A Neural Approach to Spatio-Temporal Data Release with User-Level Differential Privacy

ABSTRACT

Several companies (e.g., Meta, Google) have initiated "data-for-good" projects where aggregate location data are first sanitized and released publicly, which is useful to many applications in transportation, public health (e.g., COVID-19 spread) and urban planning. *Differential privacy (DP)* is the protection model of choice to ensure the privacy of the indivduals who generated the raw location data. However, current solutions fail to preserve data utility when each individual contributes multiple location reports (i.e., under user-level privacy). To offset this limitation, public releases by aggregators like Meta or Google use high privacy budgets (e.g., ε = 10-100), which results in poor privacy protection. In this paper, we propose a novel approach to releasing spatio-temporal data in a private and accurate way. We employ the pattern recognition power of neural networks, specifically variational auto-encoders (VAE), to reduce the noise introduced by DP mechanisms such that accuracy is increased, while the privacy requirement is still satisfied. Our extensive experimental evaluation on real datasets shows the clear superiority of our approach compared to benchmarks.

ACM Reference Format:

. 2022. A Neural Approach to Spatio-Temporal Data Release with User-Level Differential Privacy. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Mobile apps make extensive use of individual location data to customize user experience. In the process, service providers gather huge amounts of spatio-temporal user traces, which can be highly beneficial in healthcare, transportation and environmental research. Several "data-for-good" projects [1, 5, 10], that release such spatio-temporal datasets to the public have been initiated by major companies (e.g., Meta, Google), and the released data have been utilized for purposes such as improving COVID-19 spread modeling [17, 48] and understanding human mobility [14, 24]. Most often, spatio-temporal data releases are provided in the form of snapshot high resolution population density information, where the released statistics capture population counts in small areas for short time periods. Since high resolution is required for utility (e.g., in modeling COVID hotspots) privacy risks are elevated. To prevent malicious actors from using the data to infer sensitive details about individuals, the released datasets must be first sanitized. Typically, [1, 5, 7, 10], the de-facto standard of *differential privacy (DP)* is employed as protection model, due to its formal statistical protection guarantees that limit the ability of an adversary to learn whether a particular individual's data has been included in the release or not.

Most existing work on DP-compliant publication of location data focused on *single snapshot* releases, where each individual contributes a *single location report* [9, 20, 28, 38, 40, 45, 46, 60, 65, 66], and *event-level* privacy is sufficient for protection. However, when releasing *multiple* snapshots, individuals contribute multiple reports. The ability of an adversary to breach privacy increases significantly, and a shift to *user-level* privacy [21] is required. To protect privacy in this scenario, an increased amount of noise is needed, which often grows linearly in the number of user contributions. Only a handful of techniques [8, 54] considered spatio-temporal location data release, and none of them is able to preserve data accuracy for any practical spatio-temporal resolutions.

In the absence of approaches for high resolution spatio-temporal data release, industry projects used basic DP mechanisms that simply add noise to the population density information without taking into account specific dataset characteristics [1, 5, 10, 14, 30]. The amount of privacy budget spent for such data releases is often not reported, or it is excessive [7, 14, 30], thus providing insufficient protection. In addition, reports of incorrect privacy accounting in such releases [14, 30] further necessitate a thorough end-to-end study of custom DP algorithms for spatio-temporal data.

There are two key aspects that must be addressed for accurate private releases of spatio-temporal data. First, one needs to bound *sensitivity* (please see Section 2 for a formal definition) by limiting the number of location reports from any single user, which can be achieved through sampling. However, this must be done carefully, such that data utility is not compromised. Furthermore, density information must be adjusted to account for the fact that they are calculated on a subset of the actual data. Second, the effect of noise added by DP mechanisms must be addressed. Such mechanisms consider the worst-case scenario over all possible data distributions and query configurations, and err on the safe side, adding more noise than strictly necessary, compromising accuracy. A *denoising* post-processing step that leverages spatio-temporal data characteristics can significantly boost accuracy, while still satisfying the privacy constraint. Recent advances in neural networks, such as variational auto-encoders (VAE), are good at capturing complex density patterns, and can enable effective denoising.

