over five years (from April 2007 to August 2012). 91.5% of the trajectories are logged in a dense representation, e.g., every 1-5 seconds or every 5-10 meters per point. The GeoLife dataset includes a wide range of outdoor user movements, including commuting, entertainment and sports activities, such as shopping, sightseeing, dining, hiking, and cycling. The majority of the data was generated in Beijing, China, where we set our bounding box of (39.75N, 116.2W) and (40.1N, 116.55W) of approximately 580 km^2, and considered the traces within this region only. Note that Porto and GeoLife cover approximately the same area size, however, the Porto bounding box also contains its uptown area while GeoLife does not. In other words, the whole Beijing area is populated uniformly almost everywhere with approximately one seventh in available data points of the Porto dataset. Moreover, since the dataset contains continuous recordings of people's daily traces with multiple stationary locations, we cut these traces along their stationary locations and replace the original trace with its subtraces in the dataset. More precisely, if an individual stayed at one location for a long time (more than 15 minutes), then the location is regarded as stationary and we terminated the given trace and initialized a new one with the current cell as new starting point. The justification behind such slicing is twofold. First, we intend to model and release only the mobility patterns of people

which is the main focus of most practical applications (such as traffic optimization). Second, the original GeoLife dataset is small in terms of both individuals and traces; with the help of trace slicing we obtained 59,907 traces. Note that (strictly speaking) DP-Loc provides DP guarantees only to individual sub-traces in this case; however, we embrace this limitation on two accounts: i) we could not obtain a larger suitable third dataset, ii) the limited number of users in the original GeoLife dataset would result in a low utility irrespective of the actual DP-based privacy-preserving mechanism applied.

Regarding the taxi datasets our objective is to preserve the privacy of passengers and not taxi drivers, and hence a trace (sequence) is composed of the recorded location visits of a single taxi trip.

5.2 Data Preprocessing

We consider two grids with different cell size. The smaller grid consists of cells with size of $250 \times 250\text{m}^2$, and the larger one with cells of size $500 \times 500\text{m}^2$. We have found that these two sizes are small enough to capture the movement of an individual, and also large enough to enable neural networks to learn the underlying distribution with sufficient accuracy and low computational cost (see Appendix A the complexity analysis). Although, in a real-life scenario, a given application can change these sizes, we believe that most practical use-cases would not divert very much from these values. Each GPS location point is assigned to its covering cell. Therefore, every trace (taxi trip) is composed of the sequence of location visits, each containing a pair of cell and the time of the visit. All traces with velocity larger than 150 km/h (calculated between two GPS points) or being out of the bounding box are dropped. Since the sampling rate was not constant in the database, we applied two transformations to make it more regular; (1) cell visits are aggregated in time by 60 seconds keeping the cell that was the most frequent in the trace during this time frame, and (2) when there were gaps shorter than 5 minutes without any location visits, these missing visits are approximated by linear interpolation. We select the K cells that contain at least 95% of all visits in a privacy-preserving manner and we map each point to the closest top-K cell, if there is any within 1000m. We drop traces with locations that cannot be assigned to a top-K cell. Finally, if the resulting trace had only a single visit, it is dropped. All traces are truncated to length L_{max}. Since DP adds noise to the cell counts, the values of K and L can change between runs; however, larger datasets are not prone to this instability. To illustrate this, we generated the values of K and L 5 times and included their average in Table 1. The standard deviation for Porto and SF were 0, however, for GeoLife it was 20 for the K values; the L values were stable in all cases. The data loss caused by top-K filtering was $3-5\%$ for all datasets. Less than 1% of traces were dropped by the other filters.

After cleaning and smoothing the data, the timestamps are further aggregated by assigning only the hour of the day, when the dominant part of the taxi trace was present, to all visits of a single trace. For example, if a trace started at 17:58, and ended at 18:10 with 12 visits altogether, we assigned the 18^{th} hour of the day to every cell visit of the trace (including those which happened before 18:00). As our aim is only to demonstrate the feasibility of our approach, this simplification is introduced in order to decrease the size of the input and output space making training faster and the models less complex. Finally, we created 2-grams from the traces, i.e., we grouped every two consecutive data points together to create a single training sample for our TPG model, where the first and second parts served as input and output for the model during training. The first part of every gram is augmented with the destination cell of the trace where the gram comes from, and the second gram contains only the cell identifier of the next location (without timestamp).

Table 1 shows the descriptive statistics of the datasets.

5.3 Evaluation Metrics

Choosing a truly representative metric is always challenging and, to a large extent, application-dependent. We focused on some commonly used characteristics of PoIs, heatmaps, popular start and destination places, rush hours, traffic jams, trip lengths, etc. As a result, we consider four different utility metrics which are mainly borrowed from previous works [10, 24]. Each of them is evaluated both on the synthetic and the original dataset, and the difference is measured according to different distance metrics.

5.3.1 Trip Size Distribution

The Jensen-Shannon divergence (JSD) is computed between the distribution of the trip lengths in the synthetic and the original datasets in each hour of the day

Dataset	$\|D\|$	$topK$	max $\|t\|$	avg$\|t\|$	std.dev$\|t\|$	mode$\|t\|$
SF-250	**431,222**	**550**	**19**	**10.54**	**6.67**	**7**
SF-500	**431,222**	**179**	**19**	**10.46**	**6.17**	**7**
Porto-250	**450,000**	**490**	**24**	**11.63**	**7.53**	**9**
Porto-500	**450,000**	**211**	**24**	**11.52**	**8.64**	**9**
GeoLife-250	**59,907**	**406**	**14**	**13.91**	**11.38**	**10**
GeoLife-500	**59,907**	**503**	**22**	**13.86**	**11.38**	**10**

Table 1. The preprocessed datasets used in our experiments: SF-250, Porto-250, GeoLife-250 (with a cell size of 250m^2) and SF-500, Porto-500, GeoLife-500 (with a cell size of 500m^2). Trace length $|t|$ is for traces $t \in D$.

