Synthetic Trajectory Generation Through Convolutional Neural Networks

Jesse Merhi*‡, Erik Buchholz*†‡, Salil S. Kanhere*
* *University of New South Wales*, Sydney, Australia
† *CSIRO's Data61, Cyber Security CRC*
‡ *Both authors contributed equally to this research.*

***Abstract*—Location trajectories provide valuable insights for applications from urban planning to pandemic control. However, mobility data can also reveal sensitive information about individuals, such as political opinions, religious beliefs, or sexual orientations. Existing privacy-preserving approaches for publishing this data face a significant utility-privacy trade-off. Releasing synthetic trajectory data generated through deep learning offers a promising solution. Due to the trajectories' sequential nature, most existing models are based on recurrent neural networks (RNNs). However, research in generative adversarial networks (GANs) largely employs convolutional neural networks (CNNs) for image generation. This discrepancy raises the question of whether advances in computer vision can be applied to trajectory generation. In this work, we introduce a Reversible Trajectory-to-CNN Transformation (RTCT) that adapts trajectories into a format suitable for CNN-based models. We integrated this transformation with the well-known DCGAN in a proof-of-concept (PoC) and evaluated its performance against an RNN-based trajectory GAN using four metrics across two datasets. The PoC was superior in capturing spatial distributions compared to the RNN model but had difficulty replicating sequential and temporal properties. Although the PoC's utility is not sufficient for practical applications, the results demonstrate the transformation's potential to facilitate the use of CNNs for trajectory generation, opening up avenues for future research. To support continued research, all source code has been made available under an open-source license.**

***Index Terms*—Trajectory Privacy, Differential Privacy, Location Privacy, Deep Learning, Generative Adversarial Networks**

I. Introduction

Due to the omnipresence of sensor-equipped devices like smartphones, large quantities of location data are collected daily. These trajectory datasets offer the potential for various use cases, such as public transport optimisation and pandemic control. However, location data yields severe privacy implications as it allows the re-identification of individuals [1], [2] and the inference of personal attributes [3], [4]. To highlight the high information content of location traces, De Montjoye et al. [1] managed to identify 95 % of users in a mobile phone dataset based on four locations only. Moreover, researchers determined Islamic taxi drivers in an anonymised dataset by correlating the mandatory prayer times with their breaks [5].

Researchers have proposed various methods for the privacy-preserving release of trajectory datasets [3], [6], [7], [8]. These methods focus on syntactic privacy, such as k-anonymity [9], and semantic privacy, notably Differential Privacy (DP) [10]. DP has become the de facto standard due to its strong formal guarantees. However, these methods involve a privacy-utility trade-off, requiring data perturbation to ensure privacy [11]. Current approaches often fail to maintain sufficient utility in the protected data for effective analysis [12], [13], [14]. Inappropriate protection can also lead to structural anomalies, such as cars not adhering to roads or ships traversing land [15], [16], [17], [8]. These anomalies may facilitate reconstruction attacks, thus reducing the achieved privacy level [18], [16].

For these reasons, Liu et al. [19] proposed using Generative Adversarial Networks (GANs) to generate synthetic trajectory datasets. The Deep Learning (DL) model retains the original dataset's key characteristics but produces inherently private, synthetic data. Models such as LSTM-TrajGAN [12] show the potential utility of this method. Yet, these solutions lack strong privacy guarantees. There are risks of the model memorising and replicating real trajectories with minimal alterations. To guarantee that the model does not remember training data, the usage of DP-SGD [20] has been proposed [21]. However, LSTM-TrajGAN uses a real trajectory as input, unlike standard GAN models like Deep Convolutional Generative Adversarial Network (DCGAN) [22] that use Gaussian noise. DP-SGD only secures training data, thus it is ineffective for architectures that rely on real data during generation [21].

To incorporate DP-SGD in trajectory GANs, a *noise-only* model is required. As trajectories represent sequences of locations, Recurrent Neural Network (RNN)-based models are the obvious choice. However, most GAN research has centred on computer vision, focusing on Convolutional Neural Network (CNN)-based models. RNN-based GANs often train less stably than CNN models and struggle with convergence [21]. This raises the question of whether a CNN-based architecture could be utilised for trajectory generation.

In other sequential domains, the move from RNN-based models to CNN-based models showed success. WaveGAN [23] adapts DCGAN for audio data generation, employing Conv1D layers and outperforming an RNN-based model. Similarly, PAC-GAN [24] utilises a CNN-GAN for the generation of network traffic packets where most related work had used RNNs. These examples highlight the potential of CNN-based architecture's in sequential domains.

This work introduces a *Reversible Trajectory-to-CNN Transformations (RTCT)* to facilitate the use of CNN-based

architectures for trajectory datasets. We demonstrate its utility by integrating it with DCGAN [22] in a Proof-of-Concept (PoC) implementation. This study primarily focuses on the feasibility of RTCT with a CNN-based model; we do not optimise DCGAN for trajectory generation but leave this task for future work. We evaluate on two real-world datasets, Foursquare NYC [25] (FS-NYC) [25] and Geolife [26] with four metrics: 1) The widely used Hausdorff Distance (HD), and 2) the sliced Wasserstein Distance (WD) assess the spatial distribution, 3) Total Travelled Distance (TTD) the sequential properties, and 4) a novel metric, Time Reversal Ratio (TRR), for assesses time properties. Moreover, we qualitatively analyse the generated trajectories by plotting them.

Our PoC is benchmarked against Noise-TrajGAN (NTG) [21], a *noise-only input* variant of the State Of The Art (SOTA) model LSTM-TrajGAN [12]. NTG was selected despite its inferior performance due to LSTM-TrajGAN's incompatibility with DP-SGD. After establishing a baseline comparison between our PoC and NTG, we proceeded to train both models with DP-SGD.

The evaluation demonstrates that our PoC more effectively captures the distribution of points within a trajectory dataset compared to the RNN model. However, in terms of spatial and temporal properties measured through TTD and TRR, respectively, NTG outperforms our PoC. Despite these results, the overall quality of the trajectories generated by both models does not yet meet the requirements for downstream applications. The DP version of our PoC achieves results comparable to NTG. However, qualitative analysis suggests that the noise added by DP-SGD significantly impacts the model. The DP version of NTG yields surprisingly good quantitative results. Yet, visual inspection does not confirm effective learning. While the results on a toy dataset were promising and showed the feasibility of integration with DP-SGD, significant future work is required to establish a useful DP model.

The evaluation results show that the PoC cannot reach the utility provided by (non-private) SOTA models. Yet, they highlight that the proposed transformation enables the use of CNN-based models in trajectory generation. Future research could explore tailored GAN designs to enhance trajectory generation. The ultimate goal is the development of a stable CNN-based GAN that integrates with DP-SGD for formal privacy guarantees. Concretely, this paper advances the field of trajectory privacy in the following ways (See Section V):

1) We introduce a Reversible Trajectory-to-CNN Transformations (RTCT) enabling the usage of CNN models for trajectory generation. (see Section VI)
2) We integrate this transformation with the well-known DCGAN [22] architecture in a PoC (see Section VII).
3) We evaluate our PoC on two real-world datasets, FS-NYC [25] and Geolife [26], and compare it to the RNN-based NTG model. (see Section IX).
4) We integrate both models with DP-SGD and evaluate the privatised versions (see Section VIII).
5) We publicly share our source code to facilitate reproducibility and future research[1].

This paper is organised as follows: Section II provides background on trajectory datasets, DP, and GANs. Evaluation metrics are detailed in Section III. Section IV reviews related work. Our objectives and contributions are outlined in Section V. The design of the RTCT is described in Section VI. Section VIII discusses the integration of DP-SGD. Our approach is evaluated in Section IX, while Section X addresses the results and potential future research directions. The paper concludes with Section XI.

II. Background

This section lays out the background knowledge for the remainder of this paper. Section II-A defines a trajectory dataset. Section II-B provides an overview of DP, emphasising its application in deep learning via DP-SGD. In Section II-C, we explore Generative Adversarial Networks. Section II-D provides further details on the DCGAN architecture.

