Generating Mobility Trajectories with Retained Data Utility

Chu Cao, Mo Li
Nanyang Technological University, Singapore
{caochu,limo}@ntu.edu.sg

ABSTRACT

This paper presents TrajGen, an approach to generate artificial datasets of mobility trajectories based on an original trajectory dataset while retaining the utility of the original data in supporting various mobility applications. The generated mobility data is disentangled with the original data and can be shared without compromising the data privacy. TrajGen leverages Generative Adversarial Nets combined with a Seq2Seq model to generate the spatial-temporal trajectory data. TrajGen is implemented and evaluated with real-world taxi trajectory data in Singapore. The extensive experimental results demonstrate that TrajGen is able to generate artificial trajectory data that retain key statistical characteristics of the original data. Two case studies, *i.e.,* road map updating and Origin-Destination demand estimation are performed with the generated artificial data, and the results show that the artificial trajectories generated by TrajGen retain the utility of original data in supporting the two applications.

CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; • **Computing methodologies** → **Machine learning**; **Artificial intelligence**; • **Applied computing** → *Publishing*.

KEYWORDS

Mobility Trajectories Generating; Data Utility; Generative Adversarial Nets; Sequence-to-Sequence Learning

ACM Reference Format:
Chu Cao, Mo Li. 2021. Generating Mobility Trajectories with Retained Data Utility. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467158

1 INTRODUCTION

Data mining studies in mobility often face shortage of the supply of data, due mainly to the limits in data production or restrictions on data sharing, *e.g.,* NDA (Non-Disclosure Agreement) documents required by data owners often disable data sharing to unauthorized third parties. The incompleteness and unavailability of datasets hinder most existing data-driven studies. For instance, without having access to the original dataset it is difficult to reproduce or compare results of published research articles even when implementation details or source codes are provided, which significantly undermines the value of research based on the mobility data. One possible solution is to establish standard mobility datasets, such as MNIST [18] and ImageNet [8] datasets in computer vision domains. The efforts in standardizing mobility datasets however faces difficulties in generalization, where different data modes (trajectory, check in/out, location samples, *etc.*), types (vehicle, pedestrian, GPS, cellular, *etc.*), and locations (cities, *etc.*) make it hard to suit one or few standard datasets to various applications or algorithms. In this paper, we study how we can generate artificial datasets of spatial-temporal trajectories based on an existing true trajectory dataset. We want the newly generated artificial datasets to retain the utility of the original true dataset so the artificial datasets can be shared and utilized to perform similar data mining studies that can be performed on the original true dataset.

The main concern of sharing mobility data is potential leakage of private mobility traces of individuals. Many research efforts in recent studies have been made to avoid privacy leakage. [2, 39] adds perturbation to locations in mobility data so as to reduce the probability of the original data being recovered. As an early effort, [2, 39] face dilemma in the trade-off between data utility and the protection to privacy when setting appropriate level of noises added to the data. Recent efforts explore the idea of synthesizing the data by mixing different trajectories and uniformly sampling from them to form new trajectories [24]. Data synthesization relies on real data mixture procedure to form anonymized data, which potentially gives chance to recovering the original data [36]. Latest studies leverage generative adversarial learning to generate mobility data - check-in/out data [30], sequence of vehicle locations [7], and mobility data density distribution [38]. To the best of our knowledge, no existing works are able to achieve our goal of generating spatial-temporal mobility trajectories.

This paper presents TrajGen, an approach to generate new artificial datasets of spatial-temporal mobility trajectories based on an original dataset of true mobility trajectories. TrajGen separates the spatial information and temporal information in forming the new mobility trajectories. The spatial information learning is formulated into an image generation problem by mapping trajectories to image pixels. A Generative Adversarial Net (GAN) [13] is trained to generate images that follow the spatial distributions of the original dataset. The temporal information learning is formulated into sequence-to-sequence mapping problem where locations are extracted from the GAN generated images and fed into Seq2Seq model to infer proper sequences in connecting those locations to trajectories. A fully-connected ANN assigns timestamps to the sequential locations in each trajectory. In such a way new artificial trajectory data are generated based on the underlying distribution of original data, and retain their utility for supporting various mobility applications. Moreover, the generated new artificial datasets are

KDD '21, August 14–18, 2021, Virtual Event, Singapore
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467158

disentangled from the original data, and thus can be shared without leaking privacy of data owners.

In summary, this paper makes the following contributions. (i) We are the first to leverage GAN and Seq2Seq to generate mobility data. Both spatial information and temporal information can be generated in our design. (ii) We implement TrajGen, a novel approach that leverages GAN and Seq2Seq to generate the artificial trajectory dataset. (iii) We conduct extensive experiments and evaluate TrajGen using a real-world dataset. The experimental results suggest that the data generated by TrajGen retain similar utility and statistical features of the original dataset.

The rest of this paper is organized as follows. We review related works in Section 2 and present the preliminaries in Section 3. Section 4 elaborates the design details of TrajGen, and Section 5 presents the evaluations. We conclude this paper in Section 6.

2 RELATED WORK

Mobility data synthesizing. There are two ways of synthesizing mobility data: mixing of existing data or generation of new data. For data mixing, researchers have studied either mixing perturbations with the original data or mixing different true trajectories from the original data. Adding random perturbations [2, 39] synthesizes mobility data by shifting original data samples or adding noises which help improve data privacy. Such methods may disrupt the structure or statistical features of the original data, leading to impaired data utility. In practice, it is difficult to reach the balance between data utility and privacy preserving [19]. [24] proposes to synthesize new mobility trajectories by mixing multiple different trajectories and uniformly selecting from them. This type of work relies on mixing procedures, which potentially gives chances to reconstructing the original data from the mixed data [36]. Generative approaches, from a different perspective, learn the underlying distributions of the original data and generate new mobility data by sampling from the learnt distributions. [19] for the first time envisions the possibility of leveraging GAN to generate mobility data, but gives no detailed solutions. [30] implements a concrete solution based on [19] to generate mobility check-in and check-out data, which essentially mixes random noise to vectors formulated by the original data, and based on the concatenate vectors, generates new artificial data. [38] utilizes GAN to learn the underlying distribution of mobility density on a gridded map, and generates new mobility density in each grid. Neither [30] nor [38] is able to generate complete mobility trajectories of individuals. Authors in [7] propose to generate a sequence of locations via GAIL (a combination of reinforcement learning and GAN) framework, where each location is modeled as an action result (*i.e.,* drivers' choice during driving) such that each trajectory is modeled by Partially Observable Markov Decision Process (POMDP). POMDP, however, derives the next location based solely on the current location, which is not true in practice. On the other hand, [7] focuses only on locations and cannot generate temporal information of trajectories. To the best of our knowledge, the ability of TrajGen in generating spatial-temporal trajectories has not been achieved in previous works.

Other applications of GAN and Seq2Seq techniques. GAN enables to generate discriminator-accepted data samples [13]. Following this, many GAN variants were derived, such as cGAN [23], cycleGAN [21], *etc.* They perform well in image generation tasks, *e.g.,* TGAN [9] in super resolution, dualGAN [37] in image transfer learning. IGAN [41] was derived to manipulate the synthetic images. MuseGAN [10] was proposed to generate music in a different domain. [1, 14] show more GAN applications, which are proved to perform well in image-related tasks. The Seq2Seq model was widely used in neural machine translation tasks [3, 22, 31]. [12] leverages Seq2Seq model to predict bus bunching phenomenon based on smart card data. Authors in [34] use Seq2Seq model to recommend relevant content and involve users in specific conversations. [32] uses Seq2Seq model to predict intrinsically disordered regions in proteins. Combined with RNN processing natural language, GAN is able to generate images based on texts [27]. TrajGen leverages Seq2Seq to infer the temporal aspect knowledge based on default sequences of new locations generated by DCGAN.

3 PRELIMINARY

3.1 Problem Definition

We aim to generate new mobility trajectories based on the underlying distribution of the original mobility data. We believe that in order to retain the data utility, the distributions of generated mobility trajectories and the original data should be approximately and statistically similar with each other.