(the trip length is the number of cells of a trip). Note that, unlike the Kullback-Leibler (KL) divergence, JSD is symmetric and has a finite value. In our case, JSD is bounded between 0 (identical distributions) and 1 (least similar distributions).

5.3.2 Frequent Patterns

The top-N most frequent patterns (i.e., subsequences of locations) are computed both in the original D and synthetic dataset D', which are denoted by $F_N(D)$ and $F_N(D')$, respectively. The true positive ratio $\frac{|F_N(D) \cap F_N(D')|}{N}$ is reported for $N = 10, 20, 50, 100$.

5.3.3 Spatio-temporal Distribution Of Location Visits

The spatio-temporal density in a given hour of the day is the number of visits in each cell of the considered region. This histogram, where each bin corresponds to a cell, is computed from both the synthetic and the original traces individually, and the spatio-temporal density of the original dataset as well as the synthetic datasets are obtained from these histograms after normalization. The Earth Mover's Distance [32] is reported between these distributions, which measures their difference in terms of geographical distance (meters) and is a metric for probability distributions. Specifically, EMD measures the "amount of energy" (or cost) needed to transform one distribution to another where the ground distance is the geographical distance between the centers of cells. The EMD between the spatial densities of the original and synthetic data are reported for cells that include at least 80% of the data, but no more than 2000 cells for every hour, and also over all hours of the day.

5.3.4 Spatio-temporal Distribution Of Source and Destination Pairs

The joint distribution of the source and destination locations is computed from the original and synthetic datasets individually, and their EMD is reported for every hour, and over all hours of the day. In particular, we count the relative frequency of every possible pair of source and destination locations in both datasets, and compute the EMD between these two distributions as above (the distance between a pair of location points is the sum of their individual distances). As the domain of this joint distribution has a size of $|\mathbf{L}| \times |\mathbf{L}|$ (in every hour, not over all hours), the computation of EMD can be very costly. We used the same approximation as for EMD density described above. Note that besides reporting the hourly EMD, we also calculated the EMD averaged over all hours in order to compare with prior works.

5.3.5 Route Distribution

We calculate the histograms of cell visits for the inner points of a trace conditioned on the source and destination pairs, i.e. the histogram shows the distribution of cells that were visited between a given source and destination. The route distribution is calculated for both original and synthetic datasets, then their EMD is measured for each source-destination. Due to the high number of such pairs, we report the average of their EMD values. Intuitively, this metric measures how "realistic" a synthetic trajectory is. Due to the very high number of source and destination pairs we used the same approximation described above.

5.4 Baselines

As comparison to DP-Loc, we evaluate the Ngram model from [10] (Ngram), and the AdaTrace model

from [24]. We chose AdaTrace because it is the state-of-the art technique and outperforms several other models (Ngram, DPT, SGLT) as stated in [24]. Ngram is chosen because it performs very well regarding frequent patterns and is a more general solution for any sequential data (unlike AdaTrace). For the reasons explained in Section 2, we do not compare DP-Loc to SGLT [7], since it is incompatible with the mobility patterns represented by our datasets. Moreover, our goal is to generate large synthetic datasets, and SGLT was shown [24] not to scale well in this regard[6].

Since AdaTrace uses a dynamic grid configuration we had to align our grid in order to reach a fair comparison. First, we transformed the GPS coordinates with Mercator projection and applied all our preprocessing steps to the records (even top-K selection). Next, we generated synthetic traces with AdaTrace, where the output is also in the Cartesian space. For the evaluation with our metrics, we assigned the output coordinates to our non-dynamic grid cells.

As AdaTrace and Ngram apply the Laplace mechanism, for fair comparison, we changed it for the Gaussian mechanism and calculated the values of ϵ by the moments accountant method. For both models, given the value of $\delta = 1/|D|$ and $\epsilon = [0.5, 1, 2, 5]$, we analytically calculated the variance of the noise using Theorem 3.1. The L_2 sensitivity of the Adatrace can be shown to be $\Delta_2 = 1$ based on [24]. As for Ngram, we bounded the L_2 sensitivity with the L_1 for the same reason as in our top-K selection (see Section 4.3 for details).

5.5 Experiment Setup

We experimentally evaluate the performance of our solution in terms of the above four utility metrics on the datasets described in Table 1. We also present the results of Ngram and AdaTrace on the same datasets, and compare these to our work using the four metrics. As neither Ngram nor AdaTrace releases time information, we dropped all timestamps in all datasets, and synthesized the resulting datasets with the two models (by contrast, DP-Loc was always executed on the original datasets with temporal information, and its results are reported in Figure 1 and 4). We obtained the implementation from the respective authors. Experiments were conducted using Tensorflow 2.0 and Python 3.6.9 on a single Linux server with 98GB RAM and 16 cores.

