Modelling equivalent utility properties of the trace synthesis algorithm AdaTrace using differentially private GANs

Keyu Long, Yuchen Li

December 20, 2019

1 Introduction

Navigation, weather, entertainment, social and traffic applications increasingly rely on collecting personal location data to improve their performance and service their users [12]. Since users typically create accounts to send and receive these services, these applications easily collect location data across time as user traces. Previous work showed that trace anonymity does not prevent inference of users' private details, such as their transportation method, home and work places, religious and political beliefs, activities they engage in at a particular time and other people they meet on a regular basis [12].

Spatial and temporal defences introduce differential privacy by perturbing location points or increasing the time interval between measurements [9]. The problem with these two classes of defences is that the required noise and quantization for satisfactory privacy renders low utility for the structured and sparse nature of trace data [9]. Specialized query defences address this by assuming only a limited set of possible queries. For example, the AdaTrace [8] algorithm achieves top utility scores for queries on location, travel patterns and spatio-temporal travel metrics by efficiently partitioning its privacy budget across the specific purposes of quantizing the grid by the density of traffic and modelling mobility patterns, distributions of trips (i.e. pairs of start and end points) and trace length [8].

1.1 Objective

This paper examines if explicit optimization of the individual utilities in AdaTrace is necessary to achieve the same quality of synthesized private traces. Section 2 theoretically justifies that convergence of trace distributions to unprotected or differentially-private target distributions equivalently bounds convergence on trip and length distributions of the traces up to a constant multiplicative factor. Section 3 empirically evaluates the synthesized traces produced from vanilla and differentially private generative adversarial networks that are not engineered towards any particular utility. The goal is to show that some utilties belong in equivalent classes and that we can jointly optimize several utility objectives simultaneously under the same privacy budget.

1.2 Background work

Location privacy. Locations are points in 2 or 3-dimensional Euclidean space that benefit from popular point-based differential privacy mechanisms. For continuous location data, geo-indistinguishability mechanisms add variable noise that depends on location popularity [1, 5]. For discretized locations, location generalization mechanisms obfuscate the exact location of a user to an area containing multiple other candidates [6].

Trace privacy. Traces are temporally correlated locations, so privacy mechanisms will also need to protect point transition information. [7] modelled streams of motion as a Markov chain over regions with differentially private transition probabilities. [16] proposed location perturbations using the set of neighboring regions to which the location can transition into. The AdaTrace algorithm for synthesizing private traces encompasses the previous two techniques to specifically improve trace utility [8].

Quantifying trace privacy. Most existing trace privacy frameworks can be classified as either statistical (i.e. differential privacy) or syntactical (i.e. k-anonymity). Statistical frameworks are concerned about the privacy of query outcomes, for instance statistics on route length [8] and mobility patterns [3]. On the other hand, syntactical frameworks are concerned about restricting the ability of adversaries to link quasi-identifying tuples with sensitive values in the database [4, 13]. This paper focuses on differential privacy guarantees only.

Unsupervised neural approaches for learning latent features. Generative adversarial networks (GANs) consist of a generator and a discriminator where the discriminator transfers feedback to the generator to produce the target distribution [2]. This is a standard deep learning technique to retrieve latent feature distributions, which is useful for learning latent utilities of traces.

Rényi differential privacy (RDP). We can compute the privacy budget to train neural networks using Rényi differential privacy, which is a strictly stronger privacy definition compared to (ϵ, δ)-differential privacy [10, 15]. A mechanism f is (α, ϵ)-RDP if

$$D_\alpha(f(X)||f(X')) \leq \epsilon$$

where $X \sim X'$ are neighboring datasets and D_α is the Rényi divergence of order α [10]. [15] details the calculation of ϵ and [14] implements it in Python.

2 Theoretical results

We approximate traces as polynomial representations and seek to observe how the convergence of two distributions of these representations affects the convergence of their trip and length distributions. Note that the error margins we provide accounts for the noise required for differential privacy.