A. Trajectory Datasets

A trajectory dataset consists of a number of trajectories: $D_T = \{T_1, \ldots, T_n\}$. Each trajectory T_i itself represents an ordered sequence of locations $T_i = (l_{i1}, \ldots, l_{in})$, where each location consists of multiple attributes. We assume the minimal information content of a location to be spatial coordinates such as latitude and longitude, i.e., $l_{ij} = (lat_{ij}, lon_{ij})$. However, locations might be enriched with additional information. For instance, some datasets record altitude [26], while others contain temporal information [26], [25], or semantic information such as Point of Interests (POIs), i.e., the type of location. Within this work, we assume spatial details to be minimally present. Temporal and semantic information are optional and can extend all proposed approaches.

B. Differential Privacy

Differential Privacy (DP) [10] is a rigorous semantic privacy notion. In contrast to syntactic privacy notions like k-Anonymity, DP offers protection against any adversary independent of their background knowledge. DP is founded on the principle of *plausible deniability*, i.e., participation in a query should not (significantly) affect the outcome. Accordingly, participation in a dataset does not harm one's privacy. Mathematically [10]:

Definition 1 (Differential Privacy): A mechanism $\mathcal{K}$ provides (ε, δ)-differential privacy if for all *neighbouring* datasets D_1 and D_2, and all $S \subseteq Range(\mathcal{K})$ holds

$$\mathbb{P}[\mathcal{K}(D_1) \in S] \leq e^{\varepsilon} \times \mathbb{P}[\mathcal{K}(D_2) \in S] + \delta \tag{1}$$

In the original definition, *neighbouring* means that dataset D_2 can be obtained from D_1 by removing all user records u from the dataset: $D_2 = D_1 \backslash \{r_{xi} | x = u\}$. In practice, different neighbourhood definitions, also called *unit of privacy* [21], are deployed. Most commonly, especially in machine learning, *instance-level DP* is used, which assumes that each user

[1] https://github.com/jesse-merhi/CNN-TRAJGAN

contributes exactly one record. The primary privacy parameter, ε, typically ranges from 0.01 to 10 in general contexts [27], but may be higher in DL [20]. The parameter δ, representing the accepted failure probability, is often set at $\delta = \frac{1}{n}$ for n records to ensure each record's protection, although some scenarios require lower values [28]. To meet these privacy requirements, DP integrates noise addition through mechanisms such as Laplace, Gaussian, or exponential [10], [29].

Differential Private Machine Learning. Deep learning models often rely on sensitive data for training, raising privacy concerns. Consider a DL model that targets the detection of diseases based on real medical data. It must be impossible to derive training data from the model's outputs or from the model's parameters in case the trained model is shared. However, different attacks [30] demonstrate the privacy risk of deep learning models. To counteract this, differential privacy has been integrated into DL model training [31], [20].

Differentially Private Stochastic Gradient Descent (DP-SGD) [31] is the prevalent method for achieving DP in deep learning [20]. To prevent privacy leakage, the influence of training samples on the model's parameters is bounded. The first step in DP-SGD involves clipping the gradients to a norm C, which limits the impact of each training sample. Next, Gaussian noise is added to these clipped gradients, determined by the norm C and a noise multiplier σ. Finally, privacy accounting monitors the privacy budget ε during training to ensure the model meets (ε, δ)-DP upon completion.

However, DP-SGD only safeguards training data's influence on model parameters. Data used during prediction or generation in generative models remains unprotected. This limitation underpins our approach to designing a generative model that relies solely on noise during generation, as suggested in [21]. While privacy libraries simplify DP-SGD usage, guaranteeing privacy for models like LSTM-TrajGAN, which require real data during generation, remains a challenge (ref. Section I).

C. Generative Adversarial Networks

Generative Adversarial Networks (GANs) [32] are a DL algorithm used in unsupervised machine learning. These networks consist of two main components: a generator (G) and a discriminator (D). G generates data from random noise, aiming to mimic real data samples. D assesses whether a sample is real or produced by G. This interaction forms a competitive framework, described by the adversarial loss:

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] \quad (2)$$

In this equation, p_{data} is the real data distribution, and p_z is the distribution of the noise input for G. The training process refines G and D until G can produce data almost identical to real samples. One advantage of traditional GANs architecture in regard to privacy is that the generator receives only random noise as an input during generation. Real data samples in GANs are necessary only during the training phase. This characteristic facilitates seamless integration with DP-SGD. In contrast, accessing the real data during generation can lead to privacy leakage [21]. Building on this understanding of GANs, we explore the well-known DCGAN architecture next.

D. DCGAN

Most current GAN models are loosely based on DCGAN [22], owing to its effectiveness in generating valid synthetic data across various applications [33]. DCGAN is a GAN utilising convolutional layers in both its generator and discriminator that was developed for image generation. Its five key architectural features include:

- Strided and transposed convolutions for downsampling and upsampling, replacing pooling layers.
- Batch normalisation in both generator and discriminator.
- ReLU activation in the generator, except for tanh in the output layer. Note that we used sigmoid for the output layer due to our normalisation to $[0; 1]$.
- LeakyReLU activation in the discriminator, except for sigmoid in the output layer.
- Elimination of Fully Connected (FC) hidden layers.

These modifications lead to DCGAN's training stability and broad applicability.

III. Metrics

This section outlines the metrics for our utility evaluation. We first discuss two metrics for evaluating spatial distribution: HD (Section III-A) and WD (Section III-B). We then describe TTD (Section III-C) for sequential quality and introduce the novel TRR (Section III-D) to assess temporal properties.

A. Hausdorff Distance

The *Hausdorff Distance (HD)* is a distance metric that quantifies the similarity between two sets of points. It calculates the maximum distance from a point in one set to the nearest point in the other set. Formally, for two point sets A, B and a distance function d [34]:

$$HD(A, B) = \max \left\{ \max_{a \in A} \min_{b \in B} d(a, b), \max_{b \in B} \min_{a \in A} d(a, b) \right\} \quad (3)$$

This metric is widely used for evaluating spatial utility in trajectory protection and generation [35], [36], [37], [12], [14]. We generate trajectories until their combined point count matches the real test set's point count. Then, we compute the HD between both sets. However, the HD is significantly affected by outliers [21], such that we also use the WD.

B. Wasserstein Distance

The HD is prevalent in trajectory comparison but is highly susceptible to outliers, where a single misplaced point can lead to a drastically different HD. To address this, we compare the point distributions directly as proposed in [21]. A key aspect is assessing if the generated data mimics the real data's distribution, such as urban concentration versus rural sparsity. Intuitively, the Wasserstein Distance (WD) treats distributions as piles of dirt, measuring how much 'dirt' must be shifted to transform one distribution into another. Hence, this metric is also known as Earth Mover's Distance (EMD). However,

standard WD is limited to one-dimensional data, while spatial trajectories involve (at least) two dimensions (latitude and longitude). Therefore, we opt for the Sliced Wasserstein Distance (SWD), suitable for two-dimensional data. Given SWD's computational overhead, we randomly draw $10\,000$ locations from both the real and the generated dataset and compute the SWD between those sets. Since these two metrics only evaluate the point distribution, we use the subsequently discussed TTD for assessing the sequential characteristics.

C. Total Travelled Distance

The *Total Travelled Distance (TTD)* represents the length of the entire trajectory. For a trajectory $t = ((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n))$, the TTD is defined as:

$$\text{TTD}(t) = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{4}$$

The TTD is calculated by computing the total travelled distance for each trajectory in both the real and generated datasets, creating two sets of travel distances. We then use the WD to compare these sets by measuring the discrepancy between their underlying distributions, quantifying how similar the generated data is to the real data regarding travel distances.

D. Time Reversal Ratio

We identified a lack of metrics assessing the quality of the temporal features in generated trajectories. To address this gap, we developed a new metric, *Time Reversal Ratio (TRR)*, which assesses how accurately a trajectory represents the flow of time. The goal of TRR is to ensure that generated trajectories display minimal instances where time appears to move backwards, mirroring the consistent forward progression of time in real-life trajectories. Formally, for a trajectory $t = ((x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n))$ with timestamps t_i, TRR is defined as:

$$\text{TRR} = \frac{\sum_{i=1}^{n-1} I(t_i > t_{i+1})}{n-1} \tag{5}$$

Here, I is the indicator function, equalling 1 if $t_i > t_{i+1}$ (indicating a regression) and 0 otherwise. While minor errors in location points are tolerable for trajectory utility, generated trajectories exhibiting backward time movement are readily identifiable as synthetic. Therefore, optimising against such temporal inconsistencies is essential to enhance realism.