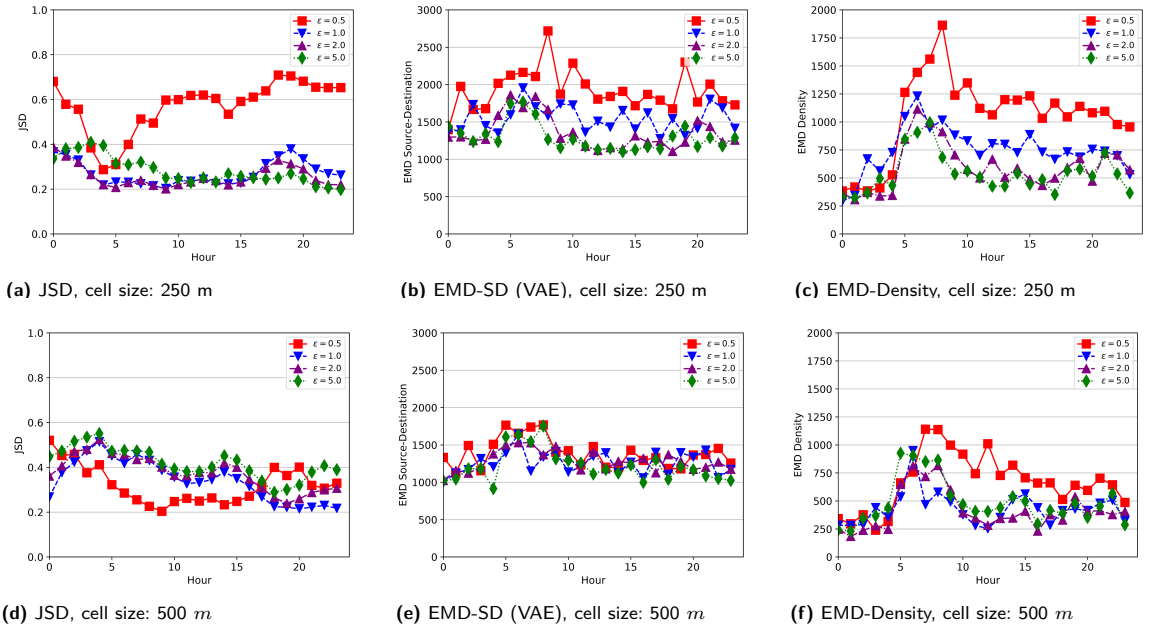
6 We also tried to run DPT, but even for 100.000 traces it ran out of 100GB of memory.

We have carried out our training in 24 different settings, combining $\epsilon = 0.5, 1, 2, 5$ with two grid resolutions (250 and 500), and three datasets (for statistics see Table 1). For the TI model, we set the L_2-norm clipping threshold C_{TI} to 1.00, the batch size $|B|$ to 200, and ran the training over $e_{TI} = 15$ epochs (recall that $C_{TI}^2\sigma_{TI}^2$ is the variance of the Gaussian noise added to the gradients in order to provide DP). With $\epsilon = 1$ and $\sigma_{TI} = 1.5$, the learning rate was set to 0.2; with $\epsilon = 2$ and $\sigma_{TI} = 1.00$, the learning rate was set to 0.5; and with $\epsilon = 5$ and $\sigma_{TI} = 0.7$, the learning rate was 0.5. Unless otherwise noted, we use $\delta = 1/|D|$ in the rest of the paper, and ϵ is the overall privacy budget of DP-Loc. For the TPG model, we set the L_2-norm clipping threshold $C_{TPG} = 3.0$, the batch size to 200, and trained the model over $e_{TPG} = 15$ epochs. With $\epsilon = 1$ and $\sigma_{TPG} = 1.6$, the learning rate was set to 0.1; with $\epsilon = 2$ and $\sigma_{TPG} = 1.00$, the learning rate was 0.15; and with $\epsilon = 5$ and $\sigma_{TPG} = 0.6$, the learning rate was 0.15. In dimensionality reduction, we add Gaussian noise to the counts of all location visits (when choosing the top-K locations, see Section 4.3) with $\epsilon = 1$, $\sigma_K = 3.8$; $\epsilon = 2$, $\sigma_K = 1.9$; and $\epsilon = 5$, $\sigma_K = 1.6$. The total ϵ of DP-Loc is computed according to Theorem 4.1 over 15 epochs for each of the TI and TPG models ($e_{TI} = e_{TPG} = 15$).

5.6 Results

In Figure 1 and 4, we report the hourly JSD and two EMD values between the original and the synthetic data generated by DP-Loc for $\epsilon = [0.5, 1, 2, 5]$. Figure 1 (and 4 in Appendix) show how the granularity of the grid influences the impact of the noise. Comparing the first and second lines of Figure 1 and 4, one can see that a larger grid size results in less error, though with coarsened data. (Coarsening the granularity eventually results in zero error of EMD, since all points end up in the same cell.) For comparison, we report the overall JSD, EMD-SD, EMD-Density and Frequent Patterns results of our synthetic traces and that of Ngram and AdaTrace in Table 5. GeoLife follows the same trends, thus we included the more comparative results in Table 5.

Trip Sizes: In Figure 1 and 4, JSD shows the same trend and takes up very similar values for $\epsilon = 1, 2, 5$. However, this is not true for $\epsilon = 0.5$, since, to ensure such a low ϵ value, the algorithm requires considerably more noise. The San Francisco dataset shows the same tendency for JSD values (see Appendix A for additional results). In the $\epsilon = 0.5$ case, JSD values are much larger when the cell size is 250m, also we can see that the



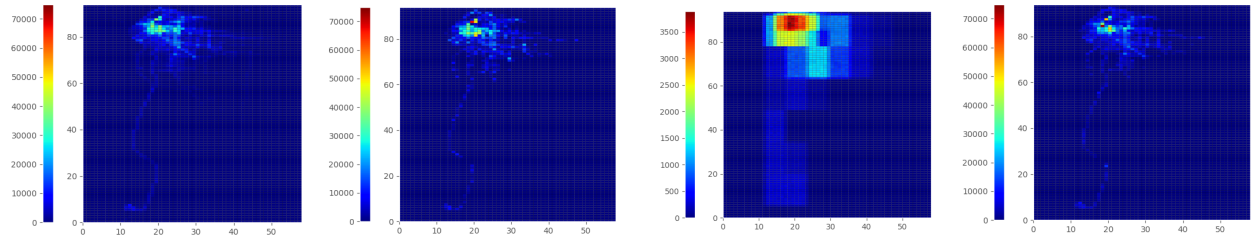
(a) JSD, cell size: 250 m **(b)** EMD-SD (VAE), cell size: 250 m **(c)** EMD-Density, cell size: 250 m

(d) JSD, cell size: 500 m **(e)** EMD-SD (VAE), cell size: 500 m **(f)** EMD-Density, cell size: 500 m

Fig. 1. Performance of our approach on Porto dataset depending on the time ($\delta = 4 \cdot 10^{-6}$).