2.1 Notation and definitions

Notation 1. *We call $(\mathbf{p}, \mathbf{q}, \mathbf{c})$ the nth order polynomial representation of the trajectory l, if:*

1. $\mathbf{p} = (p_0, p_1)$ *is the start point of l,*
2. $\mathbf{q} = (q_0, q_1)$ *is the end point of l,*
3. $\mathbf{c} = (c_0, c_1, c_2, \cdots, c_n)$ *are the bounded coefficients of the nth order polynomial regression model of l, i.e. , such that*

$$\mathbf{c} = \arg \min_{\mathbf{c} \in [-B,B]^{n+1}} MSE(l, f_c(t)),$$

where

$$f_c(t) = c_n t^n + c_{n-1} t^{n-1} + \cdots + c_0.$$

Claim 1. *The universe $\mathcal{X}$ of possible parameters of nth order polynomial representation is $\mathbb{R}^{n+3}$.*

Proof. See Appendix 5.1. □

Note the fact that **p**, **q** and **c** are all bounded, the parameter universe $\mathcal{X}$ would be normalized to $[0,1]^{n+3}$ in our following discussions, *w.l.o.g.*

Definition 1 (Wasserstein-1 Distance)**.** *Let M be the Euclidean space, d be the corresponding Euclidean distance, and $P(M)$ denote the collection of all probability measures μ on M. The Wasserstein-1 Distance between two probability measures μ and μ' in P(M) is defined as:*

$$W(\mu, \mu') = \inf_{\gamma \in \Gamma(\mu,\mu')} \int_{M \times M} d(x, y) d\gamma(x, y),$$

where $\Gamma(\mu, \mu')$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively μ and μ'

Notation 2. *Denote the distribution of the real and the synthesis trajectory datasets on $\mathcal{X}$ as μ and μ' respectively. Let π denote the projection function:*

$$\begin{aligned} \pi : \quad \mathcal{X} &\rightarrow \quad [0,1]^4 \\ (\mathbf{p}, \mathbf{q}, \mathbf{c}) &\mapsto \quad (\mathbf{p}, \mathbf{q}) \end{aligned}$$

and denote the distribution of $(\mathbf{p}, \mathbf{q})$ on $[0,1]^4$ as μ_π and μ'_π respectively, which are the start-end distributions.

2.2 Accuracy bound on learned trip distribution

Theorem 1 (Accuracy on Start-end Distribution)**.** *If for a dataset with distribution μ, we can find a distribution μ' that is close to μ regarding Wasserstein-1 distance, i.e.*

$$W_1(\mu, \mu') < \alpha$$

then the corresponding start-end distributions μ_π and μ'_π would not be too different:

$$W_1(\mu_\pi, \mu'_\pi) < \alpha.$$

Proof.

$$\begin{aligned} W_1(\mu_\pi, \mu'_\pi) &= \inf_{\gamma \in \Gamma(\mu_\pi,\mu'_\pi)} \int_{[0,1]^4 \times [0,1]^4} d_4(x, y) d\gamma(x, y) \\ &\leq \inf_{\gamma \in \Gamma(\mu,\mu')} \int_{[0,1]^4 \times [0,1]^4} d_4(x, y) d\gamma(x, y) \\ &\leq \inf_{\gamma \in \Gamma(\mu,\mu')} \int_{[0,1]^{n+3} \times [0,1]^{n+3}} d_{n+3}(x, y) d\gamma(x, y) \quad (*) \\ &= W_1(\mu, \mu') \\ &< \alpha \end{aligned}$$

where

$$\begin{aligned} d_4(x, y) &= \sqrt{(p_0 - p'_0)^2 + (p_1 - p'_1)^2 + (q_0 - q'_0)^2 + (q_1 - q'_1)^2} \\ d_{n+3}(x, y) &= \sqrt{(p_0 - p'_0)^2 + (p_1 - p'_1)^2 + (q_0 - q'_0)^2 + (q_1 - q'_1)^2 + (c_n - c'_n)^2 + \cdots (c_0 - c'_0)^2} \end{aligned}$$

so the (*) is obvious for $\forall x, y$

□

2.3 Accuracy bound on learned length distribution

Notation 3. *Let μ_L and μ'_L denote the length distributions of μ and μ', which are the distributions of the corresponding points on $\mathbb{R}$ under mapping τ:*

$$\begin{array}{rccl} \tau: & \mathcal{X} & \to & \mathbb{R} \\ & (\mathbf{p},\mathbf{q},\mathbf{c}) \mapsto & & \displaystyle\int_{p_0}^{q_0} \sqrt{(1+(f'_c(t))^2}\, dt \end{array}$$

Claim 2. *For $\forall x, y \in \mathcal{X}$, we have*

$$\left| \int_{p_0}^{q_0} \sqrt{1+(f'_c(t))^2}\, dt - \int_{p'_0}^{q'_0} \sqrt{1+(f'_{c'}(t))^2}\, dt \right| \le M_n \cdot d_{n+3}(x,y)$$

where M_n is a constant that only relates to the n (the dimension of the parameter universal $\mathcal{X}$).