IV. Related Work

Numerous privacy-preserving mechanisms for trajectory data exist. Jin et al. [7] survey various approaches, including those based on k-Anonymity, while Miranda-Pascual et al. [8] focus on DP mechanisms. Despite significant progress, a privacy-utility trade-off persists [12], [13], [14]. k-Anonymity offers practical privacy but lacks robust formal guarantees. Conversely, DP mechanisms cause significant utility loss. Generating synthetic trajectories via deep learning emerged as a potential alternative [19]. Notable implementations include the aforementioned LSTM-TrajGAN [12]. Numerous other DL-based generative models have been proposed [38], [39], [40], [41], [42], [43], [44], [45] and discussed in a recent SoK [21]. Yet, none of the existing deep learning-based generative models can provide strong formal privacy guarantees, such as DP [21]. More recent approaches not considered in this SoK, such as DiffTraj [46], improve utility further but still do not offer formal privacy guarantees and rely on the inherent privacy of generation instead. However, generated data can leak information such formal guarantees remain desirable [21], [47], [48]. Moreover, Buchholz et al. [21] mention that Conv1D layers appear to be superior for capturing the spatial distribution of trajectories compared to RNN-based models. This motivates our investigation of CNN-based generative models.

LSTM-TrajGAN [12], [49] represents the most cited deep learning-based generative trajectory model. The model consists of a generator and discriminator with similar architectures. Real trajectories are normalised and encoded, combining location, temporal, and semantic features. The generator embeds and fuses these encodings with noise into latent representations. This representation is processed by an Long Short-Term Memory (LSTM) layer, generating synthetic trajectories with multiple features. Unlike the classic GAN architecture (Section II-C), where the generator only receives Gaussian noise as input, LSTM-TrajGAN uses real trajectories as input during generation. As pointed out in [21], this has privacy implications and might lead to the generated trajectories leaking information about the input trajectories. Moreover, this architecture makes it challenging to integrate DP guarantees. Therefore, we use a variant of LSTM-TrajGAN, named **Noise-TrajGAN (NTG)** [21], as our evaluation baseline. Unlike the original, NTG omits trajectory inputs during generation. Its architecture mirrors LSTM-TrajGAN's, with the primary difference being that its generator uses a noise vector alone. We made the choice to use NTG for two reasons. First, NTG can be trained with DP guarantees by using DP-SGD (ref. Section II-B, which allows us two compare a DP version of our PoC to DP-NTG. Second, the architecture is more similar to DCGAN, which also uses noise-only input, such that the results focus more on the actual research question of comparing RNN with CNN-based models.

Sequence Generation Based on CNN. The preference for RNNs in trajectory generation stems from their suitability for sequential data. Yet, CNN-based models have demonstrated effectiveness in sequential tasks. For example, WaveGAN [23], uses Conv1D for audio generation, and PAC-GAN [24], employs a 2D CNN for network traffic generation.

WaveGAN, adapts DCGAN (ref. Section II-D) to 1D data by using Conv1D layers for the generation of synthetic audio waves [23]. Despite the sequential nature of audio waves, this model produces high-quality audio suitable for multimedia applications. WaveGAN surpasses earlier models like SampleRNN [50] and WaveNET [51], highlighting the effectiveness of Conv1D layers. The model exploits the periodicity of

sound waves, which the sliding convolutional kernel captures. While trajectories contain regular patterns, they do not exhibit the same periodicity, limiting the model's applicability [21].

PAC-GAN [24] employs a 2D CNN-based GAN to generate synthetic network traffic packets. This approach proposes a transformation algorithm that converts hex-encoded packets into 2D matrices compatible with CNNs. This representation yields an up to $99\,\%$ success rate for individual traffic type generations while an initial encoding only reached up to $30\,\%$. This outcome highlights the critical role of data representation in model performance. These examples motivate the design of a transformation enabling the usage of CNN-based models for trajectory generation, opening avenues for future research.

V. Problem Statement

As noted in Section I, location trajectories are valuable for analyses but contain sensitive information. Related work indicates a limiting privacy-utility trade-off in traditional protection methods. Therefore, the generation of synthetic data represents a promising alternative. However, while achieving high utility, current models struggle to provide rigid privacy guarantees [21]. Moreover, while most current methods rely on RNNs, CNN-based architectures have shown success in other sequential domains, and initial experiments with convolutional layers for trajectory generation yield promising results [21]. This leads to our research question: *Can a CNN-based GAN produce high-quality synthetic trajectories?* To the best of our knowledge, this is the first attempt to utilise a fully CNN-based architecture for trajectory generation. Our primary research question is subdivided into four sub-goals:

G1: Transformation: Develop a Reversible Trajectory-to-CNN Transformations (RTCT) algorithm that transforms location trajectories into CNN-compatible inputs. This transformation represents the main contribution of this research.

G2: Integration: Integrate our transformation with a standard CNN-based GAN, specifically DCGAN (ref. Section II-D), to assess its performance in a PoC. We have chosen an established GAN model to highlight the flexibility and potential of our proposed transformation. The development of a specialised CNN-based GAN tailored for trajectory synthesis remains an objective for future work.

G3: Differential Privacy: Apply DP-SGD to both our model and the considered baseline model NTG to determine the impact of formal privacy guarantees on model performance. To the best of our knowledge, this is the first work using DP-SGD for privacy-preserving trajectory generation, providing formal privacy guarantees.

G4: Evaluation: Evaluate the potential of CNN-based GANs for trajectory generation by conducting an extensive evaluation on two real-world datasets, FS-NYC and Geolife, with the four metrics outlined in Section III. Additionally, we compare our model's performance with the RNN-based model NTG, described in Section IV. Moreover, we compare the performance of both models trained with DP-SGD to measure the impact of formal privacy guarantees on utility.

VI. Reversible Transformation Design

To enable the usage of CNN-based GANs for trajectory generation, we propose a *Reversible Trajectory-to-CNN Transformations (RTCT)* addressing **G1: Transformation**. This transformation, depicted in Figure 1, comprises:

1) Normalising the trajectories' latitude, longitude, day and hour values (Section VI-A).
2) Inserting these normalised values into a $12\times12\times3$ matrix (Section VI-B).
3) Upsampling the resulting matrix to $24 \times 24 \times 3$ (Section VI-B).

After generation, the synthetic trajectories are reverted into sequential form, as detailed in Figure 2 and Section VI-C. Additionally, we demonstrate the potential of using CNN-based models for trajectory generation by integrating our transformation with DCGAN [22] into a Proof-of-Concept (PoC) implementation, addressing **G2: Integration** (Section VII).

A. Normalisation

Normalisation adjusts dataset features to a common scale, which is essential for deep learning. Typically, data is normalised to the range $[-1; 1]$ or $[0; 1]$ using techniques such as mean and standard deviation, and tanh normalisation [52]. We employ min-max normalisation, which is reversible and scales data features into the range $[0; 1]$, making it ideal for the denormalisation of generated trajectories. For a feature f, with minimum value min and maximum value max, the normalisation value v is defined as:

$$v = \frac{f - min}{max - min} \tag{6}$$

This normalisation method is applied to all features before using the trajectories for model training.

Choice of max and min. Selecting appropriate maximum and minimum values for each feature is crucial for effective normalisation. Initially, we used global extremes for latitude and longitude: $-180°$ to $180°$ for longitude and $-90°$ to $90°$ for latitude. This method proved too coarse, as slight changes in coordinates, which could represent significant distances, were not adequately captured. For example, a change from $(lat, lon) = (40, 70)$ to $(40.0001, 70)$ represents a distance of approx. 10 meters but is negligible in global normalisation. Moreover, the values have to be dataset-independent to prevent privacy leakage. Finally, we opted for *geographical constraints* tailored to the datasets, such as a ring road for a Beijing dataset or a bounding box aligned with a city's official boundaries. This approach does not access the dataset but uses public knowledge, thus preventing information leakage. For *time-based features*, we applied fixed maxima and minima, 0-6 for days and 0-23 for hours, aligning with the natural constraints of these temporal features. This ensures sufficient variability for the model to detect patterns effectively.