Definition 1. ***Location.*** *A location is determined by a three-element tuple (latitude, longitude, timestamp). One location can be expressed as* $loc = (lat, lon, t)$*, where* lat *and* lon *are latitude-longitude coordinations, and* t *is the sampled timestamp.*

Definition 2. ***Mobility Trajectory.*** *A mobility trajectory consists of a sequence of* **locations***. The* i^{th} *trajectory can be denoted as* $\tau_i = \{loc^i_1, ..., loc^i_n\}$*, where* loc^i_j *is the* j^{th} *location in* τ_i *sampled at time* t_j*,* $1 \le j \le n$*, and* n *is the total number of locations in* τ_i*.*

Mobility trajectories might be of different forms in reality according to the application background and how locations are sampled. For instance, trajectories can be generated by the timestamped GPS readings from vehicles moving in a city [4, 20] or timestamped cell tower signals [40]. A traffic surveillance system may observe vehicles' movement in a city, generating a trajectory of each vehicle across different junctions [33]. Pedestrian trajectories may give mobility paths of pedestrians [5]. TrajGen can be generally applied to all types of such mobility trajectories.

Definition 3. ***Mobility Map****. The mobility map constrains the movement of subjects (e.g., vehicles, pedestrians). It can be denoted as a graph G(E,V), where E refers to the set of edges that provide connectivity (e.g., road segments for vehicles or pedestrians), and V refers to intersections that provide transitions (e.g., road junctions).*

Mobility map includes road network. To better understand our design, we introduce **travel patterns** of the trajectories.

Definition 4. ***Travel Pattern.*** *A travel pattern is a sequence of geographically adjacent edges, and can be denoted as* $\{e_1 \rightarrow e_2 \rightarrow ... \rightarrow e_n\}$*, where* $e_i \in E$ *is the* i^{th} *edge,* $1 \le i \le n$*, and* n *is the total number of edges in the travel pattern. Each edge in a travel pattern is associated with one or multiple locations in a trajectory* τ*. Travel patterns are derived from the results of map matching algorithm on trajectories. Travel patterns reflect the travel preferences of users.*

We formulate the mobility trajectories generation problem as follows. Given a set of trajectories $\{\tau_1, \tau_2, ..., \tau_n\}$, and mobility map

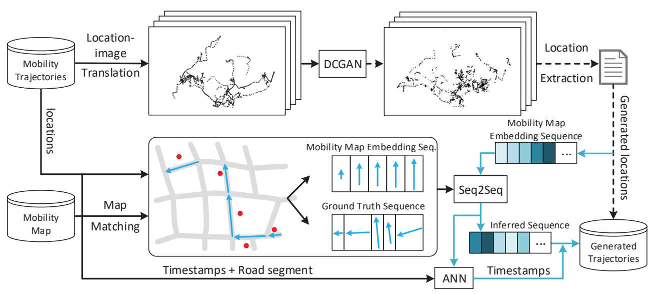


Figure 1: System design of TrajGen. Black arrows are data flow for model training, dashed arrows are data flow of data generation, and blue arrows show data flow of inference.

information, we want to generate a new dataset of mobility trajectories $\{\hat{\tau_1}, \hat{\tau_2}, ..., \hat{\tau_m}\}$, where τ_i and $\hat{\tau_i}$ are the i^{th} original trajectory and the i^{th} newly generated trajectory, respectively.

3.2 Background

In this section, we briefly describe GAN and Seq2Seq models.

Generative Adversarial Nets. GAN consists of two neural networks: one is Generator G and the other is Discriminator D. G and D are trained via competing with each other in a two-player minimax game. In the game, D tries to distinguish the real training data samples (*e.g.,* images) from synthetic data samples generated by G. G tries to generate synthetic data samples from p_z, which is usually a uniform distribution or Gaussian distribution, that can fool D. Ian Goodfellow *et al.* in [13] prove that there exists a global optimal solution for this minimax problem when $p_g = p_{data}$, where p_g is generative distribution (G) and p_{data} is the underlying data distribution. At the optimum point, p_g converges to p_{data}. Meanwhile, generator G is able to imitate p_{data}, generating samples that D cannot distinguish from real training data samples.

Seq2Seq model. Sequence to sequence learning (Seq2Seq) is widely used in neural machine translation, *e.g.,* from English to German. In Seq2Seq, sentences are represented as a sequence of words that are denoted as vectors. Seq2Seq models a conditional probability $p(y|x)$ of mapping the input sequence (*i.e.,* $x_1, x_2, ..., x_n$) into the output sequence (*i.e.,* $y_1, y_2, ..., y_m$). [31] proposes the encoder-decoder framework to learn the conditional probability. Encoder is used to encode each source sequence (*i.e.,* $x = x_1, x_2, ..., x_n$) into an intermediate representation, and based on that, decoder is trained to generate an output sequence (*i.e.,* $y = y_1, y_2, ..., y_m$) one unit by one unit. In practice, encoder and decoder are usually modeled as LSTM (Long-short Term Memory [16]) or GRU (Gated Recurrent Unit [6]). Bahdanau *et al.* propose the attention mechanism [3], a way to control Seq2Seq models to focus on different positions of input sequence. Later studies [22] find attention mechanism powerful when involved in Seq2Seq models.

4 SYSTEM DESIGN

4.1 System Overview

The system design of TrajGen is illustrated in Figure 1. It takes real mobility trajectories and the corresponding mobility map as inputs. TrajGen decouples locations from their temporal sequences in the trajectory, and processes them separately. For trajectory locations,

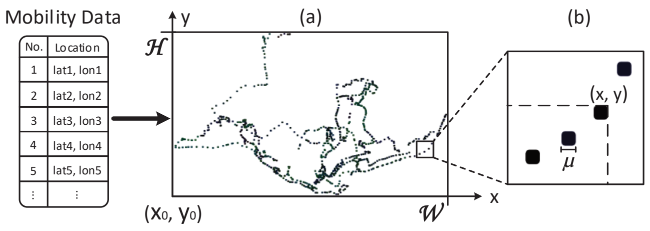


Figure 2: Illustration of location-image translation.

TrajGen translates them into images containing location dots and feeds the images into a DCGAN model to learn the spatial distribution of trajectory locations in original data (upper loop in Figure 1). DCGAN is a deep convolutional GAN optimized for images. The "Generator" in DCGAN can thus generate new images, and TrajGen extracts new trajectory locations from the newly generated images. To capture the sequence and time information of mobility trajectories, TrajGen applies map matching [25] to derive their *travel patterns*. Based on a default sequence information (*i.e.,* way ID) in mobility map, TrajGen obtains a mobility map embedding sequence, *i.e.,* a sequence of road segments, which projects the sequence information into a space spanned by edges in mobility map. Meanwhile, the original trajectory data gives the travel pattern denoted as target sequence. The mobility map embedding sequences are fed into a Seq2Seq model [31], where travel patterns provide the correct order (*i.e.,* labels). Given the input sequence (*i.e.,* mobility map embedding) and target sequence (*i.e.,* travel pattern), a Seq2Seq model can be trained to capture the travel patterns of the original data and based on that generates a new sequence to connect trajectory locations generated by DCGAN. A pre-trained ANN model is used to generate timestamps for all sequenced trajectory locations.

4.2 Location Distribution Learning

Considering the diverse characteristics between spatial information and temporal information, our design of TrajGen decouples locations from sequences and organically merges them at a later stage. We denote all locations without temporal information in a trajectory as $\overline{\tau}$. Therefore, each trajectory τ consists of its $\overline{\tau}$ and t. TrajGen leverages GAN to generate new location data. Since GAN is proved to perform well on image-like datasets [13, 29], TrajGen translates the location information in $\overline{\tau}$ into images.

Location-image translation. Mobility trajectories are collected during the movement of subjects (*e.g.,* vehicles with GPS). The map corresponding to the region where the data was collected is easy to obtain from OpenStreetMap [26]. In the location-image translation, locations are annotated in the image based on their latitudes and longitudes. Such a translation keeps the spatial structure and spatial distribution of the trajectory data. The procedure of location-image translation is illustrated in Figure 2 and elaborated below.