EMD-SD values are also high. The TI model produced cells that were too close to each other, that caused very short traces (4.8 in average) in return. Table 5 shows that a larger grid generally results in larger JSD, but smaller EMD values for DP-Loc and AdaTrace, but larger for Ngram. *Recall that Ngram and AdaTrace do not include time information*, thus we only report one value for each setting in Table 5. To ease readability, we colored the best values at each metric among the three models in red. DP-Loc's JSD results are clearly much lower (i.e., closer to the original distribution) than that of the other two models. The MH iterations and looping extension play a large part in these low results, but also by leaving out these steps we still get much lower JSD values than Ngram and AdaTrace (experiments show that such a JSD results in approximately a value of 0.5 JSD). Ngram generates traces without destination, and it does not select the globally most likely trajectory. In contrast to this, DP-Loc is more realistic. Although AdaTrace does include the destination, it still performs much worse than DP-Loc regarding trip sizes. We hypothesize that the high JSD values for AdaTrace could be due to the dynamic construction of the grid. AdaTrace works in two layers only and it is probable that many areas are not optimally divided, and our uniform grid with top-K selection performs better. Moreover, for Ngram and AdaTrace, the average generated trace length was approximately 3 (in all settings); both the mode and standard deviation were 2. In contrast to these, DP-Loc generates traces with an average length of 12, mode of 8 and standard deviation of 5, which are almost identical to the statistics in Table 1. Ngram represents the dataset with a prefix tree which contains the set of all grams occurring in the dataset along with their occurrence counts, i.e. Ngram adds noise to these counts and prunes the noisy tree by removing grams with too small noisy count in order to improve accuracy. In particular, Ngram keeps grams only with sufficiently large occurrence counts, and shorter grams have larger counts which tend to survive sanitization unlike longer grams [10] with generally smaller counts. As a result, stronger privacy requirement (smaller ϵ) results in a set of shorter grams and eventually a smaller sanitized prefix tree. Although shorter sanitized grams can be combined into longer grams in Ngram, the number of possible extensions was not high in our datasets. The finally released grams coincide with the set of most frequently visited places hence Ngram preserves the statistics of most frequent patterns accurately. The same holds for AdaTrace, because it also follows the Markov assumption to generate traces.

EMD-SD: In Figure 1b, 1e, 4b and 4e, EMD is reported between the spatial distribution of the source and destination pairs depending on the time of the day. In Figure 1b and 1e, EMD values stay steadily between 900 and 2000 meters except in Figure 1b, where the trend for $\epsilon = 0.5$ is similar but the values exceed 2500m. The results for the four different values of ϵ are almost the



(a) Original,cell size: 250 m **(b)** DP-Loc, cell size: 250 m **(c)** AdaTrace, cell size: 250 m **(d)** Ngram, cell size: 250 m

Fig. 2. Heatmaps of the synthetic and original databases.

Table 2. Result of our DP-Loc algorithm without Differential Privacy and with 10 MH iterations

Dataset	FP-10	FP-20	FP-50	FP-100	JSD len	avg len	std. dev. len	EMD src-dst	EMD density
SF-250	0.54	0.60	0.78	0.78	0.362	13	5.6	1572	602
SF-500	0.80	0.80	0.88	0.88	0.322	14	6.1	1669	629
Porto-250	0.70	0.90	0.90	0.92	0.296	13	5.3	1113	345
Porto-500	0.77	0.80	0.85	0.87	0.360	15	6.3	1201	367
GeoLife-250	0.90	0.90	0.90	0.90	0.408	10	3	4211	1201
GeoLife-500	1.00	1.00	1.00	1.00	0.480	10	3.5	4746	1116

Table 3. Result measured on a downtown area in Porto. $\epsilon = 1$, and 10 MH iterations

Dataset	FP-10	FP-20	FP-50	FP-100	JSD len	EMD-src-dst	EMD-Density
Ngram	0.90	0.90	0.84	0.98	0.878	1766	74
AdaTrace	0.40	0.50	0.64	0.58	0.709	1488	336
DP-Loc	0.90	0.85	0.95	0.90	0.278	1018	311

Table 4. Results of route EMD for different MH iterations for $\epsilon = 1$ (measured in meters)

	0	1	5	10	25	50	100	150
SF-500	1335	1252	1177	1154	1171	1177	1180	1179
Porto-500	1347	1229	1137	1116	1119	1126	1134	1135
GeoLife-500	1907	1693	1918	1794	1361	1150	1121	1119

same if the larger cell size is used, and keep a steady distance in the 250m case. Table 5 shows that DP-Loc outperforms Ngram and AdaTrace, and generates more realistic endpoints for the traces. This demonstrates the generative power of the applied VAE network. The San Francisco dataset shows slightly different results. The EMD values in Figure 4 are almost the same for different values of ϵ, except for $\epsilon = 0.5$ in Figure 4b where EMD values are higher. We can also see that around 2 pm there is a peak in the two EMD values. We hypothesize that this is due to insufficient quality and quantity of data at that time period, which is not present in the Porto dataset. Nonetheless, DP-Loc still gives outstanding result in most cases. Note that smaller values are better, and EMD sums up all costs over the whole distribution. The distance between two neighboring cells is 250/500 m (center-to-center); if, e.g., EMD equals to 1000m, it is the size of 2/4 cells over the distribution.