B. Trajectory Encoding

Our trajectory encoding underwent two main iterations. Initially, we used an asymmetrical matrix as a "strip" to

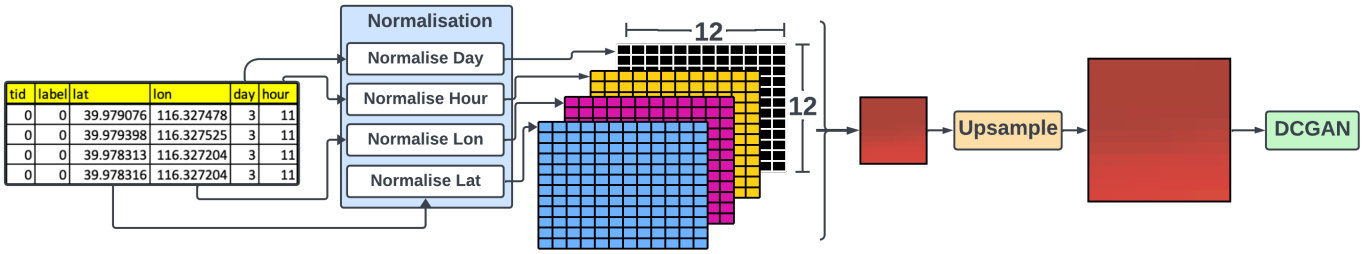

Fig. 1. **RTCT:** A trajectory's latitude, longitude, day, and hour values are normalised into a 12×12 matrix and upscaled to $24 \times 24 \times 4$ for DCGAN.

store trajectory information, but this format yielded unsatisfactory results. To overcome these limitations, we evolved our approach into a two-dimensional square convolution with multiple channels and clustered features. The final encoding scheme is depicted in Figure 1.

Strip Encoding Our initial encoding used a 144×4 strip-matrix, inspired by WaveGAN's audio encoding (see Section IV). The matrix's first dimension, 144, represents the maximum trajectory length, which depends on the dataset. The second dimension captures features, here: latitude, longitude, day, and hour. Early tests showed the model's learning was sub-optimal, possibly due to the lack of consistent periodicity in trajectories compared to audio waves. Thus, we adopted a two-dimensional representation for each feature and a replication strategy to reduce the effect of minor perturbations.

Multi-Dimensional Square Convolution Encoding Our final encoding scheme evolved into a $12 \times 12 \times 4$ matrix, shown in Figure 1. It accommodates 144 trajectory points, with each cell holding four separate feature channels. This format generalises to trajectories with maximum length x and y features, resulting in a matrix size of $\lceil \sqrt{x} \rceil \times \lceil \sqrt{x} \rceil \times y$. The main drawback of CNN-based models over RNN-based ones is the need to pre-define an upper trajectory length. To mitigate perturbation effects, we upscaled our matrix to $24 \times 24 \times 4$, replicating features into 3 additional cells [53].

C. Reversion

The generator outputs synthetic trajectories in the introduced encoded format. Therefore, the reversibility of this encoding into trajectories is essential. Figure 2 illustrates the reversion process. Starting with a synthetic trajectory encoding of shape $24 \times 24 \times 4$, we downsample it to $12 \times 12 \times 4$. The downsampling divides the matrix into 2×2 squares, selecting one cell per square using the nearest-neighbour method. After downsampling, values v are denormalised into the feature f using the rearranged normalisation Equation 6:

$$f = v \times (max - min) + min \tag{7}$$

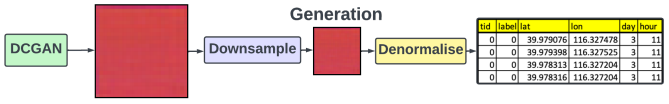

Fig. 2. **Reversion:** DCGAN generates a $24 \times 24 \times 4$ trajectory, which is downsampled to $12 \times 12 \times 4$ and denormalised to retrieve the original format.

This reversion is applied to each cell. The features are then concatenated to revert to the original trajectory format.

VII. DCGAN Integration

As highlighted in Section II-D, DCGAN is a popular GAN choice due to its ease of use and versatility. Therefore, we integrated DCGAN with RTCT as a Proof-of-Concept[2]. Although DCGAN was primarily designed for image generation and may not be ideal for trajectories, its adaptability makes it suitable for our PoC addressing **G2: Integration**, as this work focuses on transformation rather than the generative model itself. In the following, we provide details on the implementation and optimisation strategies.

A. Implementation

We based our PoC on the DCGAN implementation from the PyTorch GAN repository [54]. PyTorch data loaders facilitate dynamic access to data, crucial for training. Our transformation is embedded in the data loader, managing loading raw trajectories from files to the encoding with RTCT.

Padding and Masking. As described in Section VI-B, CNNs require constant length inputs, unlike RNNs. Therefore, we pad trajectories shorter than the upper limit of 144 points using 0-post padding. Masks based on the trajectories' original lengths are generated, with a vector for a trajectory of length l comprising l ones and $144 - l$ zeros. When applied to trajectories, this masking excludes zero-padding from affecting computations within the computational graph.

B. Optimising DCGAN

To enhance the baseline PyTorch DCGAN performance, we employed several optimisation strategies:

1) Implementing Two Time Update Rule (TTUR) [55].
2) Using a learning rate scheduler [56].
3) Applying label smoothing [57].

The **Two Time Update Rule (TTUR)** [55] refers to using a lower learning rate for the generator than for the discriminator. This optimisation facilitates the discriminator converging to a local minimum while the generator progresses more slowly [55]. We integrated TTUR through a *generator factor* that reduces the generator's learning rate to one-tenth of the discriminator's learning rate.

[2]The implementation is available at https://github.com/jesse-merhi/CNN-TRAJGAN

Learning rate schedulers are widely used for optimising performance in deep learning [56]. While adaptive rates in optimisers like the Adam Optimiser [58] are common, we observed improved results by manually reducing the learning rates of both the discriminator and generator at specific milestones. A learning rate scheduler speeds up initial learning and helps find a local minimum later, avoiding overstepping.

Label smoothing, our final optimisation, modifies trajectory labels to reduce overfitting [57]. We adjusted valid labels to 0.9 and fake labels to 0.1. This reduces the discriminator's confidence and results in better gradients during training, facilitating a smoother and more stable learning process [57].

VIII. DP-SGD

As discussed in Section IV, the core limitation of existing deep learning models for trajectory generation are the missing formal privacy guarantees. Today's de-facto standard is DP described in Section II-B. Despite efforts to deploy DP for trajectory generation, studies [8], [21] show frequent misapplication, affecting the integrity of the privacy guarantees. Thus, [21] recommends using the established method of DP-SGD, which ensures instance-level DP if each training sample corresponds to one trajectory (see Section II-B).

To address **G3: Differential Privacy**, we implemented DP-SGD for our PoC and the baseline NTG (ref. Section IX-B) to evaluate the impact of DP and allow for comparative analysis. We employed the Opacus [59] library to integrate DP-SGD due to its straightforward interface. This established framework helps us avoid the common pitfalls of custom DP implementations, which have compromised the integrity of privacy guarantees in other studies [21], [8]. Any layers incompatible with DP-SGD were automatically replaced by Opacus' model fixer, in particular, DCGAN's BatchNorm layers were replaced by GroupNorm. Typically, DP-SGD is applied to the discriminator in a GAN [48]. This approach ensures that the generator also adheres to DP through the post-processing property of DP since it only receives indirect data access via feedback from the discriminator. However, we aimed at enabling usage of the WGAN-LP loss, which is reported to yield better results than the standard adversarial loss [60], [61], [21]. Opacus does not support DP-SGD for models trained with gradient penalty at this time. Moreover, the discriminator is updated more frequently than the generator when using WGAN [62] (usually $\approx 5\times$ as often), such that adding noise to the discriminator would result in more noise being added by DP-SGD. Therefore, we decided to train the generator with DP-SGD instead, i.e., the noise is added to the generator's gradients instead of the discriminator's gradients. Utilising the MNIST Sequential (MNIST-Seq) as a toy dataset [21], we confirmed that the baseline model NTG could produce samples of comparable quality with DP-SGD applied to the generator as it did without DP-SGD. Due to a bug in the underlying framework, we had to use the standard WGAN loss instead of WGAN-LP for the DP version of the baseline model NTG. The DP version of our PoC was successfully trained with WGAN-LP.