We initialize a 2D space with width of $\mathcal{W}$ (along longitude direction) and height of $\mathcal{H}$ (along latitude direction), and rasterize the space using $\mu * \mu$ square as the minimal unit, where $\mathcal{W}$ and $\mathcal{H}$ can be calculated from the map file. Thus, the resolution of the space is $\lceil(\mathcal{W}/\mu)\rceil * \lceil(\mathcal{H}/\mu)\rceil$. The space here can be regarded as an image and the minimal unit is the pixel in the image. We set up a coordination system in the space with x-axis along the longitude direction and y-axis along the latitude direction, originated at

(x_0, y_0). The position of each square is denoted as the coordinate of its upper right corner. The location loc^i with (lat^i, lon^i) in $\overline{\tau_i}$ can be mapped to a square located at (x, y), where $x = \lceil (lon^i - x_0)/\mu \rceil$, $y = \lceil (lat^i - y_0)/\mu \rceil$. The color of the mapped squares as well as its 8-neighbor surrounding squares in the image are set to black, representing a location. The other squares without mapped locations remain as white. By doing so, each non-temporal trajectory $\overline{\tau_i}$ can be translated into an image img_i. In Figure 2(a), we can see the translated image and the black dots indicate all the locations in $\overline{\tau_i}$. When zooming in, the minimal square is shown in Figure 2(b).

When two locations are close enough, they may be mapped into the same square. This may result from two cases: a) the two locations are sequentially and geographically close; b) the two locations are not consecutive but geographically close. For the first case, we regard them as one pixel on image. For the second case, we are able to differentiate one location from another when considering the sequence and time information.

Training the DCGAN with location images. TrajGen leverages DCGAN to generate new images from which we can extract new locations. For all the non-temporal trajectories $\{\overline{\tau_1}, \overline{\tau_2}, \dots\}$ in one city, we regard them as a set of trajectories sampled from an unknown distribution $\mathcal{P}_{\mathcal{T}}(\overline{\tau})$, where $\mathcal{T}$ means the set of all $\overline{\tau}$ in one city, and $\overline{\tau}$ is a sample from $\mathcal{P}_{\mathcal{T}}(\overline{\tau})$. Due to the characteristics of this unknown distribution $\mathcal{P}_{\mathcal{T}}(\overline{\tau})$, each sampled $\overline{\tau}$ can be well matched with the mobility map of the city. This unknown distribution, however, is not easy to derive mathematically. TrajGen uses the generator in DCGAN to approximate $\mathcal{P}_{\mathcal{T}}(\overline{\tau})$. The location-translated images are fed into DCGAN for training. The set of all the location images reflects the distribution of p_{data} in DCGAN and is used to train DCGAN model to harness its generative distribution p_g. Upon convergence, the generator G in DCGAN has the ability to generate images which can be regarded as samples from $\mathcal{P}_{\mathcal{T}}(\overline{\tau})$. We use generative distribution p_g in DCGAN to approximate the unknown distribution $\mathcal{P}_{\mathcal{T}}(\overline{\tau})$. When given initial conditions, generator G can generate new images. This action of G imitates the process of sampling new trajectories from $\mathcal{P}_{\mathcal{T}}(\overline{\tau})$. TrajGen extracts locations from the generated images.

Location extraction. On each generated images, TrajGen uses Harris corner detector [15] to identify the locations of corners. Each position of identified corners can be converted into a location with latitude and longitude based on the mapping relationship between image and locations in trajectories. These new locations are fed into the Temporal Information Learning module.

4.3 Temporal Information Learning

The newly generated locations have no sequence information available to form a trajectory. We need to recover its travel pattern. TrajGen uses the intrinsic sequence to infer the travel pattern.

To obtain the intrinsic sequence of locations, TrajGen runs the distance-based map matching algorithm to extract the closest road segment of each newly generated location. Therefore, the locations can be represented with their matched road segments. Based on the way ID information in mobility map, each of the extracted road segments have their identities. The way ID information is stable and can be used as the intrinsic sequence.

Mobility map embedding. Mobility map embedding helps project the trajectories from the latitude-longitude space of earth

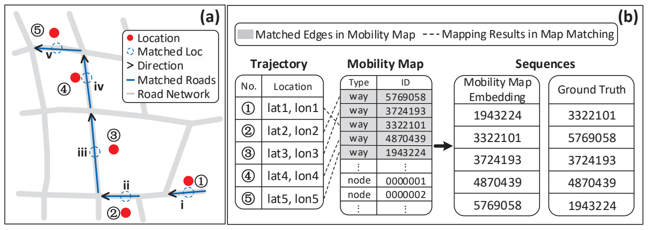


Figure 3: Illustration of Mobility Map Embedding.

into edge space of mobility map. Meanwhile, the sequence information in trajectories is encoded into a sequence of edges (*i.e.,* Travel Pattern), which is illustrated in Figure 3. Figure 3(a) shows a trajectory with five locations denoted as red dots (*i.e.,* from ① to ⑤), and matched locations are denoted as dashed circled dots (*i.e.,* from i to v). Based on matched locations, the matched road segments can be identified, denoted as blue line. The black arrows in Figure 3(a) indicate the moving directions. The method proposed in [25] is used to perform map matching, projecting the trajectory locations into mobility map space. Figure 3(b) shows the conversion from trajectory representation into matched edges (*i.e.,* ways). The sequence of way ID inherits the sequence information in trajectories. As illustrated in Figure 3(b), locations from ① to ⑤ are matched with the five ways highlighted with gray in mobility map table. TrajGen extracts their IDs to form two sequences. One is the mobility map embedding sequence and the other one is the travel pattern whose order follows the order of locations (*i.e.,* from i to v). Given the travel pattern and matching relation between road segments and locations, the original sequence information of locations in trajectories can be recovered.

Training the Seq2Seq model. We first train a Seq2Seq model with sequence information in real trajectories. When given a default sequence of new locations, the well trained Seq2Seq model infers the travel pattern. In training phase, the source sequence is a series of way IDs in a default order and the target sequence is a series of way IDs in the travel pattern. TrajGen infers the target sequence (*i.e.,* travel pattern) based on a mobility map embedding. To get the default sequence, we sort the way IDs in target sequence based on the value of each way ID. As shown in Figure 3(b), the ground truth (*i.e.,* {3322101, 5769058, 3724193, 4870439, 1943224}) inherits the sequence information of trajectory and the default sequence column (*i.e.,* {1943224, 3322101, 3724193, 4870439, 5769058}) is the ascending sort of way ID. Each trajectory provides a training sample, *i.e.,* a pair of source sequence and target sequence, for Seq2Seq model. Seq2Seq model encodes the source sequence into a vector and decodes the vector into a target sequence. Through such an encoder-decoder structure, Seq2Seq constructs the conditional probability of mapping source sequence to target sequence. It learns the travel pattern of subjects (*e.g.,* vehicles). To infer the travel pattern of generated locations, Seq2Seq selects the most likely order to traverse the locations based on the learned knowledge. The rationale behind is that there are limited travel patterns in one specific city, which provides us an opportunity to learn the travel patterns of subjects in a city from historical data.