Proof. See Appendix 5.2. □

Theorem 2 (Accuracy on Length Distribution)**.** *If we have*

$$W_1(\mu,\mu') < \alpha,$$

then

$$W_1(\mu_L,\mu'_L) < M_n \cdot \alpha,$$

Proof.

$$\begin{aligned} W_1(\mu_\pi,\mu'_\pi) &= \inf_{\gamma\in\Gamma(\mu_L,\mu'_L)} \int_{\mathbb{R}} d(x,y)d\gamma(x,y) \\ &\le \inf_{\gamma\in\Gamma(\mu,\mu')} \int_{\mathbb{R}} d(x,y)d\gamma(x,y) \\ &= \inf_{\gamma\in\Gamma(\mu,\mu')} \int_{\mathbb{R}} \left| \int_{p_0}^{q_0} \sqrt{1+(f'_c(t))^2}\, dt - \int_{p'_0}^{q'_0} \sqrt{1+(f'_{c'}(t))^2}\, dt \right| d\gamma(x,y) \\ &\le \inf_{\gamma\in\Gamma(\mu,\mu')} \int_{[0,1]^{n+3}\times[0,1]^{n+3}} M_n \cdot d_{n+3}(x,y)d\gamma(x,y) \quad \text{(from Claim 2)} \\ &= M_n \cdot W_1(\mu,\mu') \\ &< M_n \cdot \alpha \end{aligned}$$

□

3 Empirical study

The theoretical results suggest that we should empirically observe a monotonic improvement of trip and length distributions with the minimization of the Wasserstein-1 distance between trace distributions. There should be no need to optimize the two utilities separately like AdaTrace [8]. The Wasserstein generative adversarial network (WGAN) is a general purpose algorithm for minimizing the Wasserstein-1 distance between a parameterized distribution and a target [2]. Additionally, we can also train the critic using Rényi differentially private (RDP) stochastic gradient descent [10, 15]. Since ϵ-RDP is strictly stronger than (ϵ,δ)-DP [10], it is fair to directly observe the effect of the privacy budget ϵ on the quality of synthesized traces compared to that of AdaTrace.

3.1 Methods

Appendix figure 3 describes the exact specifications for the architecture of the generator and critics. The critic uses spectral normalization to maintain 1-Lipschitz gradients [11]. For RDP-critics, we use the PyVacy implementation of RDP-Adam [14] with learning rate 2×10^{-4}, $\beta_1 = 0$ and $\beta_2 = 0.999$ over 15,000 epochs. We balance the noise multiplicative constant with batch size 64 and 128 to achieve $\epsilon = 1.5$ and $\epsilon = 40$, respectively, for two different models, as computed by the RDP accountant implemented in PyVacy [14]. Each critic trains for two iterations before the generator is optimized once, using Adam with learning rate 5×10^{-5}. These values were chosen based on the best observed performance of a vanilla WGAN on Brinkhoff, which is the same trace dataset used in [8] for reporting results of AdaTrace. We prepare Brinkhoff for critic input by scaling all points linearly into $[-1, 1]^2$ and padding the sequence of points to the longest trajectory length of 240. This is concatenated along the time axis with the mask corresponding to the original sequence, where each 0.5 value in the mask corresponds to non-padded values and -0.5 corresponds to the padded values. The final input to the critic and target output of the generator is a matrix is of shape 3×240. The code to replicate the experiments is in

https://github.com/ychnlgy/Private-Trace-Synthesis.

3.2 Results

Figure 1 compares the same 7 utility metrics of AdaTrace [8] to synthesized datasets from vanilla and differentially private WGANs. See [8] for details, but in brief they compare the following distributions of utility features across the synthetic and original dataset:

1. Query error measures differences in the number of traces passing through regions
2. Location popularity Kendall-tau coefficient measures changes in popularity ranking of regions
3. Frequent travel pattern (FP) F1 measures frequency of traces that span sequences of regions
4. FP error measures differences in the number of traces that pass through sequences of regions
5. Trip error measures differences in distributions of trajectory start and destination regions
6. Diameter error measures differences in distributions of distance between adjacent nodes
7. Length error measures differences in distributions of total distances covered per trajectory

While the vanilla WGAN appears to improve monotonically across all metrics and scores better than AdaTrace in all but trip error at the end of training, this trend becomes less clear when we use RDP Adam to train the critic with $\epsilon = 40$ and absent with $\epsilon = 1.5$.

To visualize the implications of the scores, we randomly sample 100 traces from the trained generators (Figure 2). The vanilla WGAN appears to generate the popular traces well as expected, but becomes noisy along areas of less traffic. AdaTrace produces noisier traces that form a noticeable grid-like pattern near the dense center regions, which aligns well with their adaptive mesh algorithm. The traces of RDP-WGAN appear to be contained in the range of traces in the original dataset, but the sequence of nodes appear to largely deviate in interval.