EMD-Density: In Figure 1c, 1f, 4c and 4f, EMD is reported between the distribution of location visits for every hour of the day. Values show a trend similar to EMD-SD, but remain below 1000m in most cases, and $\approx 6-700$m on average. The values for $\epsilon = 0.5$ again keep a distance from the other results with a peak just below 2000m for the smaller cell size. Ngram outperforms DP-Loc and Adatrace (except on Porto-500 where DP-Loc is the best). However, Ngram was originally designed to focus on the accurate release of the most frequent subsequences, and reconstructs traces from these. Therefore, as long as the number of visits per cell follows a power-law distribution, Ngram is expected to remain superior. Nevertheless, Table 5 also shows that DP-Loc significantly outperforms AdaTrace in all cases. Figure 2 shows the San Francisco heatmaps of the synthetic datasets generated by all three models compared to the original one in Figure 2a. Note that the scale of Ada-

Trace is lower than that of the rest (using the same scale would render AdaTrace's patterns almost invisible). We can see that the original (left) image is more detailed than any other image, but DP-Loc and Ngram imitate it closely. The original image contains less highly populated areas (deep red), and Ngram has the densest cell of all (ruby red) in downtown SF.

Frequent Patterns: Table 5 reports the overall results of the *Frequent Patterns* metric, that is, the true positive ratio of the top-N location subsets between the original and the private synthetic databases. For all datasets, DP-Loc and Ngram perform similarly; from all 120 cases DP-Loc outperforms Ngram 40 times, and underperforms 51 times, but in most cases the margin is very small. In all cases the performance of DP-Loc is higher than that of AdaTrace. The low FP values of DP-Loc regarding the GeoLife dataset show that the dataset size was insufficient for the neural networks to learn the underlying distribution. Beijing is approximately twice as big as Porto, however, the data available in the GeoLife dataset was only 1/8th of that. Comparing Figure 2 along with the FP results to the JSD results, we can see why there is not a single universal metric for fair comparison. Ngram shows very good results when it comes to density (and frequent patterns), but the generated traces are highly unrealistic. DP-Loc might not perform as outstanding as Ngram regarding density metrics, but it produces realistic timestamped trajectories.

DP-Loc On Smaller Bounding-box: The San Francisco and Porto datasets contain the whole urban area with their airports included on the map. For fair comparison we have run DP-Loc on the downtown area of Porto, similarly to [24]. The results are shown in Table 3; we only evaluated the downtown area in one scenario where the cell size is $500m$ and $\epsilon = 1$. It is visible that the performance of AdaTrace is better than on a larger grid, however, the ratio among the best results is the same as measured with a larger bounding-box.

DP-Loc Without DP: We also evaluated DP-Loc without Differential Privacy, i.e. no noise was added to the model at any stage. Table 2 shows the results for the non-privacy preserving generated datasets. We can see that the numbers are almost identical to the ones in Table 5. The JSD and FP values are similar, and the two EMD values are lower than their private counterpart.

Justification for MH: We applied the Metropolis-Hastings algorithm to the shortest paths in order to obtain convergence to a target stationary distribution over all paths, where the probability of a path is computed from the routing graph. In Table 4 we report the results of the route EMD metric, i.e., the EMD distance (in meters) between the original and the synthetic routes (cells) taken between source and destination pairs. Results show that for large datasets 10 iterations of MH results in the most realistic traces. However, for smaller dataset, where the added noise has a higher influence on the generated traces, 100 and 150 iterations of MH show sufficient improvement. Consequently, after 10 or 100 iterations we could also observe a small drop in the EMD-Density values as well.

6 Conclusions

We proposed a novel approach to release location data with strong privacy guarantees. In contrast to prior works, DP-Loc is capable to release time information along with location visits without suffering significant utility loss. Our framework consists of generating the source and destination pairs of every trace, computing the transition probabilities between neighboring locations, then generating synthetic trajectories between the source and destination using a Monte Carlo algorithm. The transition probability depends on the time and destination, and is computed between the most frequent locations. We evaluated our proposal on three public location datasets and designed neural networks to model the distribution of trajectories. These networks are simple and hence fast to train even with DP guarantees. Results show that the provided utility is meaningful. Therefore, our technique can be a compelling new approach to the privacy-preserving release of complete location trajectories with time information. Importantly, we produce synthetic datasets that preserve many different statistics of the original dataset. Undoubtedly, releasing only a few targeted statistics with or without Differential Privacy, instead of the complete synthesized dataset, is a different approach which should always result in greater accuracy but only with respect to the released statistics. The proposed framework is general and finding the best generative models to a given type of data is difficult and requires domain expertise. We believe that our general approach may be applicable to other types of sequential data than location trajectories such as different time series.

Acknowledgement

The research was supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program.

Table 5. Summary of results with 10 MH iterations. AdaTrace and Ngram ignores the time of trips, hence we report the the overall JSD and EMD values over the whole period for our approach. Best values are in red.