Timestamp inference. Now the generated data consist of locations with sequence information. TrajGen generates timestamps

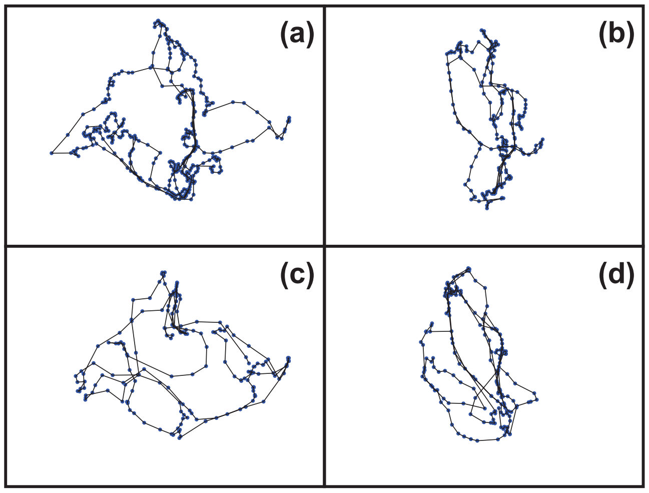


Figure 4: Visualized examples of original trajectories (a, b) and TrajGen generated trajectories (c, d).

for the locations by inferring the initial timestamp t_0 as the sampling rate (*i.e.,* $1/\Delta t$) is fixed in the same dataset. If t_0 is known, the following i^{th} timestamp can be calculated as $t_i = t_0 + i \times \Delta t$. To infer t_0, we build a non-linear ANN model. The input of ANN consists of the length of a trajectory and the matched road segment of its first location. The desired output of ANN model is the time slot when the first location is sampled with the highest probability. Each time slot in output is 15 seconds, and we have 5,760 time slots during one day. The model captures how the initial locations of trajectories distribute in the original mobility trajectories.

Figure 4 shows the visualized trajectories from the original dataset and the new dataset generated by TrajGen. The blue points in Figure 4 represent the locations (downsampled by 5 times for visualization purpose) which corresponds to the inputs/output of DCGAN and black lines are the direct line connection between consecutive locations, which includes the input/output of Seq2Seq. Trajectories in Figure 4(a, b) are sampled from the original data, and trajectories in Figure 4(c, d) are generated by TrajGen. We deliberately select the trajectories covering similar areas for comparison purpose. The training phase does not involve the mobility map information. The mobility map information, however, is implicitly included in mobility trajectories as they are recorded by vehicles moving on road network. As a result, the generated trajectories in Figure 4(c, d) follow the road structure of Singapore, when compared with the road network of Singapore (main road network of Singapore can be found in Appendix Figure 16).

5 EVALUATION

5.1 Experimental Setup

Performance criteria. We evaluate TrajGen at both a macro-level and a micro-level. At the macro-level, we mainly consider the overall distribution of the mobility dataset. We leverage heat map to visualize the location distribution of trajectories. Specifically, we divide the base map into 800 (determined by city size) regions with each region covering a $1km \times 1km$ area. Inside each region, we count the number of locations during one day. The heat map indicates the overall distribution of locations across the whole city from a qualitative perspective. Then, we quantitatively calculate the cosine similarity between heat maps. The heat map can be regarded as an 800-dimensional vector $(x_1, x_2, ..., x_{800})$, where x_i is the number of observed locations in the i^{th} region.

At a micro-level, we assess the statistical features via multiple metrics, *i.e.,* distance from a location to its mapped road segment, travel distance, the proportion of ordinary ways and express ways, and trajectory covered area. We calculate the travel distance d_{τ_i} of a trajectory $\tau_i = \{loc^i_1, ..., loc^i_m\}$ by $d_{\tau_i} = \sum_{k=1}^{m-1} D_{mm}(loc^i_k, loc^i_{k+1})$, where $D_{mm}(a, b)$ is the length of traveled routes on mobility map from points a to b. The distance between a location to its matched road segment can be computed as $D_{gc}(loc, \widehat{loc})$, where $D_{gc}(a, b)$ is the great circle distance from point a to point b on the earth, loc is the location, and $\widehat{loc}$ is the closest point on the matched road segment to loc. The travel covered area of τ_i is defined as the area of the smallest rectangle covering τ_i and is calculated as: $s_{\tau_i} = w_{\tau_i} \times h_{\tau_i}$, where $h_{\tau_i} = D_{gc}(loc_{bl}, loc_{tl})$, $w_{\tau_i} = D_{gc}(loc_{bl}, loc_{br})$ and loc_{bl}, loc_{br} are computed below. Given τ_i, its k^{th} location is $loc^i_k = (lat^i_k, lon^i_k, t^i_k)$, and we define lat_- as the minimal latitude, lon_- as the minimal longitude, lon_+ as the maximal longitude among all the locations in τ_i. Therefore, we have $loc_{bl} = (lat_-, lon_-)$, $loc_{tl} = (lat_+, lon_-)$, and $loc_{br} = (lat_-, lon_+)$. These features explicitly reflect whether the generated mobility trajectories maintain the statistical characteristics of the original data.

Benchmarks. We compare TrajGen with three baselines: two traditional geomasking methods and one variant of TrajGen.

Random Perturbation (RP). RP is a geomasking approach [2], where each location is displaced in latitude-longitude space by a randomly determined distance and direction. A distance threshold is typically set to allow the maximal displacement distance. $2km$ is effective for protecting geo-privacy [11] and a $\pm 10s$ shift is randomly added on timestamp.

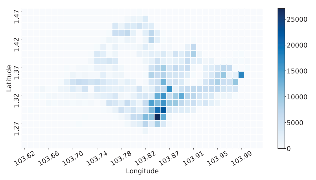
Gaussian Perturbation (GP). GP displaces the original location with a distance sampled from a Gaussian distribution in a random direction [39]. The mean of Gaussian distribution is the original location itself and standard deviation is 0.05 because such a standard deviation setting protects the geo-privacy well [11]. We add time shift on timestamps. The shift is sampled from a Gaussian distribution (standard deviation is 10 and mean is 0).

TrajGen-v is a variant of TrajGen. In TrajGen, the location information used for DCGAN training and sequence information used for Seq2Seq training are extracted from the same dataset, while in TrajGen-v, they are from different datasets. Its performance validates our design where spatial information and temporal information can be separated in mobility trajectory generation tasks.

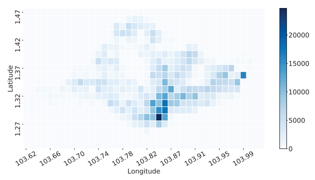
5.2 Evaluation Results

Given the original dataset, we randomly select 500 trajectories without replacement and denote them as base data, while the remaining trajectories are denoted as raw data. Based on TrajGen and benchmark schemes, we have four generated datasets: TrajGen dataset, TrajGen-v dataset, RP dataset and GP dataset.

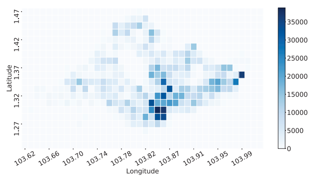
Spatial Distribution. We calculate the location distribution similarity between the original data and generated data in one day. The distribution of base data is denoted as $\mathcal{X}_0 = (x^{(0)}_1, x^{(0)}_2, ..., x^{(0)}_{800})$.



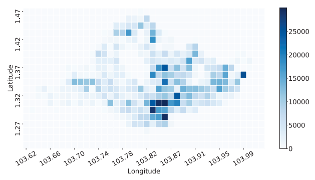
(a) Base data: location distribution of randomly sampled 500 taxis' trajectories during one day.



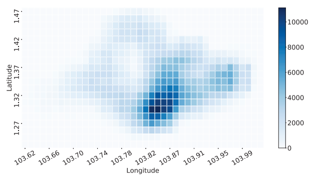
(b) Location distribution of randomly sampled trajectories from raw data. It has a cosine similarity of 0.976 with Figure 5(a).



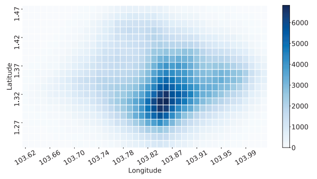
(c) Location distribution of randomly sampled trajectories from TrajGen. It has a cosine similarity of 0.956 with Figure 5(a).



(d) Location distribution of randomly sampled trajectories from TrajGen-v. It has a cosine similarity of 0.898 with Figure 5(a).



(e) Location distribution of randomly sampled trajectories from RP data. It has a cosine similarity of 0.814 with Figure 5(a).



(f) Location distribution of randomly sampled trajectories from GP data. It has a cosine similarity of 0.775 with Figure 5(a).

Figure 5: Visualized location distribution of different data sources. Each grid in the figure represents an approximate $1km \times 1km$ square area in the real world.