4 Discussion and Conclusion

Although Figure 1 indicates that AdaTrace produces superior utility metrics compared to RDP-WGAN methods, this might not be the case if we decrease granularity of the mesh on which evaluation of the traces is carried out. For frequent travel pattern F1, [8] chose a 6×6 uniform

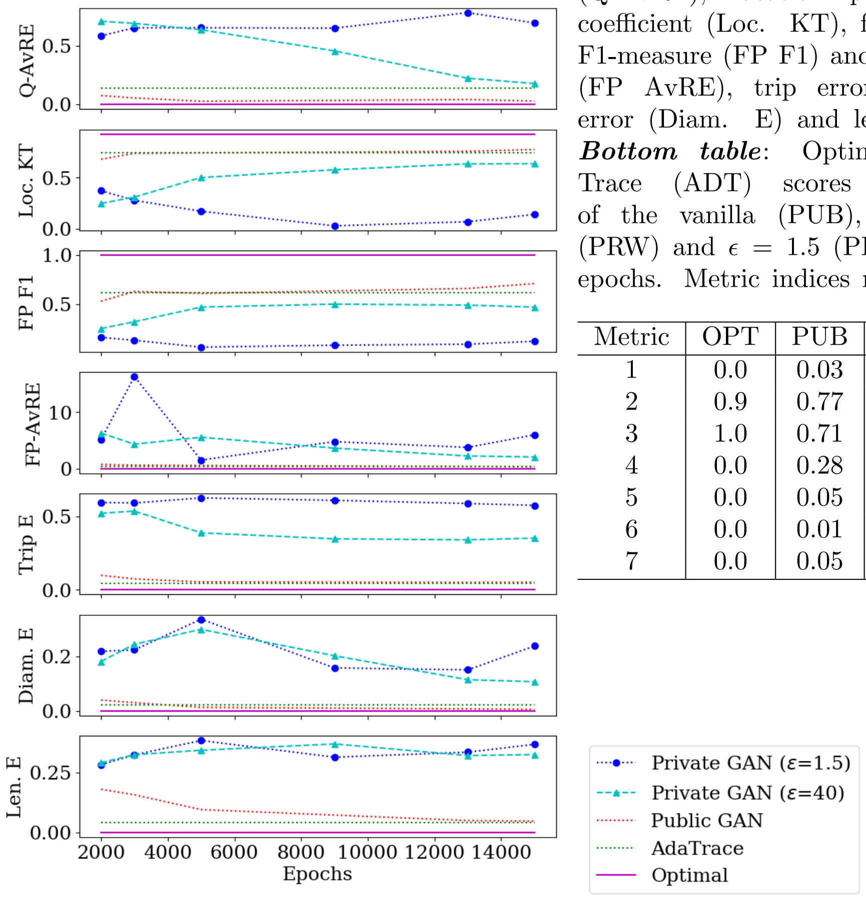

Figure 1: ***Left figure***: Utility scores over training epochs for query average relative error (Q-AvRE), location popularity Kendall-tau coefficient (Loc. KT), frequent travel pattern F1-measure (FP F1) and average relative error (FP AvRE), trip error (Trip E), diameter error (Diam. E) and length error (Len. E). ***Bottom table***: Optimal (OPT) and AdaTrace (ADT) scores versus final scores of the vanilla (PUB), DP with $\epsilon = 40$ (PRW) and $\epsilon = 1.5$ (PRV) WGANs at 15000 epochs. Metric indices refer to rows of figure.

Metric	OPT	PUB	ADT	PRW	PRV
1	0.0	0.03	0.14	0.18	0.70
2	0.9	0.77	0.74	0.63	0.14
3	1.0	0.71	0.62	0.47	0.12
4	0.0	0.28	0.38	2.04	5.98
5	0.0	0.05	0.04	0.35	0.58
6	0.0	0.01	0.02	0.11	0.24
7	0.0	0.05	0.04	0.33	0.37

grid, which appears to benefit the coarse and noisy traces produced by AdaTrace but not the restricted range of RDP-WGAN methods as shown by the visualization of the trajectories (Figure 2). For instance, the RDP-WGAN approach may do better for query and trip error and on grids with cell granualities that capture a large proportion of the empty space in Brinkhoff, simply because traces from AdaTrace cover more area (Figure 2).

Another source of uncertainty is from the training of the WGANs. Both quantitative and qualitative results seemed to indicate that the vanilla WGAN did not converge completely, suggesting that this model is insufficient in some way to fully model the trace data. The privacy budget may have compounded with the limitations of the method, further deteriorating the results from the RDP-WGANs. The results indicate it is not trivial to convert a working vanilla WGAN system into a differentially private one.