To minimise the performance degradation caused by the noise from DP-SGD, we adhered to the guidelines from Google's DP-fy ML paper [20]. We set $\varepsilon = 10.0$, considered the upper limit for realistic privacy in deep learning [20]. For a dataset with n samples, we adopted $\delta = 1/n^{1.1}$ as commonly recommended [28], [20]. Compared to the non-DP model, we increase both the batch size b and the number of epochs e by a factor $F = 10$ to keep the number of steps ($s = e \cdot \frac{n}{b}$) constant but increase the batch size which reduces the noise that is added [20]. We selected a gradient clipping norm of $C = 0.1$. Due to time constraints and the high computational cost of DP-SGD, we could not complete a full ClipSearch [20] and learning rate sweep, but we verified that these heuristics yield good results on the MNIST-Seq dataset.

IX. Evaluation

This chapter addresses goal **G4: Evaluation**. Section IX-A details the datasets used and their preprocessing methods. Section IX-B introduces the baseline model, NTG. Subsequent sections discuss evaluation results: Section IX-C covers parameter choices, Section IX-D outlines the hardware setup, Section IX-E presents results from standard models, Section IX-F focuses on outcomes from training with DP-SGD, and Section IX-G provides a qualitative analysis.

A. Datasets

To demonstrate the generalisability of our approach, we evaluate our implementation on two different datasets. First, the FS-NYC dataset [25], used as a benchmark in LSTM-TrajGAN [12], comprises $3,079$ trajectories with a maximum of 144 locations each, primarily within New York City's bounds[3]. We use the dataset as provided in [49] without further preprocessing. Second, the Geolife dataset [26] covers a larger geographical area around Beijing. Moreover, $91\,\%$ of the dataset's trajectory have a sampling rate of 1–$5\,\mathrm{s}$ or 5–$10\,\mathrm{m}$ [26] yielding more fine granular trajectories. For consistency with FS-NYC, we constrained Geolife's data within the fourth ring road of Beijing[4]. We capped trajectories at 144 locations and discarded those with fewer than 96 locations to prevent extensive padding. This preprocessing reduced the available trajectories from $17\,621$ to 7270.

B. Baseline Model

LSTM-TrajGAN, as highlighted in Section IV, is a leading generative model for trajectories. Its use of real trajectory inputs during generation, however, raises privacy concerns and complicates the provision of DP guarantees [21]. To address these issues, we selected the noise-only variant NTG, introduced in [21], for its compatibility with DP-SGD and its architectural similarity to our PoC, ensuring a fair comparison. Despite NTG's inferior performance compared to LSTM-TrajGAN, it has the advantage that it integrates well with DP-SGD, an essential feature for our comparative analysis.

[3]Bounding Box = $(40.6811, -74.0785)$ to $(40.8411, -73.8585)$
[4]Bounding Box = $(39.8279,\ 116.2676)$ to $(39.9877,\ 116.4857)$

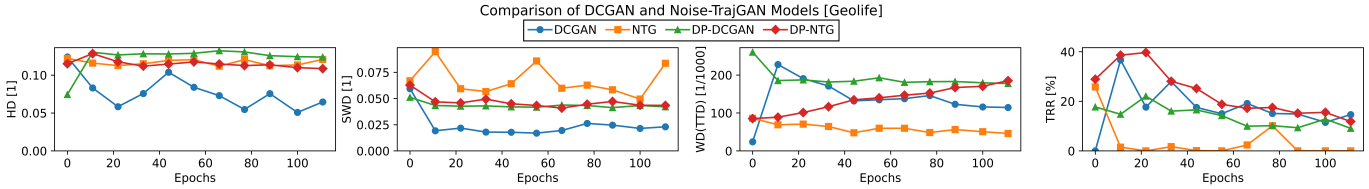

Fig. 3. On Geolife [26], DCGAN significantly outperforms NTG for the spatial metrics, i.e., the model can capture the point distribution better. However, the RNN-based NTG excels at capturing the distance between consecutive points (TTD). DCGAN struggles to generate sensible timestamps (TRR).

We extended the model by training with DP-SGD to facilitate comparisons of the DP versions with our PoC.

Hyperparameters were generally maintained as per [21]. We trained NTG using the WGAN-LP loss, which has been reported to yield the best outcomes [21]. For the DP version of the model, we had to revert to the standard WGAN loss due to an error caused by the combination of Opacus with the gradient penalty. Adjustments were made to the learning rate, set at $1e-4$, an increased batch size of 64 for alignment with our PoC, and extending the number of epochs to ensure a minimum of $10\,000$ training steps, matching the step count of our PoC. The parameters for the DP version, referred to hereafter as DP-NTG, follow these modifications except for the changes outlined in Section VIII.

C. Hyperparameters

This section outlines the training parameters used for the evaluation[5]. Both our PoC and the baseline model were trained for over $10\,000$ steps for consistency, with a batch size of 64. The learning rate for our PoC was set to $2e-4$. While we implemented the TTUR as described in Section VII-B, we empirically determined that a generator factor of 1.0, updating the generator and discriminator at the same frequency, yielded the best results. A learning rate scheduler, referenced in Section VII-B, reduces the learning rate by a factor of 0.1 after $4\,000$ steps, and label smoothing is applied to the discriminator. Contrary to NTG, which employs the WGAN-LP loss, DCGAN showed superior performance with the standard adversarial loss (ref. Section II-C). However, when training with DP-SGD, the WGAN-LP loss with 5 discriminator iterations per generator iteration proved superior for DCGAN. The generator in our PoC accepts noise input shaped $(batch\_size, 100)$. Network dimensions otherwise align with the DCGAN specifications from the PyTorch GAN repository [54]. All remaining hyperparameters follow the PyTorch defaults. Hyperparameters for the baseline model are outlined in Section IX-B. During DP-SGD training, most parameters were unchanged, with modifications to batch size, epochs, and learning rates as specified in Section VIII.

D. Evaluation Setup

All measurements were performed on a server (2x Intel Xeon Silver 4310, 128 GB RAM) with Ubuntu 22.04.4 LTS. The server contains 4 NVIDIA GeForce RTX 3080 (10 GB RAM each), but only one GPU was used per experiment. All measurements were executed with 5-fold cross-validation.

[5]The repository contains detailed configuration files for all measurements.

E. Results

This section presents a comparative utility evaluation between our PoC and NTG. The results of both models on the Geolife dataset are displayed in Figure 3, and those on FS-NYC in Figure 4. Examples of one generated trajectory and the generated point distributions are provided in Figures 5 and Figure 6, respectively. All results are reported at the end of the training, i.e., after approx. $100\,000$ steps.

1) Hausdorff Distance: The HD measures the maximum disparity between two sets of points (ref. Section III-A). A lower HD indicates closer similarity between the compared sets, thereby indicating better performance. On Geolife, our approach achieved a HD of 0.0646. NTG, with a HD of 0.120, exhibited 1.87 times worse performance. On FS-NYC, our PoC still performs better, but only by a factor of 1.45.

2) Sliced Wasserstein Distance: The SWD was employed as a second metric to evaluate the spatial distribution utility. It assesses the shape, size, and density of the points within a generated dataset, with lower scores indicating a higher resemblance to the real dataset. The results confirm the findings of the HD, that our PoC is superior at capturing the spatial distribution of the dataset. On the Geolife dataset, DCGAN performs 3.64 times better, and on FS-NYC, 1.34 times.

3) Total Travelled Distance: The TTD (ref. Section III-C) compares the distribution of travel distances in a generated dataset against the real-world dataset via the WD. Lower values indicate a greater resemblance to the actual dataset. On the Geolife dataset, NTG outperforms the PoC by a factor of 2.5. This outcome is expected, as the TTD measures the distance between consecutive locations and depends on the model's ability to capture the sequential dependency of these locations. Since RNNs excel at capturing sequential properties, it appears reasonable that they outperform the CNN-based model in this metric. However, on the FS-NYC dataset, NTG performs significantly worse than DCGAN in terms of TTD, which is surprising. This variation could be attributed to the Geolife data consisting of relatively uniformly sampled trajectories, in contrast to the FS-NYC dataset, which features user check-ins with greatly varied granularity and distances.