We then randomly select 500 trajectories from the raw dataset and each of the four generated datasets. The location distribution of one dataset can be denoted as $\mathcal{X}_s = (x_1^{(s)}, x_2^{(s)}, ..., x_{800}^{(s)})$, where s indicates data source. We compute the cosine similarity between $\mathcal{X}_0$ and each of $\mathcal{X}_{raw}$, $\mathcal{X}_{\text{TrajGen}}$, $\mathcal{X}_{\text{TrajGen}-v}$, $\mathcal{X}_{RP}$, and $\mathcal{X}_{GP}$.

We repeat the sampling and similarity computation procedures 500 times and record the results. Figure 6 shows the CDF of cosine similarities. The similarity between $\mathcal{X}_0$ and $\mathcal{X}_{raw}$ is in the range of 0.946 and 0.976, while the counterparts of $\mathcal{X}_{\text{TrajGen}}$ and $\mathcal{X}_{\text{TrajGen}-v}$ are in (0.91, 0.966) and (0.893, 0.958), respectively. For RP data and GP data, their similarities are in the range of (0.797, 0.866) and (0.755, 0.822), which are far away from the position of raw data. TrajGen dataset have closer location distribution to raw data, indicating that the trajectories generated by TrajGen have similar overall location distribution with the original data.

To better illustrate the detailed difference, we visualize some of the location distributions in Figure 5. Figure 5(a) is the visualized base data. Figure 5(b) shows the distribution of 500 sampled trajectories from raw data, which is the largest cosine similarity value 0.976. Figure 5(c) has a 0.956 cosine similarity with Figure 5(a), formed by trajectory locations from TrajGen data. Figure 5(d) is the result from TrajGen-v dataset, with the similarity of 0.898. The heat maps of locations sampled from RP data and GP data are provided in Figure 5(e) (with 0.814 similarity) and Figure 5(f) (with 0.775 similarity), respectively. Figure 5 suggests TrajGen maintains the overall trend of data distribution. For example, south central area (*i.e.,* city centre) and east most area (*i.e.,* airport) are the densest regions. This suggests TrajGen is able to generate mobility trajectories that have similar location distribution with the original data.

Temporal Distribution. To evaluate the generated temporal information, we compare the location distribution in different time slots. If the temporal information is properly generated, the location distribution during specific time slots should be close to that of the original data. We use the same setting with spatial distribution, and calculate the cosine similarity between $\mathcal{X}_0$ and each of $\mathcal{X}_{raw}$, $\mathcal{X}_{\text{TrajGen}}$, $\mathcal{X}_{\text{TrajGen}-v}$, $\mathcal{X}_{RP}$, and $\mathcal{X}_{GP}$. The only difference is the time span. In spatial distribution setting, each of the distribution vector $\mathcal{X}$ includes locations in one day while we use the location distribution in each hour here. Figure 7 illustrates the results. Overall, the performances of TrajGen and TrajGen-v are closer to raw data than that of RP data and GP data. This indicates that TrajGen is able to generate artificial trajectories that have similar distributions of temporal information with the original data.

Statistical Features. TrajGen generates mobility trajectories, which carry the characteristics. It is infeasible to enumerate all of the characteristics. We take the following four as examples.

Distance from location to matched road segment. We randomly selected 50 trajectories from raw data and generated datasets. The average length of sampled trajectories is 1,750. After map matching, we have $(location, road)$ pairs. Great circle distance $D_{gc}(loc, \widehat{loc})$ is used to measure the distance from location to matched segment. We show the statistical result of calculated distances in Figure 8. Theoretically, the closer the CDF line is to the top left corner, the better, which indicates that the locations are close to road segments. Figure 8 depicts that the locations of TrajGen data are similar to that of raw data. Among all the distances in raw dataset, 50% of them are less than $4.2m$. The counterparts of TrajGen data and TrajGen-v data are $5.0m$ and $4.4m$, respectively. For RP data and GP data, however, their median distances are $8.5m$ and $15.6m$, which are farther than that of TrajGen datasets. This suggests that TrajGen is able to generate trajectories that have similar location distribution (*i.e.,* closeness to roads) with the original data.

Travel distance. 500 trajectories are randomly selected from raw data and each of the generated datasets. The travel distance of each trajectory can be computed by travel distance equation in § 5.1. We run map matching and extract travel routes of trajectories. Figure 9 shows the CDF of travel distance in different datasets. The distributions of TrajGen datasets are closer to that of raw dataset. Among the sampled trajectories in raw data, TrajGen data, and TrajGen-v data, 50% of them traveled less than $307.6km$, $414.9km$ and $419.3km$, respectively. In RP data and GP data, the median travel distances are $683.5km$ and $967.7km$, which are much longer than TrajGen datasets. In terms of the longest traveled distance, it is around $800km$ in TrajGen data, which is close to $790km$ of raw data. In RP and GP datasets, their longest travel distance is $1336km$ and $1880km$, which are much longer than raw data. The perturbations from RP and GP might destroy the spatial structure of trajectories. After map matching, the trajectory is matched with complicated detours on roads, leading to a longer travel distance. The results in

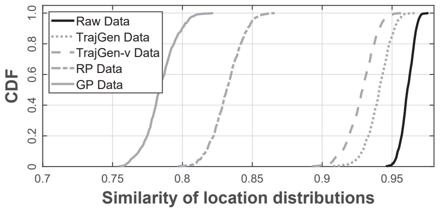


Figure 6: CDF of similarities across different data sources.

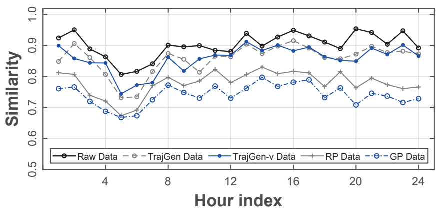


Figure 7: Averaged similarity of location distributions in each hour.

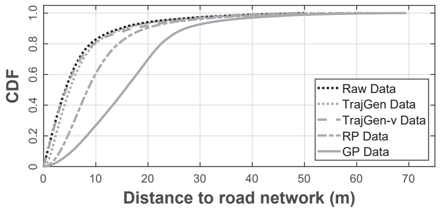


Figure 8: CDF of distance from location to map matched road segment.

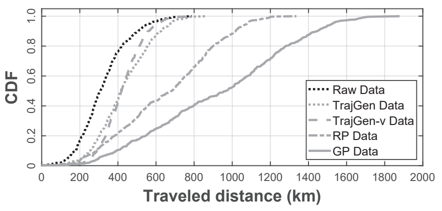


Figure 9: CDF of travel distances of randomly sampled trajectories from different data sources.

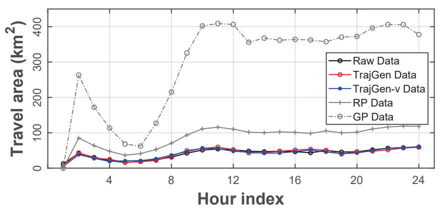


Figure 10: Averaged travel covered area of randomly sampled trajectories from different datasets during each hour.

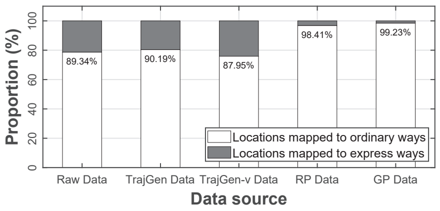


Figure 11: The proportion of ordinary way and express way in different data sources.

Figure 9 demonstrate that TrajGen is able to generate artificial data that have similar travel distance with the original data.

Travel covered area in each hour. The one day trajectory usually covers the whole city which has a fixed area size. Therefore, the travel covered area during one day is close to the city size (*i.e.,* 1,250 km^2). We divide the sampled trajectories into 24 parts based on time and calculate the averaged travel covered area during each hour. Figure 10 depicts the result, showing that TrajGen and TrajGen-v data are close to the raw data. For RP data and GP data, some of the trajectories are largely distorted, resulting in a bigger traveled area comparing to raw data. The results in Figure 10 show that the generated mobility trajectories by TrajGen have similar distribution of travel covered area with the original data.