The utility metrics implemented by [8] suggest that RDP-WGAN synthesizes strictly worse quality traces relative to AdaTrace. As such, RDP-WGANs should not be used when privacy is of concern; on the otherhand, WGAN approaches may provide many equivalent classes of utility without explicit specification.

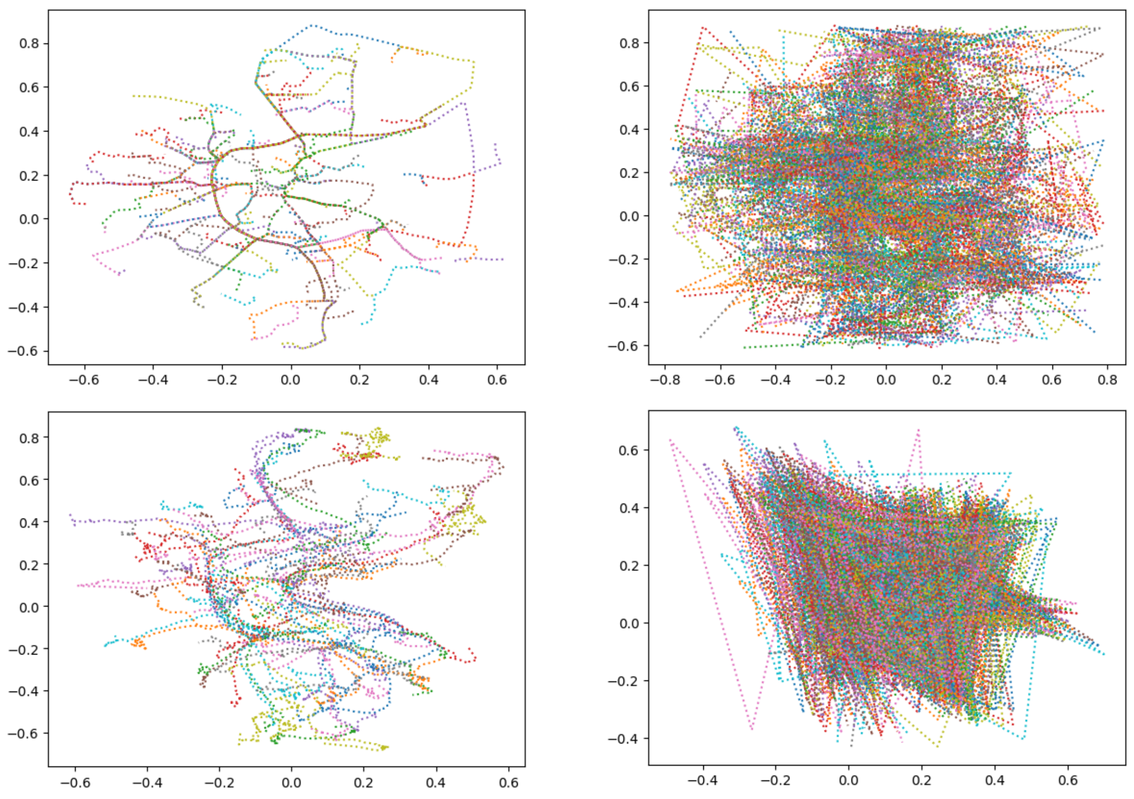

Figure 2: Comparison of 100 randomly sample traces from the original Brinkhoff dataset (*top-left*) versus synthesized traces by AdaTrace (*top-right*), trained vanilla (*bottom-left*) and $\epsilon = 1.5$ RDP-WGAN (*bottom-right*) generators. Colors are meant to distinguish between different traces. Visualization of the effect of decreasing the privacy budget is shown in Appendix figure 4.

4.1 Open questions

1. Can we group utility metrics into equivalent classes in a systematic way? It may turn out that the utilities that we optimize for individually may actually overlap, so we can save on the privacy budget.

2. Training the discriminator in a WGAN with differentially-private optimization techniques appears to be wasteful, especially since the discriminator requires several more iterations than the generator and will eventually be discarded anyways. Is there a way to be more efficient with the privacy budget?

References

[1] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. *arXiv preprint arXiv:1212.1984*, 2012.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan, 2017.

[3] V. Bindschaedler and R. Shokri. Synthesizing plausible privacy-preserving location traces. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 546–563. IEEE, 2016.

[4] K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. Anonymity protocols as noisy channels. *Information and Computation*, 206(2-4):378–401, 2008.