4) Time Reversal Ratio: The *Time Reversal Ratio (TRR)*, explained in Section III-D, counts the number of times two consecutive timestamps are impossible, i.e., an optimal score is 0. On both datasets, the PoC implementation performs

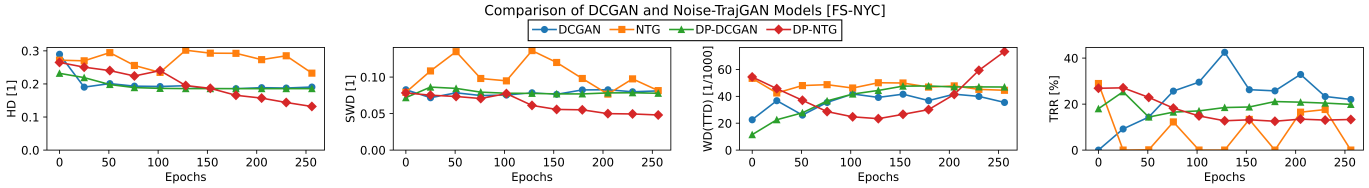

Fig. 4. On the FS-NYC [25] dataset, DCGAN captures the spatial distribution better than NTG, although the difference is less significant than on Geolife. Interestingly, NTG performs the worst in regard to the TTD on FS-NYC. The TRR shows similar results to Geolife.

worst with 15 % bad transitions on Geolife and 22 % on FS-NYC, while NTG produces barely any backward transitions. As outlined for TTD, the RNN appears superior at capturing the dependency of a timestamp to its predecessor.

F. DP Training

This section describes the evaluation of training with DP-SGD. The results for both models are displayed in Figures 3 and 4, and the generated point cloud of DP-DCGAN is shown in Figure 6. Initially, we conducted training on the toy dataset MNIST-Seq and achieved results comparable to those of the model trained without DP-SGD, using $\varepsilon = 10$ and the hyperparameter variations detailed in Section VIII. On Geolife, the DP version of our PoC performs similar to NTG in regard to the spatial metrics, but worse for the sequential metrics. The point cloud visualisation (ref. Figure 6) shows that the produced points cloud looks nearly normally distributed, indicating a significant negative impact of the noise added to the gradients by DP-SGD. Yet, the centre of the distribution aligns with the highest density of real points, showing learning of the model. On FS-NYC, the results for the DP version are similar to those of the standard PoC, which is a very promising outcome. Nevertheless, DP-SGD causes a significant utility degradation, and optimisation is required to make the DP model practical.

Remarkably, DP-NTG appears to outperform its non-DP counterpart in regard to the spatial metrics and even achieves the best values for both HD and SWD on the FS-NYC dataset. Meanwhile, the sequential properties degrade substantially compared to the standard NTG model. On closer investigation, we noticed that the points generated by DP-NTG are randomly scattered over the entire map and do not exhibit any structure, which naturally yields very low HD values. The outputs of DP-NTG appear to be mainly noise. We assume that the combination of the LSTM with DP-SGD is highly unstable, as we encountered several issues during the evaluation and were unable to use WGAN-LP. Moreover, the results show that DP-DCGAN and DCGAN perform similarly with DP-DCGAN being slightly worse, while the relationship between NTG and DP-NTG is less clear. While our results are insufficient to make definitive claims, the results indicate that the combination of DP-SGD with CNN-based models might be preferable. Our experiments show that applying DP-SGD to these models is feasible but results in utility degradation.

G. Qualitative Analysis

Figure 5 displays an example of a generated trajectory, while Figure 6 displays the distribution of generated locations over those in the real dataset. While the generated data do not yet match the utility required for more complex downstream applications, they show a foundational ability to approximate spatial distributions. This aligns with the quantitative results described in the previous section. The point clouds primarily capture the denser areas of the original dataset, demonstrating the model's capability to learn essential spatial patterns. However, finer details such as street layouts are not discernible, suggesting limitations in the current DCGAN architecture's capacity to model highly detailed urban geography. A deeper network could improve on this shortcoming.

X. Discussion

This study's exploration into using CNN-based GANs for trajectory generation has yielded promising insights despite not achieving state-of-the-art results. Our findings indicate that CNN-based models are superior at capturing spatial distributions, whereas RNN-based models better capture sequential properties, aligning with observations from related work [21]. This difference in performance is likely attributed to the architecture of each model. CNN-based models find patterns in inputs using a sliding window that processes several inputs at once. The RTCT transformation places values in a matrix, such that values within the same window are not necessarily sequentially adjacent. Therefore, sequentiality is not necessarily the model's focus. In contrast, RNN-based layers always process values in a sequential manner, such that sequential dependencies are emphasised. When comparing our prototype with state-of-the-art models such as LSTM-TrajGAN, it is important to note that these models exhibit higher utility because they modify real trajectories rather than generating new samples from Gaussian noise, as our PoC does. While such approaches can improve utility, they may compromise privacy by handling real data directly. Our model, by generating synthetic trajectories, does not incur these privacy issues. Additionally, the primary focus of our work was on the proposed RTCTtransformation, hence the use of DCGAN "as-is" without tailoring it for trajectory generation.

While the prototype's ability to approximate spatial distributions is promising, its failure to capture finer details like specific street layouts underscores the limitations of the current DCGAN model. This suggests a need for more sophisticated or

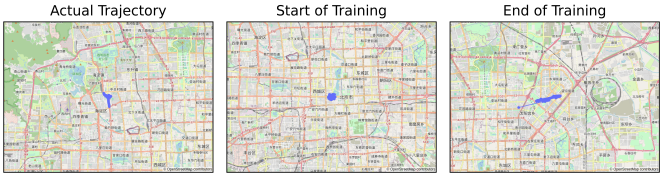

Fig. 5. The real trajectory is dense and follows a primary direction. Generated trajectories are initially clustered in the map's centre but become more meaningful after training.

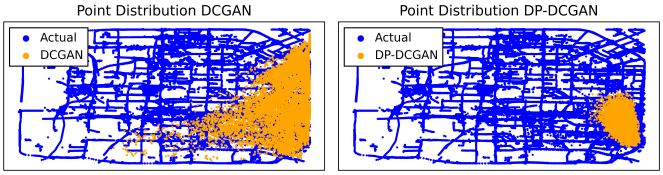

Fig. 6. The main density of points generated by DCGAN aligns with the dataset but lacks detail. The DP version seems to be nearly normal distributed.

larger network architectures to adequately capture the spatial patterns inherent in geographic data. Similar challenges were noted by GeoPointGAN [17], which found that a standard PointNet could not capture minor patterns such as small roads, but using deeper networks improved results. We plan to explore more complex CNN-based models in future work, as it is beyond the scope of this paper.

Furthermore, we have demonstrated that training our PoC with DP-SGD is feasible with acceptable overhead. While the addition of noise impacts the results, this effect could be manageable for sufficiently large datasets and a privacy budget of $\varepsilon = 10$. Considering the limited quality of the baseline results, more research is needed to fully assess the utility degradation caused by DP training. Nonetheless, DP-SGD represents a promising direction for trajectory generation with formal privacy guarantees. Overall, this project lays the groundwork for privacy-preserving, CNN-based trajectory generation, which is discussed in the following section.

A. Future Work

The discussion highlights several avenues for future research. The RTCT shows potential for adapting image-generation GANs from computer vision to trajectory generation. This could allow the use of advanced generative models.

An important area for future development is a domain-specific CNN-based GAN. Our current work has established a suitable encoding for CNN models, but the next step is to refine the model architecture itself. From our discussion, several specific paths emerge. First, exploring deeper, more complex models could enable capturing detailed aspects of datasets, such as the road network, beyond just the primary density clusters (ref. Figure 6). A starting point for this direction could be the large PointNet proposed by GeoPointGAN for the generation of geographical point clouds [17]. Second, the model could focus more on the sequential properties, for instance, through combining a CNN with an RNN or positional encodings. Third, future work could further examine the premise that RNNs excel in sequential property representation while CNNs are better suited for spatial distributions. Fourth, implementing a specialised loss function akin to LSTM-TrajGAN's trajLoss could benefit model performance.