We propose VAE-based Density Release (VDR), a novel system specifically designed to address accurate, DP-compliant release of spatio-temporal datasets. A key intuition behind VDR is the observation that noisy spatio-temporal data histograms are similar to a sequence of images, with spatial patterns in location data akin to visual patterns in images. This observation allows one to leverage a vast amount of work on image pattern recognition and apply it to spatio-temporal data releases. VDR first sanitizes density

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

information by adding DP-compliant noise, then uses a novel neural network-based approach to improve accuracy by performing an advanced post-processing *denoising* step based on convolutional neural networks (CNN). CNNs are able to capture subtle patterns specific to location datasets. We utilize *variational auto-encoders (VAE)* to capture data patterns without fitting to the noise. We also employ multi-resolution learning through a data augmentation step that captures location data patterns at multiple granularities, thus improving accuracy for a broad range of query extents.

We devise a comprehensive strategy to reduce sensitivity of user-level privacy through sampling. We reduce the number of input samples from any individual in a DP-compliant way. To counterbalance the effect of sampling, we design a novel private statistical estimator which scales up query results to preserve accuracy. This permits us to control the sensitivity in user-level privacy without significantly affecting accuracy.

Specifically, our contributions are:

- We propose an end-to-end privacy-preserving system for spatio-temporal datasets that satisfies user-level differential privacy and preserves data accuracy;
- We introduce a novel approach to user-level sampling that reduces sensitivity by bounding the number of location reports from each user; the bound is chosen to provide a good trade-off between keeping sensitivity low and preserving density information across time;
- We design a novel denoising technique that uses variational auto-encoders and image feature extraction concepts to accurately model patterns in spatio-temporal data;
- We design a technique to offset the effects of location sampling in order to provide accurate query answers; to that extent, we employ DP-compliant statistical estimators;
- We perform an extensive experimental evaluation on real data which shows that the proposed approach is effective in preserving query accuracy under strict privacy budgets, and clearly outperforms all existing approaches.

We provide background information and formulate the studied problem in Section 2. Section 3 introduces the proposed sampling and denoising techniques. Section 4 explores system design trade-offs. Section 5 presents the experimental evaluation results. We survey related work in Section 6 and conclude in Section 7.

2 BACKGROUND

Differential Privacy. ε-differential privacy [22] provides a rigorous framework with formal protection guarantees. Given *privacy budget* parameter $\varepsilon \in (0, +\infty)$, a randomized mechanism $\mathcal{M}$ satisfies ε-differential privacy iff for any *sibling* datasets D and D' differing in a single tuple, and for all $E \subseteq \text{Range}(\mathcal{M})$

$$\Pr[\mathcal{M}(D) \in E] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in E] \quad (1)$$

$\Pr[\mathcal{M}(D) \in E]$ denotes the probability of mechanism $\mathcal{M}$ outputting an outcome in the set E for a database D, and $\text{Range}(\mathcal{M})$ is the co-domain of $\mathcal{M}$. Mechanism $\mathcal{M}$ hides the presence of an individual in the data by ensuring that for any given outcome, the probability difference between any two sibling datasets being the source of that outcome is bounded by e^{ε}. The protection provided by DP is stronger when ε approaches 0.

The *sensitivity* of a function (e.g., a query) f, denoted by Δ_f, is the maximum amount the value of f can change when adding or removing a single individual's contribution from the data. The ε-DP guarantee can be achieved by adding random noise derived from the Laplace distribution $\text{Lap}(\Delta_f/\varepsilon)$. For a query $f : D \rightarrow \mathbb{R}$, the *Laplace mechanism* (LPM) $\mathcal{M}$ returns $f(D) + \text{Lap}(\Delta_f/\varepsilon)$, where $\text{Lap}(\Delta_f/\varepsilon)$ is a sample drawn from the probability density function $\text{Lap}(x|(\Delta_f/\varepsilon)) = (\varepsilon/2\Delta_f)\exp(-|x|\varepsilon/\Delta_f)$ [22].