[5] E. ElSalamouny and S. Gambs. Differential privacy models for location-based services. *Transactions on Data Privacy*, 2016.

[6] B. Gedik and L. Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, 2007.

[7] M. Götz, S. Nath, and J. Gehrke. Maskit: Privately releasing user context streams for personalized mobile applications. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 289–300. ACM, 2012.

[8] M. E. Gursoy, L. Liu, S. Truex, L. Yu, and W. Wei. Utility-aware synthesis of differentially private and attack-resilient location traces. *ACM SIGSAC Conference on Computer and Communications Security*, 2018. doi: https://doi.org/10.1145/3243734.3243741.

[9] J. Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13:391–399, 2009. doi: https://doi.org/10.1007/s00779-008-0212-5.

[10] I. Mironov. Renyi differential privacy. *CoRR*, abs/1702.07476, 2017. URL `http://arxiv.org/abs/1702.07476`.

[11] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018. URL `http://arxiv.org/abs/1802.05957`.

[12] V. Primault, A. Boutet, S. B. Mokhtar, and L. Brinoe. The long road to computational location privacy: a survey. *arXiv preprint: 1810.03568*, 2018.

[13] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux. Quantifying location privacy. In *2011 IEEE symposium on security and privacy*, pages 247–262. IEEE, 2011.

[14] C. Waites. PyVacy: Privacy Algorithms for PyTorch, 2019. URL `https://github.com/ChrisWaites/pyvacy`.

[15] Y. Wang, B. Balle, and S. P. Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. *CoRR*, abs/1808.00087, 2018. URL `http://arxiv.org/abs/1808.00087`.

[16] Y. Xiao and L. Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1298–1309. ACM, 2015.

5 Appendix

5.1 Proof for Claim 1

Proof. The dependencies

$$p_1 = c_n p_0^n + c_{n-1} p_0^{n-1} \cdots + c_0$$
$$q_1 = c_n q_0^n + c_{n-1} q_0^{n-1} \cdots + c_0$$

reduce the dimension of $(\mathbf{p}, \mathbf{q}, \mathbf{c})$ by 2, from $(2 + 2 + (n + 1))$ to $(n + 3)$. □

5.2 Proof for Claim 2

First, we make the following claim:

Claim 3.

$$a_1 + a_2 + \cdots + a_n \leq 2^{(n-1)/2} \cdot \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$$

Proof. We use induction to get the result:

1. when $n = 2$, it is obvious that

$$a_1 + a_2 \leq \sqrt{2(a_1^2 + a_2^2)}$$

2. if the claim holds for n, then for the case $n+1$:

$$\begin{aligned} a_1 + a_2 + \cdots + a_n + a_{n+1} &\leq \sqrt{2((a_1 + a_2 + \cdots + a_n)^2 + a_{n+1}^2)} \\ &\leq \sqrt{2(2^{n-1}(a_1^2 + a_2^2 + \cdots + a_n^2) + a_{n+1}^2)} \\ &\leq \sqrt{2^n(a_1^2 + a_2^2 + \cdots + a_{n+1}^2)} \\ &= 2^{(n+1-1)/2}\sqrt{a_1^2 + a_2^2 + \cdots + a_{n+1}^2} \end{aligned}$$

□

Now we prove Claim 2.

Proof.

$$\left| \int_{p_0}^{q_0} \sqrt{1 + (f_c'(t))^2}\, dt - \int_{p_0'}^{q_0'} \sqrt{(1 + (f_{c'}'(t))^2}\, dt \right|$$

$$= \left| \int_{p_0}^{q_0} \sqrt{1 + (f_c'(t))^2}\, dt - \int_{p_0}^{q_0} \sqrt{(1 + (f_{c'}'(t))^2}\, dt + \int_{p_0}^{q_0} \sqrt{1 + (f_{c'}'(t))^2}\, dt - \int_{p_0'}^{q_0'} \sqrt{1 + (f_{c'}'(t))^2}\, dt \right|$$

$$\leq \left| \int_{p_0}^{q_0} \left(\sqrt{1 + (f_c'(t))^2} - \sqrt{1 + (f_{c'}'(t))^2} \right) dt \right| + |F_{c'}(q_0) - F_{c'}(p_0) - (F_{c'}(q_0') - F_{c'}(p_0'))|$$

(where $F_{c'} = \int \sqrt{1 + (f_{c'}'(t))^2}\, dt$; note that this is legal since $\sqrt{1 + (f_{c'}'(t))^2}$ is integrable.)

$$= \left| \int_{p_0}^{q_0} \frac{(1 + (f_c'(t))^2) - (1 + (f_{c'}'(t))^2)}{\sqrt{1 + (f_c'(t))^2} + \sqrt{1 + (f_{c'}'(t))^2}}\, dt \right| + |F_{c'}(q_0) - F_{c'}(q_0') + F_{c'}(p_0) - F_{c'}(p_0')|$$

$$\leq \left| \int_{p_0}^{q_0} \frac{(f_c'(t))^2 - (f_{c'}'(t))^2}{2}\, dt \right| + |F_{c'}(q_0) - F_{c'}(q_0') + F_{c'}(p_0) - F_{c'}(p_0')| \qquad (**)$$

(the coefficients of f_c and f_c' are bounded in $[0,1]^n$, and so are p_0 and q_0.)