Additionally, future research could aim for a more equitable comparison between RNN-based and CNN-based GANs. This could involve developing a CNN-based GAN with an architecture similar to LSTM-TrajGAN, i.e., a CNN-based model that receives a real trajectory encoding as input. Though a traditional GAN framework relying on noise generation is preferred for privacy, such an approach would facilitate a detailed assessment of the distinct strengths of CNNs and RNNs in trajectory generation. Furthermore, concrete attacks and downstream applications could extend the privacy and the utility analysis, respectively. For example, the Trajectory User Linking (TUL) [63] success rate could be determined as empirical privacy metric **LSTM-TRAJGAN**, although an assignment of trajectory IDs to generated samples is not straightforward due to the noise-only architecture. In terms of downstream tasks, next week trajectory prediction [64], range queries [17], or hotspot preservation [17] could be deployed. Finally, future work could explore the impact of the privacy budget ε on the provided utility and privacy by training multiple identical models and only varying ε.

XI. Conclusion

Location trajectories, while valuable for various applications, inherently contain sensitive information, posing significant privacy concerns. Traditional trajectory privacy-protection mechanisms face a restrictive privacy-utility trade-off. Therefore, deep learning-based generative models have been proposed as a promising alternative. Yet, current models do not provide formal privacy guarantees, such as DP. Moreover, they mostly rely on RNN-based architectures. In this work, we 1) introduced a Reversible Trajectory-to-CNN Transformations enabling the usage of CNN-based GANs from computer vision for trajectory generation, 2) integrated this transformation with the well-known DCGAN in a Proof-of-Concept implementation, 3) integrated both this PoC and the baseline work NTG with DP-SGD, and 4) evaluated the resulting four models across two datasets and four metrics. To the best of our knowledge, this is the first instance of a fully CNN-based GAN applied to trajectory generation and the initial application of DP-SGD to ensure privacy in this context. Although our PoC reproduces major density patterns of the datasets, it fails to capture details and is less adept at capturing sequential and temporal patterns. While the Proof-of-Concept does not provide sufficient utility for real-world application, this development opens promising avenues for further research into the application of generative models from computer vision. This study underlines the potential of CNN-based generative models to either supplement or replace existing RNN-based architectures in trajectory data privacy, encouraging further exploration in this promising research direction.

ACKNOWLEDGMENT

The authors would like to thank UNSW, the Commonwealth of Australia, and the Cybersecurity Cooperative Research Centre Limited for their support of this work. We thank the reviewers for their valuable feedback.

REFERENCES

[1] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the Crowd: The privacy bounds of human mobility," *Sci. Rep.*, vol. 3, no. 1, pp. 1–5, Dec. 2013.

[2] L. Rossi, J. Walker, and M. Musolesi, "Spatio-temporal techniques for user identification by means of GPS mobility data," *EPJ Data Sci.*, vol. 4, no. 1, p. 11, 2015.

[3] V. Primault, A. Boutet, S. B. Mokhtar, and L. Brunie, "The Long Road to Computational Location Privacy: A Survey," *IEEE Commun. Surv. Tutor.*, vol. 21, no. 3, pp. 2772–2793, 2019.

[4] O. Abul, F. Bonchi, and M. Nanni, "Never Walk Alone: Uncertainty for Anonymity in Moving Objects Databases," in *IEEE 24th Int. Conf. Data Eng.*, vol. 00. Cancun, Mexico: IEEE, Apr. 2008, pp. 376–385.

[5] L. Franceschi-Bicchierai, "Redditor cracks anonymous data trove to pinpoint Muslim cab drivers," 2015. Available: https://mashable.com/archive/redditor-muslim-cab-drivers

[6] F. Z. Errounda and Y. Liu, "An Analysis of Differential Privacy Research in Location and Trajectory Data," In Review, Preprint, Oct. 2020.

[7] F. Jin, W. Hua, M. Francia, P. Chao, M. Orlowska, and X. Zhou, "A Survey and Experimental Study on Privacy-Preserving Trajectory Data Publishing," Jan. 2021.

[8] A. Miranda-Pascual, P. Guerra-Balboa, J. Parra-Arnau, J. Forné, and T. Strufe, "SoK: Differentially Private Publication of Trajectory Data," *PoPETs*, pp. 496–516, 2023.

[9] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, no. 05, pp. 571–588, Oct. 2002.

[10] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Found. Trends Theor. CS.*, vol. 9, no. 3-4, pp. 211–407, 2013.

[11] V. Primault, S. B. Mokhtar, C. Lauradoux, and L. Brunie, "Differentially Private Location Privacy in Practice," 2014.

[12] J. Rao, S. Gao, Y. Kang, and Q. Huang, "LSTM-TrajGAN: A Deep Learning Approach to Trajectory Privacy Protection," *Leibniz Int. Proc. Inform.*, vol. 177, no. GIScience, pp. 1–16, 2020.

[13] Y. Qu, J. Zhang, R. Li, X. Zhang, X. Zhai, and S. Yu, "Generative adversarial networks enhanced location privacy in 5G networks," *Sci. China Inf. Sci.*, vol. 63, no. 12, p. 220303, Dec. 2020.

[14] T. Ma and F. Song, "A Trajectory Privacy Protection Method Based on Random Sampling Differential Privacy," *ISPRS Int. J. Geo-Inf.*, vol. 10, no. 7, p. 454, Jul. 2021.

[15] E. Naghizade, L. Kulik, E. Tanin, and J. Bailey, "Privacy- and Context-aware Release of Trajectory Data," *ACM Trans. Spat. Algorithms Syst.*, vol. 6, no. 1, pp. 1–25, Feb. 2020.

[16] E. Buchholz, A. Abuadbba, S. Wang, S. Nepal, and S. S. Kanhere, "Reconstruction Attack on Differential Private Trajectory Protection Mechanisms," in *Proc. 38th Annu. Comput. Secur. Appl. Conf.*, ser. ACSAC '22. New York, NY, USA: Association for Computing Machinery, Dec. 2022, pp. 279–292.

[17] T. Cunningham, K. Klemmer, H. Wen, and H. Ferhatosmanoglu, "GeoPointGAN: Synthetic Spatial Data with Local Label Differential Privacy," May 2022.

[18] M. Shao, J. Li, Q. Yan, F. Chen, H. Huang, and X. Chen, "Structured Sparsity Model Based Trajectory Tracking Using Private Location Data Release," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 6, pp. 2983–2995, 2020.

[19] X. Liu, H. Chen, and C. Andris, "trajGANs: Using generative adversarial networks for geo-privacy protection of trajectory data (Vision paper)," in *Locat. Priv. Secur. Workshop*, Melbourne, Australia, 2018, pp. 1–7.

[20] N. Ponomareva *et al.*, "How to DP-fy ML: A Practical Guide to Machine Learning with Differential Privacy," Mar. 2023.

[21] E. Buchholz, A. Abuadbba, S. Wang, S. Nepal, and S. S. Kanhere, "SoK: Can Trajectory Generation Combine Privacy and Utility?" *Proceedings on Privacy Enhancing Technologies*, vol. 2024, no. 3, July 2024.

[22] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," Jan. 2016. Available: http://arxiv.org/abs/1511.06434

[23] C. Donahue, J. McAuley, and M. Puckette, "Adversarial Audio Synthesis," Feb. 2019. Available: http://arxiv.org/abs/1802.04208

[24] A. Cheng, "PAC-GAN: Packet Generation of Network Traffic using Generative Adversarial Networks," in *2019 IEEE 10th Annu. Inf. Technol. Electron. Mob. Commun. Conf. IEMCON*. Vancouver, BC, Canada: IEEE, Oct. 2019, pp. 0728–0734.

[25] Dingqi Yang, Daqing Zhang, V. W. Zheng, and Zhiyong Yu, "Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs," *IEEE Trans. Syst. Man Cybern, Syst.*, vol. 45, no. 1, pp. 129–142, Jan. 2015.

[26] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in *Proc. 18th Int. Conf. World Wide Web*, ser. WWW '09. New York, NY, USA: Association for Computing Machinery, Apr. 2009, pp. 791–800.

[27] U. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response," in *Proc. 2014 ACM SIGSAC Conf.* New York, USA: ACM, Nov. 2014, pp. 1054–1067.