Proportion of ordinary way and express way. After map matching, the matched routes of the selected trajectories are known. Based on the #*tag* information in OpenStreetMap [26], we count the road segments as expressways when their #*tag* information includes "expressway". Otherwise, we regard the road segments as ordinary ways. Figure 11 gives the proportion of ordinary way and express way based on trajectories in raw dataset and each of generated datasets. Majority ($> 85\%$) of the matched road segments are ordinary ways in all the datasets. Express ways account for around 10% in raw data (10.66%), TrajGen data (9.81%), and TrajGen-v dataset (12.05%). The proportion dramatically decrease to 1% for RP and GP datasets, mainly because RP and GP destroy the trajectory structure and lead to failures in map matching. The results of Figure 11 suggests the generated artificial trajectories by TrajGen have similar proportion of ordinary way and express way with the original data.

5.3 Case Study

Other than the analysis of statistical features, we conduct two case studies: road map updating and origin-destination travel demand estimation, which are classical and well-performed applications on mobility data, to assess the utility of generated mobility trajectories. We use Singapore Taxi data in this section.

Case Study 1: Road Map Updating. It aims to discover uncharted road segments in digital maps. The first phase is map matching, where trajectories are processed to extract the locations that cannot be matched to existing road networks. Those unmatched locations are likely to be recorded by vehicles moving on uncharted routes in existing digital maps. When more and more unmatched locations are extracted, they are clustered based on Hausdorff distance [35] between trajectories. In each cluster, the threshold is set to 2, which is the minimal number of trajectories to derive road segments. Road map updating algorithm can derive a route based on those unmatched locations in the same cluster.

We leverage the algorithm in [35] to do road map updating. To evaluate the utility of generated dataset, we select a central region $\mathcal{R}$, which is the road densest region in existing maps, and deliberately remove some of the road segments inside $\mathcal{R}$. In total, there are 3,759 road segments in $\mathcal{R}$ and 2,025 road segments are randomly removed. Such removed road segments can be ground truth. From the raw dataset and each of the generated datasets, we randomly selected 500 trajectories that have passed through $\mathcal{R}$ such that each trajectory could contribute map updating. We feed each trajectory into map updating algorithm so as to compare their results. As we have ground truth, precision and recall are used to evaluate the performance. Precision is the proportion of truly discovered road segments among all the derived road segments. Recall is the proportion of truly discovered road segments among all the removed road segments. In practice, a derived road segment has its own direction and length. We compare the direction and length of derived segments with road segments in ground truth. For one derived segment and one ground truth segment, if they have intersected, the direction deviation between them is less than 45° and their length difference is less than $100m$, we qualify the derived

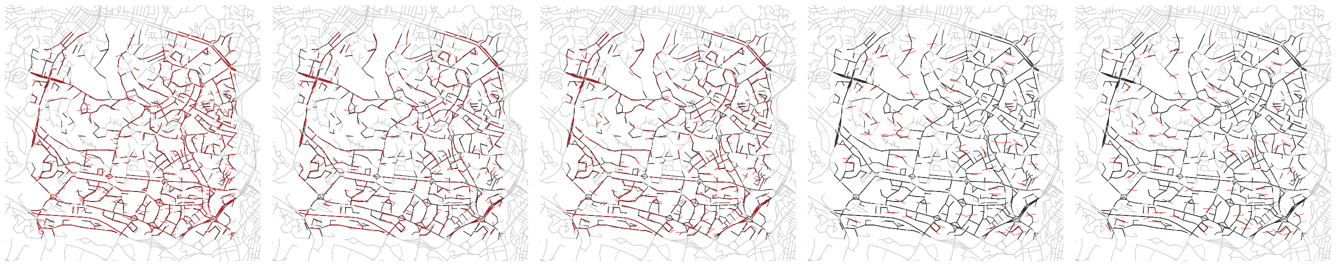

(a) Results of Raw data. (b) Results of TrajGen data. (c) Results of TrajGen-v data. (d) Results of RP data. (e) Results of GP data.

Figure 12: Results of road map updating. Gray lines are road networks (3,759 in total), black lines are removed road segments (2,025 in total), and red lines are newly found road segments from raw data and generated mobility datasets.

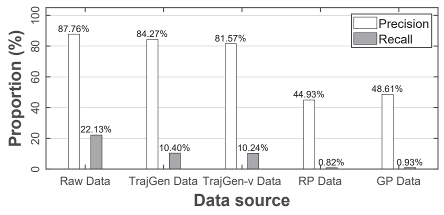


Figure 13: Precision and recall in road map updating.

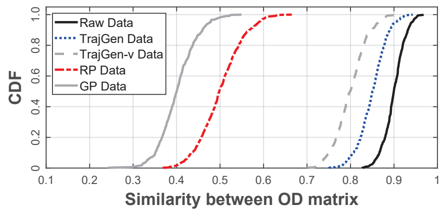


Figure 14: CDF of similarities between OD matrices.

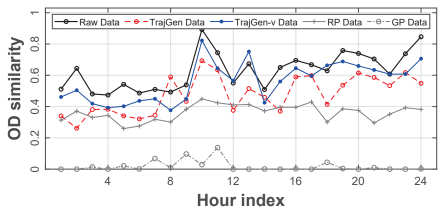


Figure 15: Averaged OD matrix similarity in each hour.

segment as truly discovered road segments. The map updating results are shown in Figure 12, where gray lines are the original road network, black lines are removed road segments, and red lines are the derived road segments using different datasets.

Figure 13 gives the precision and recall. 948 road segments are discovered in raw data, and 832 (87.76%) are truly discovered road segments. The precision for TrajGen and TrajGen-v dataset are 84.27% and 81.57%, respectively. The number of total discovered road segments in TrajGen (464) and TrajGen-v (472) are less than raw data, leading to a lower recall. RP data found 69 segments and only one is truly discovered. For GP data, there is one segment qualified as truly discovered road segment. Low recall value results from limited number of trajectories used for map updating. The comparable precision of TrajGen data validates that the generated mobility trajectories by TrajGen is able to achieve similar performance with the original data in road map updating application.

Case Study 2: Origin-Destination Demand Estimation. We conduct Origin-Destination demand estimation (OD estimation for short). OD estimation is a classical application used to describe the travel demand of citizens. It aims to calculate the OD matrix. Each value of the matrix indicates the travel demand from one location (*i.e.,* origin) to another (*i.e.,* destination). The taxi OD demand is defined as the total number of taxi requests from the original region to the destination region in each time interval [28].

We partition the map into N small regions based on latitude and longitude, where $N = H \times W$, H and W are the height and width of the city grid map respectively. We denote the OD matrix as a 2D matrix $X \in R^{N \times N}$. Specifically, $X(r_i, r_j)$ is the taxi demand from origin region r_i to destination region r_j, where r_i and r_j are index of regions in partitioned city map, and $0 \leq r_i \leq N-1$, $0 \leq r_j \leq N-1$.

As one trajectory may include more than one journey (*i.e.,* multiple OD paris included), we need to derive the origin and destination of each journey. From the gasoline cost perspective, we assume that drivers prefer to moving in a small region or staying at pick-up stops when the taxi is empty. Therefore, such locations generated by moving in a small region or staying at taxi pick-up stops could be used to separate journeys in a trajectory. For each trajectory τ_i, we run a DBSCAN [17] algorithm to determine the core locations. The radius is 1,000m and the point number threshold is 20. Core locations in the results of DBSCAN are separators to cut τ_i into multiple journeys. The first and the last locations of each journey form one OD pair. Therefore, we can extract the corresponding origin region (*e.g.,* r_o) and destination region (*e.g.,* r_d) for a journey, adding one to $X(r_o, r_d)$ in the OD matrix.