1. For $\int_{p_0}^{q_0} \frac{(f'_c(t))^2-(f'_{c'}(t))^2}{2}\,dt$:

$$\begin{aligned}
&\because f'_c(t)^2 - f'_{c'}(t)^2\\
&= (nc_nt^{n-1} + (n-1)c_{n-1}t^{n-2} + \cdots + c_0)^2 - (nc'_nt^{n-1} + (n-1)c'_{n-1}t^{n-2} + \cdots + c'_0)^2\\
&= n^2t^{2n-2}c_n^2 + (n-1)^2t^{2n-4}c_{n-1}^2 + \cdots + c_0^2 + 2n(n-1)t^{2n-3}c_nc_{n-1} + \cdots + 2tc_1c_0\\
&\quad - (n^2t^{2n-2}(c'_n)^2 + (n-1)^2t^{2n-4}(c'_{n-1})^2 + \cdots + (c'_0)^2 + 2n(n-1)t^{2n-3}c'_nc'_{n-1} + \cdots + 2tc'_1c'_0)\\
&= n^2t^{2n-2}(c_n + c'_n)(c_n - c'_n) + \cdots + (c_0 + c'_0)(c_0 - c'_0)\\
&\quad + 2n(n-1)t^{2n-3}(c_nc_{n-1} - c'_nc'_{n-1}) + \cdots + 2t(c_1c_0 - c'_1c'_0)\\
&= n^2t^{2n-2}(c_n + c'_n)(c_n - c'_n) + \cdots + (c_0 + c'_0)(c_0 - c'_0)\\
&\quad + 2n(n-1)t^{2n-3}(c_nc_{n-1} - c_nc'_{n-1} + c_nc'_{n-1} - c'_nc'_{n-1}) + \cdots + 2t(c_1c_0 - c_1c'_0 + c_1c'_0 + c'_1c'_0)\\
&\le M_1(|c_n - c'_n| + \cdots + |c_0 - c'_0|) + M_2(|c_n - c'_n| + \cdots + |c_0 - c'_0|)\\
&\text{where } M_1 = 2n^2, M_2 = 4n(n-1) \quad (\text{since } t \in [0,1] \text{ and } c \in [0,1])\\
&\le (M_1 + M_2)(|c_n - c'_n| + \cdots + |c_0 - c'_0|)\\
&\text{and according to claim 3, we have}\\
&\le (M_1 + M_2)2^{(n-1)/2}\sqrt{(c_0 - c'_0)^2 + \cdots + (c_n - c'_n)^2},
\end{aligned}$$

$$\begin{aligned}
\therefore \int_{p_0}^{q_0} \frac{(f'_c(t))^2 - (f'_{c'}(t))^2}{2}\,dt &\le \int_{p_0}^{q_0} \frac{(M_1 + M_2)\sqrt{(c_0 - c'_0)^2 + \cdots + (c_n - c'_n)^2}}{2}\,dt\\
&\le (q_0 - p_0)\frac{(M_1 + M_2)\sqrt{(c_0 - c'_0)^2 + \cdots + (c_n - c'_n)^2}}{2}\\
&\le (M_1 + M_2)\cdot d_3(x, y)
\end{aligned}$$

2. For $|F_{c'}(q_0) - F_{c'}(q'_0) + F_{c'}(p_0) - F_{c'}(p'_0)|$:

$$\begin{aligned}
|F_{c'}(q_0) - F_{c'}(q'_0) + F_{c'}(p_0) - F_{c'}(p'_0)| &\le |F_{c'}(q_0) - F_{c'}(q'_0)| + |F_{c'}(p_0) - F_{c'}(p'_0)|\\
&\le M_3(|q_0 - q'_0| + |p_0 - p'_0|)
\end{aligned}$$

where $M_3 = \max\limits_{t\in[0,1]}$. From Claim 3, we have

$$\begin{aligned}
&\le \sqrt{2}M_3 \cdot \sqrt{(q_0 - q'_0)^2 + (p_0 - p'_0)^2}\\
&\le \sqrt{2}M_3 \cdot d_3(x, y)
\end{aligned}$$