[28] F. McSherry, "Delta in Differential Privacy," Oct. 2022. Available: https://github.com/frankmcsherry/blog/blob/6aec29018eacc31c40e9dfd696f6c9ee591f8f5c/posts/2017-02-08.md

[29] F. McSherry and K. Talwar, "Mechanism Design via Differential Privacy," in *48th Annu. IEEE Symp. Found. Comput. Sci. FOCS07*. Providence, RI, USA: IEEE, Oct. 2007, pp. 94–103.

[30] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," in *2017 IEEE SP*. San Jose, CA, USA: IEEE, May 2017, pp. 3–18.

[31] M. Abadi *et al.*, "Deep Learning with Differential Privacy," in *Proc. 2016 ACM SIGSAC Conf*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 308–318.

[32] I. J. Goodfellow *et al.*, "Generative Adversarial Networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2014.

[33] S. Chintala, "How to Train a GAN? Tips and tricks to make GANs work," May 2023. Available: https://github.com/soumith/ganhacks

[34] D. Huttenlocher, G. Klanderman, and W. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, no. 9, pp. 850–863, Sept./1993.

[35] J. Hua, Y. Gao, and S. Zhong, "Differentially private publication of general time-serial trajectory data," in *IEEE INFOCOM*, vol. 26. Hong Kong, China: IEEE, Apr. 2015, pp. 549–557.

[36] M. Li, L. Zhu, Z. Zhang, and R. Xu, "Achieving differential privacy of trajectory data publishing in participatory sensing," *Inf. Sci.*, vol. 400–401, pp. 1–13, Aug. 2017.

[37] S. Chen, A. Fu, J. Shen, S. Yu, H. Wang, and H. Sun, "RNN-DP: A new differential privacy scheme base on Recurrent Neural Network for Dynamic trajectory privacy protection," *J. Netw. Comput. Appl.*, vol. 168, no. February, p. 102736, 2020.

[38] J. Zhang, Q. Huang, Y. Huang, Q. Ding, and P.-W. Tsai, "DP-TrajGAN: A privacy-aware trajectory generation model with differential privacy," *Future Gener. Comput. Syst.*, vol. 142, no. C, pp. 25–40, Dec. 2022.

[39] W. Jiang, W. X. Zhao, J. Wang, and J. Jiang, "Continuous Trajectory Generation Based on Two-Stage GAN," *J. Data Sci.*, vol. 19, no. 1, pp. 126–141, 2021.

[40] J. W. Kim and B. Jang, "Deep learning-based privacy-preserving framework for synthetic trajectory generation," *Journal of Network and Computer Applications*, vol. 206, p. 103459, Oct. 2022.

[41] Z. Zhang, X. Xu, and F. Xiao, "LGAN-DP: A novel differential private publication mechanism of trajectory data," *Future Gener. Comput. Syst.*, vol. 141, no. C, pp. 6392–703, Dec. 2022.

[42] R. Ozeki, H. Yonekura, H. Rizk, and H. Yamaguchi, "Balancing Privacy and Utility of Spatio-Temporal Data for Taxi-Demand Prediction," in *2023 24th IEEE Int. Conf. Mob. Data Manag. MDM*, vol. 24. Los Alamitos, CA, USA: IEEE, Jul. 2023, pp. 215–220.

[43] J. Shin, Y. Song, J. Ahn, T. Lee, and D.-H. Im, "TCAC-GAN: Synthetic Trajectory Generation Model Using Auxiliary Classifier Generative Adversarial Networks for Improved Protection of Trajectory Data," in *2023 IEEE Int. Conf. Big Data Smart Comput. BigComp*. Jeju, Republic of Korea: IEEE, Feb. 2023, pp. 314–315.

[44] Y. Song, J. Shin, J. Ahn, T. Lee, and D.-H. Im, "Except-Condition Generative Adversarial Network for Generating Trajectory Data," in *Database Expert Syst. Appl.*, ser. Lecture Notes in Computer Science, C. Strauss, T. Amagasa, G. Kotsis, A. M. Tjoa, and I. Khalil, Eds. Cham: Springer Nature Switzerland, 2023, pp. 289–294.

[45] X. Wang, X. Liu, Z. Lu, and H. Yang, "Large Scale GPS Trajectory Generation Using Map Based on Two Stage GAN," *J. Data Sci.*, vol. 19, no. 1, pp. 126–141, Feb. 2021.
[46] Y. Zhu, Y. Ye, S. Zhang, X. Zhao, and J. J. Q. Yu, "DiffTraj: Generating GPS Trajectory with Diffusion Probabilistic Model," in *Adv. Neural Inf. Process. Syst.*, ser. 1, vol. 23. New Orleans, USA: arXiv, 2023, p. 21.
[47] J. Jordon, J. Yoon, and M. van der Schaar, "PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees," in *International Conference on Learning Representations*, 2019.
[48] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially Private Generative Adversarial Network," Feb. 2018.
[49] J. Rao, S. Gao, Y. Kang, and Q. Huang, "LSTM-TrajGAN," 2020. Available: https://github.com/GeoDS/LSTM-TrajGAN
[50] S. Mehri *et al.*, "SampleRNN: An Unconditional End-to-End Neural Audio Generation Model," 2016.
[51] A. van den Oord *et al.*, "WaveNet: A Generative Model for Raw Audio," Sep. 2016. Available: http://arxiv.org/abs/1609.03499
[52] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Appl. Soft Comput.*, vol. 97, p. 105524, Dec. 2020.
[53] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," 2014.
[54] Erik Linder-Norén, "Pytorch-GAN," 2021. Available: https://github.com/eriklindernoren/PyTorch-GAN
[55] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," 2017.
[56] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," 2012.
[57] Soumith Chintala, "How to train a GAN, NIPS 2016," Aug. 2017. Available: https://www.youtube.com/watch?v=myGAju4L7O8
[58] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 San Diego CA USA May 7-9 2015 Conf. Track Proc.*, vol. abs1412.69, pp. 1–15, Dec. 2014.
[59] A. Yousefpour *et al.*, "Opacus: User-Friendly Differential Privacy Library in PyTorch," Sep. 2021. Available: https://arxiv.org/abs/2109.12298v4
[60] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved Training of Wasserstein GANs," in *Adv. Neural Inf. Process. Syst.*, ser. NIPS'17, vol. 30. Long Beach, California, USA: Curran Associates, Inc., 2017, pp. 5769–5779.
[61] H. Petzka, A. Fischer, and D. Lukovnicov, "On the regularization of Wasserstein GANs," Mar. 2018.
[62] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn. - Vol. 70*, ser. ICML'17. Sydney, NSW, Australia: JMLR.org, Aug. 2017, pp. 214–223.
[63] L. May Petry, C. Leite Da Silva, A. Esuli, C. Renso, and V. Bogorny, "MARC: A robust method for multiple-aspect trajectory classification via space, time, and semantic embeddings," *Int. J. Geogr. Inf. Sci.*, vol. 34, no. 7, pp. 1428–1450, 2020.
[64] I. Fontana, M. Langheinrich, and M. Gjoreski, "GANs for Privacy-Aware Mobility Modeling," *IEEE Access*, vol. 11, pp. 29 250–29 262, 2023.
[65] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs," Dec. 2017.

APPENDIX

A. Glossary

CNN Convolutional Neural Network.
DCGAN Deep Convolutional Generative Adversarial Network.
DL Deep Learning.
DP Differential Privacy.
DP-SGD Differentially Private Stochastic Gradient Descent, refer Section II-B.
EMD Earth Mover's Distance.
FC Fully Connected layer. Also called *Dense* or *Linear* layer.
FS-NYC Foursquare NYC [25].
GAN Generative Adversarial Network.
HD Hausdorff Distance.
LSTM Long Short-Term Memory.
LSTM-TrajGAN LSTM-TrajGAN [12].
MNIST-Seq MNIST Sequential Dataset: Images are transformed to sequences of length 28 with 28 features each [65].
NTG Noise-TrajGAN.
PoC Proof-of-Concept.
POI Point of Interest.
RNN Recurrent Neural Network.
RTCT Reversible Trajectory-to-CNN Transformations.
SOTA State Of The Art.
SWD Sliced Wasserstein Distance.
TRR Time Reversal Ratio.
TTD Total Travelled Distance.
TTUR Two Time Update Rule.
TUL Trajectory User Linking.
WD Wasserstein Distance.
WGAN Wasserstein GAN[62].
WGAN-LP WGAN with Lipschitz Penalty [61].