		SF				Porto				GeoLife			
		250		500		250		500		250		500	
		Ngram	Ada	DP-Loc	Ada	DP-Loc	Ngram	Ada	DP-Loc	Ngram	Ada	DP-Loc	Ngram	Ada	DP-Loc
FP10	$\epsilon = 0.5$	0.80	0.20	0.80	0.20	0.80	0.80	0.20	0.70	0.90	0.28	0.60	0.66	0.05	0.30	0.70	0.01	0.70
	$\epsilon = 1$	0.80	0.10	0.80	0.10	0.82	0.80	0.20	0.65	0.90	0.30	0.70	0.70	0.05	0.30	0.80	0.05	0.60
	$\epsilon = 2$	0.80	0.20	0.85	0.30	0.85	0.80	0.10	1.00	0.90	0.40	0.70	0.80	0.10	0.28	0.80	0.14	0.87
	$\epsilon = 5$	0.80	0.10	0.85	0.40	0.90	0.90	0.10	1.00	0.90	0.40	0.70	0.90	0.16	0.30	0.92	0.20	1.00
FP20	$\epsilon = 0.5$	0.75	0.21	0.90	0.41	0.88	0.75	0.11	0.90	0.70	0.30	0.90	0.77	0.10	0.30	0.55	0.05	0.60
	$\epsilon = 1$	0.70	0.15	1.00	0.40	0.85	0.75	0.10	0.80	0.80	0.30	0.88	0.77	0.11	0.30	0.60	0.05	0.60
	$\epsilon = 2$	0.75	0.15	1.00	0.55	0.85	0.85	0.05	1.00	0.80	0.50	0.76	0.70	0.20	0.28	0.80	0.15	0.87
	$\epsilon = 5$	0.75	0.35	1.00	0.50	0.85	0.90	0.30	1.00	0.90	0.45	0.75	0.80	0.20	0.30	0.90	0.25	1.00
FP50	$\epsilon = 0.5$	0.95	0.18	1.00	0.60	1.00	0.90	0.15	0.90	0.80	0.45	1.00	0.80	0.10	0.25	0.58	0.12	0.53
	$\epsilon = 1$	1.00	0.12	1.00	0.60	1.00	0.95	0.14	0.90	0.80	0.42	0.93	0.80	0.10	0.28	0.58	0.12	0.40
	$\epsilon = 2$	1.00	0.34	1.00	0.62	0.95	0.95	0.26	1.00	0.90	0.40	0.88	0.85	0.16	0.28	0.80	0.27	0.87
	$\epsilon = 5$	0.91	0.36	1.00	0.62	0.92	1.00	0.26	1.00	0.80	0.42	1.00	0.80	0.29	0.30	1.00	0.27	1.00
FP100	$\epsilon = 0.5$	1.00	0.40	1.00	0.54	1.00	0.95	0.24	0.95	0.80	0.49	1.00	0.90	0.12	0.25	0.68	0.17	0.53
	$\epsilon = 1$	1.00	0.37	1.00	0.55	1.00	1.00	0.24	1.00	0.90	0.53	0.90	0.90	0.15	0.28	0.58	0.17	0.4
	$\epsilon = 2$	1.00	0.47	1.00	0.67	0.95	1.00	0.26	1.00	0.89	0.55	0.88	1.00	0.20	0.28	0.77	0.24	0.87
	$\epsilon = 5$	0.97	0.49	1.00	0.67	0.95	1.00	0.37	1.00	1.00	0.54	1.00	1.00	0.30	0.30	0.96	0.30	1.00
FP200	$\epsilon = 0.5$	1.00	0.50	1.00	0.65	1.00	1.00	0.31	1.00	1.00	0.70	1.00	0.90	0.23	0.25	0.78	0.24	0.53
	$\epsilon = 1$	1.00	0.57	1.00	0.69	1.00	1.00	0.36	1.00	1.00	0.70	0.808	1.00	0.23	0.28	0.71	0.27	0.71
	$\epsilon = 2$	1.00	0.56	1.00	0.82	0.95	1.00	0.37	1.00	1.00	0.68	0.80	1.00	0.31	0.30	0.85	0.30	0.87
	$\epsilon = 5$	0.97	0.55	1.00	0.82	0.95	1.00	0.38	1.00	1.00	0.68	0.85	1.00	0.35	0.30	1.00	0.34	1.00
JSD	$\epsilon = 0.5$	0.910	0.750	0.410	0.701	0.300	0.909	0.819	0.431	0.899	0.810	0.240	0.909	0.852	0.470	0.868	0.856	0.540
	$\epsilon = 1$	0.929	0.752	0.310	0.759	0.360	0.963	0.828	0.203	0.900	0.806	0.304	0.908	0.850	0.510	0.849	0.851	0.480
	$\epsilon = 2$	0.912	0.754	0.320	0.762	0.399	0.953	0.829	0.202	0.873	0.807	0.348	0.899	0.843	0.53	0.910	0.804	0.500
length	$\epsilon = 5$	0.886	0.753	0.335	0.760	0.402	0.936	0.828	0.234	0.828	0.807	0.377	0.890	0.804	0.520	0.855	0.800	0.530
EMD src-dst (meters)	$\epsilon = 0.5$	2311	4002	2299	3356	2910	1771	2299	3002	1813	2999	2639	1389	5199	6988	5675	5799	8085	5760
	$\epsilon = 1$	2219	4112	1766	3438	2718	1682	2263	2995	1543	2923	2637	1259	5109	6509	5777	5652	7998	5638
	$\epsilon = 2$	2278	3166	1690	3470	1859	1673	2272	2874	1356	2899	2545	1284	4809	6444	4697	5032	7503	4769
	$\epsilon = 5$	2354	3127	1613	3137	1835	1499	2242	2885	1295	2801	2555	1219	4557	5985	4392	4841	7100	4588
EMD density (meters)	$\epsilon = 0.5$	199	1333	699	40	1303	1002	55	706	621	689	887	670	680	2003	2082	2174	2002	1990
	$\epsilon = 1$	154	1251	709	425	1308	808	46	783	650	480	769	429	405	1850	2081	2185	1999	1909
	$\epsilon = 2$	147	1087	797	471	1260	732	92	737	510	394	782	409	380	1300	987	1233	1677	1189
	$\epsilon = 5$	179	1083	741	451	734	788	135	1300	490	357	723	488	353	1114	1174	1150	1588	1290

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 308–318.

[2] John M. Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Yike Guo and Faisal Farooq (Eds.). ACM, 2867.

[3] Gergely Acs, Luca Melis, Claude Castelluccia, and Emiliano De Cristofaro. 2018. Differentially private mixture of generative neural networks. *IEEE Transactions on Knowledge and Data Engineering* 31, 6 (2018), 1109–1121.