We evaluate whether TrajGen retains the utility in OD demand estimation by comparing the OD matrix. We randomly pick out 500 trajectories from raw dataset and each of the generated datasets. We calculate the OD matrix of different datasets, denoted as X^{base}, X^{raw}, X^{TrajGen}, $X^{\text{TrajGen}-v}$, X^{RP} and X^{GP}. Each OD matrix can be reshaped into a vector consisting of $N \times N$ elements. We compute the cosine similarity between the vector of base data and other corresponding vectors. Such procedures are repeated 500 times and Figure 14 gives the statistical results. The OD matrix similarity between base data and raw dataset is in the range of 0.83 and 0.97, while the counterparts of TrajGen and TrajGen-v datasets are in (0.75, 0.94) and (0.7, 0.9), respectively. For RP data and GP data, their similarity performances are in the range of (0.37, 0.66) and (0.24, 0.55), which are far away from the distribution of raw data. The results of Figure 14 suggest that the newly generated trajectories by TrajGen is able to achieve similar performance with the original data in OD travel demand estimation.

We evaluate OD demand during each hour, which indicates the spatial-temporal characteristics of generated mobility trajectories. We extract the corresponding trajectories in each hour and calculate the OD matrix. For the OD matrix in same hour, we compute the similarity between X_i^{base} and each of $\{X_i^{raw}, X_i^{\text{TrajGen}}, X_i^{\text{TrajGen}-v}, X_i^{RP}, X_i^{GP}\}$ 500 times, and record the averaged similarities in each hour, where $i \in \mathbf{Z}$ is hour index, $1 \leq i \leq 24$. Figure 15 gives the results of OD matrix similarities in different hours. For most of the hours, the performances of trajectories generated by TrajGen and TrajGen-v are closer to the original data. This indicates that TrajGen is able to generate artificial mobility trajectories owing similar OD demand with the original data across different hours. This also validates our design where spatial information and temporal information can be decoupled.

6 CONCLUSION

We present TrajGen, an approach to generate artificial dataset of mobility trajectories with retained data utility. The generated mobility trajectories are disentangled with the original data and can be shared without privacy issue. TrajGen decouples spatial information from temporal information in mobility data. Experimental results on a real-world taxi dataset show that the generated artificial mobility trajectories by TrajGen follow similar distribution with the original data. The results of road map updating and OD travel demand estimation suggest that TrajGen is able to retain the utility of the original data in mobility data studies.

ACKNOWLEDGMENTS

This research is supported by NRF SDSC grant SDSC-2019-001, Singapore MOE Tier 1 RG18/20, and NTU CoE Grant 04INS000269C130. We thank Singapore Land Transport Authority (LTA) for sharing with us the taxi data. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of funding agencies.

REFERENCES

[1] Hamed Alqahtani, Manolya Kavakli-Thorne, and Gulshan Kumar. 2019. Applications of generative adversarial networks (gans): An updated review. *Archives of Computational Methods in Engineering* (2019), 1–28.
[2] Marc P Armstrong, Gerard Rushton, and Dale L Zimmerman. 1999. Geographically masking health data to preserve confidentiality. *Statistics in medicine* 18, 5 (1999), 497–525.
[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473* (2014).
[4] Chu Cao, Zhenjiang Li, Pengfei Zhou, and Mo Li. 2018. Amateur: Augmented reality based vehicle navigation system. *ACM IMWUT* 2, 4 (2018), 1–24.
[5] Chu Cao, Zhidan Liu, Mo Li, Wenqiang Wang, and Zheng Qin. 2018. Walkway discovery from large scale crowdsensing. In *IPSN*. IEEE, 13–24.
[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv:1406.1078* (2014).
[7] Seongjin Choi, Jiwon Kim, and Hwasoo Yeo. 2020. TrajGAIL: Generating Urban Vehicle Trajectories using Generative Adversarial Imitation Learning. *arXiv:2007.14189* (2020).
[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*. IEEE, 248–255.
[9] Zihan Ding, Xiao-Yang Liu, Miao Yin, and Linghe Kong. 2019. Tgan: Deep tensor generative adversarial nets for large image generation. *arXiv:1901.09953* (2019).
[10] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. 2018. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *AAAI*, Vol. 32.
[11] Song Gao, Jinmeng Rao, Xinyi Liu, Yuhao Kang, Qunying Huang, and Joseph App. 2019. Exploring the effectiveness of geomasking techniques for protecting the geoprivacy of Twitter users. *Journal of Spatial Information Science* (2019).
[12] Zengyang Gong, Bo Du, Zhidan Liu, Wei Zeng, Pascal Perez, and Kaishun Wu. 2020. SD-seq2seq: A Deep Learning Model for Bus Bunching Prediction Based on Smart Card Data. In *ICCCN*. IEEE, 1–9.
[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*.
[14] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. 2020. A review on generative adversarial networks: Algorithms, theory, and applications. *arXiv:2001.06937* (2020).
[15] Christopher G Harris, Mike Stephens, et al. 1988. A combined corner and edge detector.. In *Alvey vision conference*, Vol. 15. Citeseer, 10–5244.
[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[17] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. 2011. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1, 3 (2011), 231–240.
[18] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998).
[19] X. Liu, Hanzhou Chen, and Clio Andris. 2018. trajGANs : Using generative adversarial networks for geo-privacy protection of trajectory data (Vision paper).
[20] Zhidan Liu, Pengfei Zhou, Zhenjiang Li, and Mo Li. 2018. Think Like A Graph: Real-Time Traffic Estimation at City-Scale. *TMC* (2018).
[21] Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang. 2018. Attribute-guided face generation using conditional cyclegan. In *ECCV*.
[22] Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*. 76–79.
[23] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv:1411.1784* (2014).
[24] Mehmet Ercan Nergiz, Maurizio Atzori, and Yucel Saygin. 2008. Towards trajectory anonymization: a generalization-based approach. In *ACM SIGSPATIAL*.
[25] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *ACM SIGSPATIAL*. ACM, 336–343.
[26] OpenStreetMap. 2017. Open Street Map. Retrieved June 12, 2017 from https://www.openstreetmap.org
[27] Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. 2019. Mirrorgan: Learning text-to-image generation by redescription. In *CVPR*. 1505–1514.
[28] Zhilin Qiu, Lingbo Liu, Guanbin Li, Qing Wang, Nong Xiao, and Liang Lin. 2019. Taxi origin-destination demand prediction with contextualized spatial-temporal network. In *ICME*. IEEE, 760–765.
[29] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434* (2015).
[30] Jinmeng Rao, Song Gao, Yuhao Kang, and Qunying Huang. 2020. LSTM-TrajGAN: A Deep Learning Approach to Trajectory Privacy Protection. (2020).
[31] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*.
[32] Yi-Jun Tang, Yi-He Pang, and Bin Liu. 2020. IDP-Seq2Seq: identification of intrinsically disordered regions based on sequence to sequence learning. (2020).
[33] Panrong Tong, Mingqian Li, Mo Li, Jianqiang Huang, and Xiansheng Hua. 2021. Large-Scale Vehicle Trajectory Reconstrubtion with Camera Sensing Network. In *ACM MobiCom*.
[34] Johnny Torres, Carmen Vaca, Luis Terán, and Cristina L Abad. 2020. Seq2Seq models for recommending short text conversations. *Expert Systems with Applications* 150 (2020), 113270.
[35] Yin Wang, Xuemei Liu, Hong Wei, George Forman, Chao Chen, and Yanmin Zhu. 2013. Crowdatlas: Self-updating maps for cloud and personal use. In *MobiSys*.
[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 *WWW*. 1241–1250.
[37] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*. 2849–2857.
[38] Dan Yin and Qing Yang. 2018. GANs based density distribution privacy-preservation on mobility data. *Security and Communication Networks* 2018 (2018).
[39] Paul A Zandbergen. 2014. Ensuring confidentiality of geocoded health data: assessing geographic masking strategies for individual-level data. *Advances in medicine* 2014 (2014).
[40] Pengfei Zhou, Yuanqing Zheng, and Mo Li. 2012. How long to wait? Predicting bus arrival time with mobile phone based participatory sensing. In *ACM MobiSys*. 379–392.
[41] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. 2016. Generative visual manipulation on the natural image manifold. In *ECCV*. Springer.