Therefore,

$$(**) \quad \le \quad (M_1 + M_2 + \sqrt{2}M_3)\cdot d_3(x, y) \quad = \quad M_n \cdot d_3(x, y)$$

$\square$

5.3 Additional experimental details and results

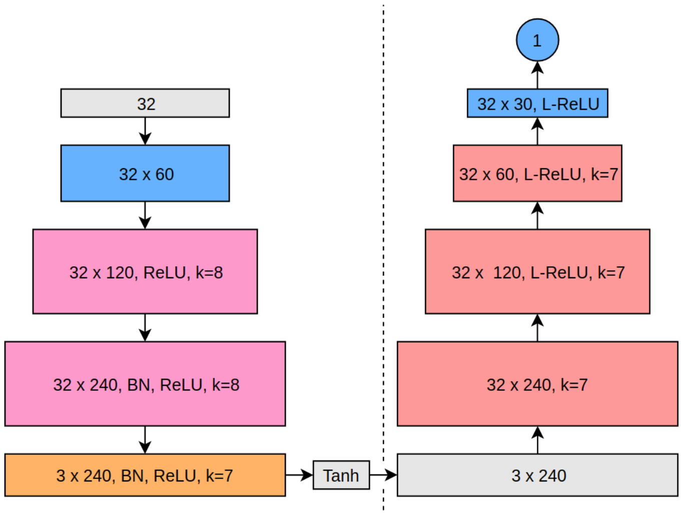

Figure 3: Architecture of the generator (*left*) and discriminator (*right*) used in the experiments. Grey blocks represent tensors with no parameters, blue blocks represents fully connected layers, pink blocks represent convolution transpose upsampling, orange blocks represent convolutional layers, and red blocks represent convolutional layers with spectral normalization [11]. All convolutional layers are 1D, BN means batchnorm, L-ReLU means leaky ReLU with slope 0.01 and k represents the kernel size for the convolution. Convolution stride and padding are chosen to satisfy the input and output tensor shapes.

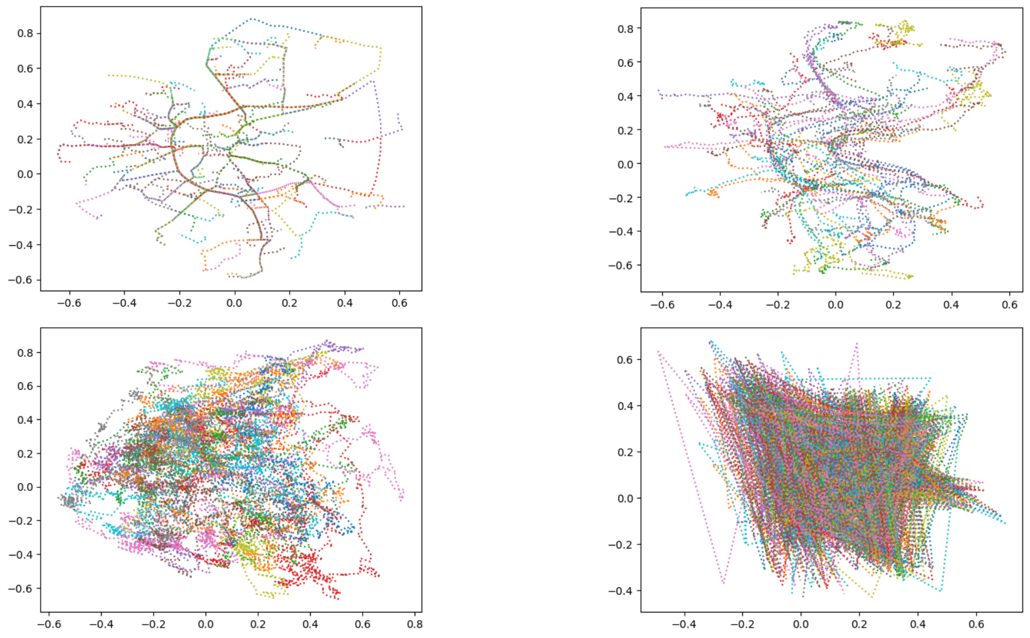

Figure 4: Comparison of 100 randomly sampled traces from the original Brinkhoff dataset (*top-left*) versus synthesized traces by trained vanilla (*top-right*), $\epsilon = 40$ (*bottom-left*) and $\epsilon = 1.5$ RDP-WGAN (*bottom-right*) generators. Colors are meant to distinguish between different traces.