[4] Brett K. Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P. Bhavnani, James Brian Byrd, and Casey S. Greene. 2018. Privacy-preserving generative deep neural networks support clinical data sharing. *bioRxiv* (2018). https://doi.org/10.1101/159756

[5] Alastair R Beresford and Frank Stajano. 2003. Location privacy in pervasive computing. *IEEE Pervasive computing* 2, 1 (2003), 46–55.

[6] Gergely Biczok, Santiago Díez Martínez, Thomas Jelle, and John Krogstie. 2014. Navigating MazeMap: indoor human mobility, spatio-logical ties and future potential. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*. IEEE, 266–271.

[7] Vincent Bindschaedler and Reza Shokri. 2016. Synthesizing Plausible Privacy-Preserving Location Traces. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 546–563.

[8] Dingfan Chen, Tribhuvanesh Orekondy, and Mario Fritz. 2020. GS-WGAN: A Gradient-Sanitized Approach for Learning Differentially Private Generators. *arXiv preprint arXiv:2006.08265* (2020).

[9] Dingfan Chen, Tribhuvanesh Orekondy, and Mario Fritz. 2020. GS-WGAN: A Gradient-Sanitized Approach for Learning Differentially Private Generators. In *Advances in Neural Information Processing Systems (NeurIPS) 2020, December 6-12, 2020, virtual*.

[10] Rui Chen, Gergely Acs, and Claude Castelluccia. 2012. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 638–649.

[11] 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 (2013), 1376.

[12] G. Dimitrakopoulos and P. Demestichas. 2010. Intelligent Transportation Systems. *IEEE Vehicular Technology Magazine* 5, 1 (2010), 77–84.

[13] Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).

[14] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*. Springer, 1–19.

[15] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 486–503.

[16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[17] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[18] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM, 1054–1067.

[19] Marco Fiore, Panagiota Katsikouli, Elli Zavou, Mathieu Cunche, Françoise Fessant, Dominique Le Hello, Ulrich Matchi Aivodji, Baptiste Olivier, Tony Quertier, and Razvan Stanica. 2019. Privacy in trajectory micro-data publishing: a survey. *arXiv preprint arXiv:1903.12211* (2019).

[20] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. 2019. Differentially Private Generative Adversarial Networks for Time Series, Continuous, and Discrete Open Data. In *International Conference, SEC 2019*, Vol. 562. Springer, 151–164.

[21] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. 2012. Next place prediction using mobility markov chains. In *Proceedings of the first workshop on measurement, privacy, and mobility*. 1–6.

[22] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. 2008. Understanding individual human mobility patterns. *nature* 453, 7196 (2008), 779–782.

[23] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. 2010. Differentially private combinatorial optimization. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 1106–1125.

[24] Mehmet Emre Gursoy, Ling Liu, Stacey Truex, Lei Yu, and Wenqi Wei. 2018. Utility-Aware Synthesis of Differentially Private and Attack-Resilient Location Traces. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM, 196–211.

[25] Xi He, Graham Cormode, Ashwin Machanavajjhala, Cecilia M Procopiuc, and Divesh Srivastava. 2015. DPT: differentially private trajectory synthesis using hierarchical reference systems. *Proceedings of the VLDB Endowment* 8, 11 (2015), 1154–1165.

[26] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. 2019. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[27] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. 2013. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1393–1402.

[28] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. 2017. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net.

[29] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data synthesis based on generative adversarial networks. *arXiv preprint arXiv:1806.03384* (2018).

[30] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. 2009. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from https://crawdad.org/epfl/mobility/20090224. https://doi.org/10.15783/C7J010

[31] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. 2018. Knock Knock, Who's There? Membership Inference on Aggregate Location Data. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society.

[32] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision* 40, 2 (2000), 99–121.

[33] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and XiaoFeng Wang. 2017. Privacy Loss in Apple's Implementation of Differential Privacy on MacOS 10.12. *CoRR* abs/1709.02753 (2017). http://arxiv.org/abs/1709.02753

[34] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. 2020. DP-CGAN: Differentially Private Synthetic Data and Label Generation. *CoRR* abs/2001.09700 (2020). arXiv:2001.09700 https://arxiv.org/abs/2001.09700

[35] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially Private Generative Adversarial Network. *CoRR* abs/1802.06739 (2018). arXiv:1802.06739 http://arxiv.org/abs/1802.06739

[36] Fengli Xu, Zhen Tu, Yong Li, Pengyu Zhang, Xiaoming Fu, and Depeng Jin. 2017. Trajectory Recovery From Ash: User Privacy Is NOT Preserved in Aggregated Mobility Data. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. ACM, 1241–1250.

[37] Hui Zang and Jean Bolot. 2011. Anonymization of location data does not work: A large-scale measurement study. In *Proceedings of the 17th annual international conference on Mobile computing and networking*. 145–156.

[38] Xinyang Zhang, Shouling Ji, and Ting Wang. 2018. Differentially Private Releasing via Deep Generative Model. *CoRR* abs/1801.01594 (2018). arXiv:1801.01594 http://arxiv.org/abs/1801.01594

[39] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. 2008. Understanding mobility based on GPS data. In *Proceedings of the 10th international conference on Ubiquitous computing*. 312–321.

[40] Yu Zheng, Xing Xie, Wei-Ying Ma, et al. 2010. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39.

[41] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th international conference on World wide web*. 791–800.