A REPRODUCIBILITY

A.1 Data Preparation

Mobility trajectory dataset. We use the mobility trajectory data of taxis in Singapore (data sample provided in a GitHub repository [1]). The mobility dataset are collected from 17,610 taxis during four months (*i.e.,* 1 July to 31 August in year 2015, and 12 March to 12 May in year 2016) in Singapore. The vehicles are equipped with sensors to record the status (*e.g.,* license ID, latitude, longitude, occupied or not, *etc.*) every 30 seconds. The dataset contains millions of records in total. Each record in the dataset has a location (*i.e.,* latitude, longitude) as well as a sampling timestamp.

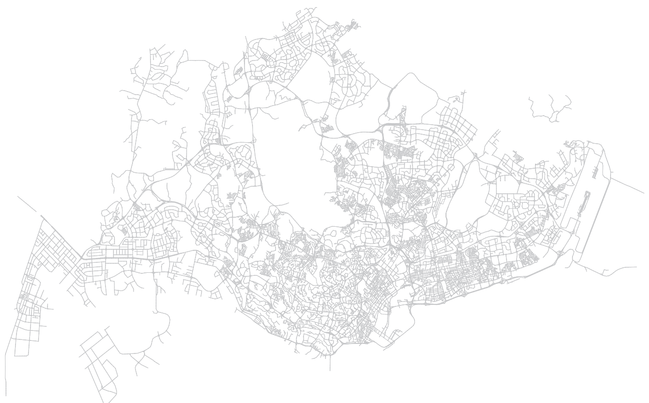


Figure 16: Main structure of Singapore road network used in this paper. (We ignore some detailed road segments and only show partial of the whole map for a better visualization.)

The city base map. The base map used in this paper is fetched from an online open-sourced map provider OpenStreetMap, which is the largest crowd-sourced map project with more than 2 million users registered to contribute. Figure 16 shows the main road network of Singapore. We use the OSM map extracted on the day of 2016-08-20 as the city base map (provided in our GitHub repository) for map matching and evaluations just because our mobility dataset were mainly collected during the year of 2016. The digital map in total consists of 81,471 intersections and 56,914 ways. Its covered area is a rectangular region of 45km × 25km, approximately.

A.2 Implementation

We implement majority of the algorithms mentioned in TrajGen in Python and partially in Matlab. The proposed approach is tested on a workstation that has a 3.3GHz i9-9820X CPU (equipped with 10 cores and 20 threads) from Intel, 128G memory and four 2080ti GPU cards from Nvidia. The training data are stored on a 2TB SSD disk. We leverage some of the open-sourced Python libraries (*e.g.,* NetworkX, NumPy, TensorFlow, PyTorch, *etc.*) to compute, train model, and visualize the results on a basic road map.

Given the original dataset of mobility trajectories, we conduct a location-image translation module to convert each of the trajectory in one day into images. Algorithm 1 show the pseudo code of how we conduct the translation. It is noted that the digital map information is used to determine the size of the image and the points or edges information in maps are not involved. In algorithm 1, from line 3 to line 6, the size of converted image is determined, which is roughly the same size as the covered area of the digital map. Therefore we have a mapping relationship between locations and the converted image. We annotate the pixels where there is an observed location as black, and so do its 8-neighborhood pixels, which is from line 10 to line 12 in the algorithm 1.

ALGORITHM 1: Location-image translation.

```
1  Input: mobility trajectories $\{\tau_1, \tau_2, ...\}$, side length of minimal unit square $\mu$, city digital map $M$;
2  Output: images containing spatial information;
3  $x_0 = min\{longitude_M\}$;
4  $y_0 = min\{latitude_M\}$;
5  $\mathcal{W} = max\{longitude_M\} - x_0$;
6  $\mathcal{H} = max\{latitude_M\} - y_0$ ;   // $\{longitude_M\}$ and $\{latitude_M\}$ are the sets of all longitudes and latitudes in city map, respectively
7  for each trajectory $\tau_i$ in mobility dataset do
8      Initialize an image $img_i$ with width of $\mathcal{W}$, height of $\mathcal{H}$ and white background ;   // The size of $img_i$ is ($\mathcal{W}$, $\mathcal{H}$, 3)
9      for each location $loc^i$ do
10         $x = \lceil (lon^i - x_0)/\mu \rceil - 1$;
11         $y = \mathcal{H} - \lceil (lat^i - y_0)/\mu \rceil$;
12         Set 8-neighborhood pixels of (x, y) as (0, 0, 0);
13         $img_i(x, y) = (0, 0, 0)$ ; // set the pixel located at $(x, y)$ as black
14     end
15 end
```

To train the DCGAN model, we transform the trajectories into images. Those images are fed into DCGAN model so as to capture the spatial characteristics of raw dataset. Particularly, there are 9,956 trajectories that are translated into 9,956 images and used for training. All of them are from one day in our dataset. Involving more training data might have few accuracy gain while result in longer training time. Meanwhile, we find that one day of mobility data cover almost all road segments and enough travel patterns of the city. In our DCGAN, p_z is set as a uniform distribution. Both the discriminator and generator are fully connected neural networks. The dimension of each fully connected hidden layer is set to 1,024. All the parameters are randomly initialized based on a zero-centered normal distribution with standard deviation 0.02. Batch size is set as 64. We use Adam optimizer to update the parameters in DCGAN with a learning rate of 0.0001.

For the images generated by DCGAN, we first leverage Harris corner detector (line 5 in algorithm 2) to extract the positions of corners that are potentially the locations. If the gray scale value of the extracted position is large enough (line 7 in algorithm 2), we convert it back to a pair of latitude and longitude based on the mapping relationship between image space and latitude-longitude

[1] *www.github.com/generating_mobility_trajectories_with_retained_data_utility*
This repository contains the source code, OSM map, and data samples of TrajGen.

ALGORITHM 2: Image-location translation.

```
1  Input: images $\{img_1, img_2, ...\}$, side length of minimal unit
     square $\mu$, threshold $\epsilon$, coordination of the origin point
     $(x_0, y_0)$;
2  Output: location pairs formatted by (latitude, longitude);
3  for each image $img_i$ generated by DCGAN do
4      Create a new file $f_i$;
5      pointLists = Harris($img_i$);
6      for each point $(x_i, y_i)$ in pointLists do
7          if $img(x_i, y_i) > \epsilon$ then
8              $lat_i = (\mathcal{H} - y_i) \times \mu + y_0$;
9              $lon_i = x_i \times \mu + x_0$;
10             Write $(lat_i, lon_i)$ into file $f_i$;
11         end
12     end
13 end
```

space. In algorithm 2, line 8 and line 9 are the reverse procedures of translation from locations to images. The parameter ϵ is determined empirically based on the maximal length of trajectories, original data, and generated images. In our setting, we set $\epsilon = 0.43$.

We implement the map matching algorithm to project trajectories into road segment space. For a given observed location, we leverage quad-tree based indexing to accelerate the query of its nearby road segment candidates. The underlying digital map is extracted from OpenStreetMap. According to the experimental results, we set the error range of the distance from a location to potential road segments as $200m$ and the noise standard deviation as $50m$, which is the robustest setting for different situations. Given one trajectory, map matching is able to find the best matched road segment for each location in the trajectory. The road segment embedding can be known when sorting all the segments based on their unique identifiers. The unique identifier is the ID attribute of each way in the OpenStreetMap file.

A Seq2Seq model is trained to learn the sequence information. The inputs are road segment embedding sequences and outputs are ground truth sequence generated directly from the original mobility dataset. All the parameters of Seq2Seq model are randomly initialized and updated using Adam optimizer. We leverage GRU as RNN units in Seq2Seq model because of its higher computational efficiency than LSTM. The dimension of hidden state is 1000 and the learning rate is set as 0.0001. We regularized the GRUs with a dropout rate of 0.1. The attention mechanism is leveraged to make the Seq2Seq model emphasize on important road segments. During training of Seq2Seq model, we adopt the padding technique to ensure all the inputs are at the same length.

Time inference model is trained to infer the timestamp of initial location in each trajectory. The fully connected ANN model has 3-hidden layers (first two layers have 1,000 neurons and the last layer has 5,760 neurons). Each units in ANN use a standard hyperbolic tangent (*i.e.,* tanh) activation function.