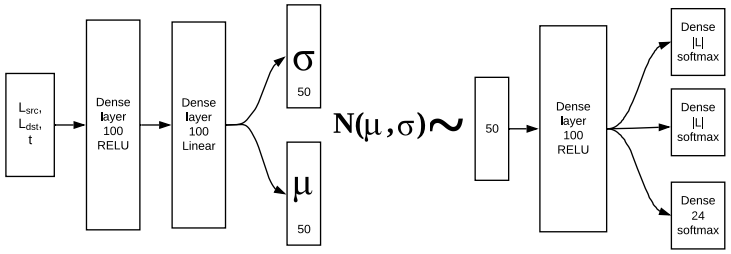
A Additional Results

Trace Generation Complexity: We compute the number of generation steps of a single synthetic trajectory for each scheme as follows. Let us assume that all generative models are precalculated and saved. For a trace t with length $|t|$, DP-Loc takes $Z_{TI} + Z_{TPG}(K) + K^2 \log K + 2 \cdot |t|$ steps, where Z_{TI} and $Z_{TPG}(K)$ are the respective steps of the TI and TPG models, $K^2 \log K$ comes from Dijkstra's algorithm, $2 \cdot |t|$ from the MH and looping parts of the model, and K is the number of top-K locations. As the graphs generated by TPI are precalculated, $Z_{TPG}(K)$ is ignored in he rest of the analysis. Since $|t| = O(K)$, the number of DP-Loc's generative steps is dominated by Dijkstra's algorithm, thus its complexity is $O(K^2 \log K)$.

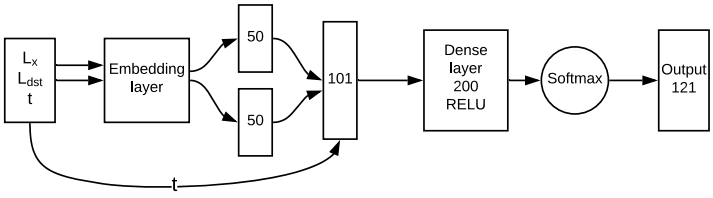
AdaTrace's generation steps depend on two random walks started from the start and destination points. The total number of generation steps accumulates into $2 \cdot m \cdot l$, where m is the size of the grid, and l is the maximal size of the generated trace (l is drawn from a distribution calculated from the original lengths). Since $l \leq m$, AdaTrace's complexity is $O(m^2)$. If the grid is an order of magnitude larger than the number of top-K locations in DP-Loc, the empirical number of steps of AdaTrace can be larger than $K^2 \log K + Z_{TI} + 2 \cdot |t|$, e.g. $m = K^2$.

The expected number of steps in Ngram depends on the underlying data distribution. Evaluating Ngram on the taxi datasets, the average length of the generated traces are 3 steps only.

Experiments were conducted using Tensorflow 2.0 and Python 3.6.9 on a single Linux server with 98GB RAM and 16 cores. Running time is heavily dependent on the size of the input dataset. For the largest, the SF dataset the generation time for Ngram is on average 1 minute, for AdaTrace is 20 second. In case of DP-Loc the trainings of the neural networks take up a lot of time. The overall running time of DP-Loc is approximately 3 hours. However, due to the fact that we are considering offline models, it only has to be done once.

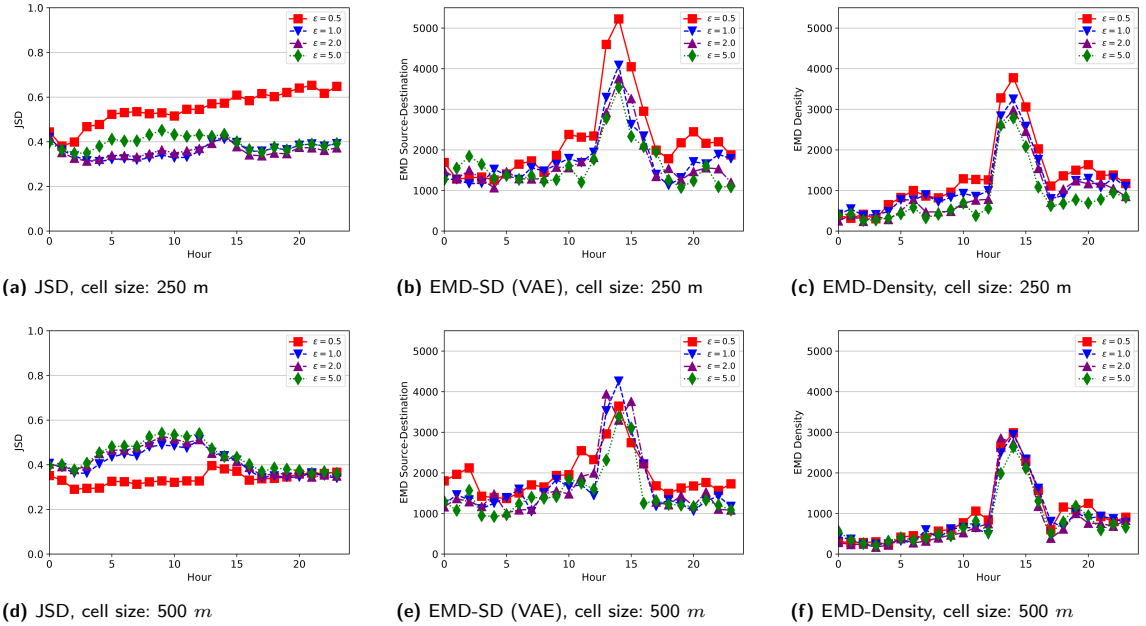


(a) Trajectory Initialization model



(b) Transition Probability Generator model

Fig. 3. The neural network architectures used in DP-Loc



(a) JSD, cell size: 250 m **(b)** EMD-SD (VAE), cell size: 250 m **(c)** EMD-Density, cell size: 250 m

(d) JSD, cell size: 500 m **(e)** EMD-SD (VAE), cell size: 500 m **(f)** EMD-Density, cell size: 500 m

Fig. 4. Performance of our approach on San Francisco dataset depending on the time ($\delta = 4 \cdot 10^{-6}$).