Sensitivity is significantly influenced by whether one considers *event-level* or *user-level* privacy. In the studied case of spatio-temporal data, f can be formulated as a vector-valued function that outputs the population count in a location histogram at *every* time snapshot. When an individual's data are removed, all the location reports of that individual are deleted from the data, causing changing in multiple elements of f's output (in the worst case, the change is proportional to the maximum number of location reports across all individuals). Contrast this with event-level privacy, where one considers sibling datasets to differ in a single location report. As a result, enforcing user-level privacy causes a significant increase in sensitivity, which must be carefully controlled to prevent utility loss due to excessive noise addition.

Problem Definition. We are given a dataset D consisting of user location reports collected over time. Each user report consists of four attributes: latitude (*lat*), longitude (*lon*), timestamp (*time*) and user id. The goal is to release high-resolution density information of D for arbitrary spatial regions over time. We do this by releasing an $\mathbb{M}\times\mathbb{M}\times\mathbb{T}$ histogram of the data, where $\mathbb{M}$ determines the spatial resolution and $\mathbb{T}$ determines the temporal resolution. M and $\mathbb{T}$ are determined by the application requirements, e.g., release a histogram at 30×30m resolution and one hour time granularity over a duration of 24 hours [5, 38]. More specifically, we design a mechanism $\mathcal{M}$ that takes D as an input and outputs a histogram, $\hat{H}$, where $\mathcal{M}$ preserves ε-DP. The histogram of the original (i.e., non-private) data with true counts is denoted as H. In real-world scenarios, the DP histogram is used to answer spatio-temporal queries. In this paper we focus on the following three representative statistical query types on spatio-temporal aggregate data:

Range count queries. Given a query range, defined by minimum and maximum values (i.e., a range) for *lat*, *lon* and *time*, find the number of user location reports in D that satisfy this range predicate. For a query q, we measure the utility of its estimated DP-preserving answer, y, compared to the true answer u using the *relative error metric*, defined as $\text{RE}(y, u) = \frac{|y-u|}{\max\{u,\psi\}}$, where ψ is a smoothing factor necessary to avoid division by zero.

Nearest hot-spot queries. Given a query location q (lat, lon, time), a density threshold, ν, and a spatio-temporal extent SR (defined by a time duration and lengths of lat and lon geo-coordinates), find the closest cell to query q within extents SR that contains at least ν number of locations signals. The hotspot query may be answered using an expanding search in the 3-d histogram until a cell within SR having at least ν points is found. If none is found, the cell with the maximum count is reported. We evaluate this query in two ways: the distance penalty is measured as the Mean Absolute Error (MAE) between the true distance (as computed on on H) and reported distance (computed on $\hat{H}$) to the hotspot. To capture hotspot density estimation errors, we measure *Regret*, defined as

the deviation of the reported density of the found hotspot (on noisy histogram $\hat{H}$) from the specified threshold ν. Regret for a query is zero if the reported hotspot meets the density threshold.

Forecasting query. Given a timeseries of density counts for a 2-d region (defined with minimum and maximum *lat* and *lon* values), and a forecasting horizon h *not covered* within the timeseries, predict the count of location reports for h future timesteps. To evaluate this query, we utilize *holdout testing*, which removes the last h data points of the timeseries, calculates the forecasting model fit for the remaining historical data, makes forecasts for h timesteps, and compares the error between the forecast points and their corresponding, *held-out*, data points. We report the symmetric mean absolute percentage errors (sMAPE) as $\text{sMAPE} = \frac{1}{h}\sum_{t=1}^{h}\frac{|F_t - A_t|}{(A_t+F_t)/2}$, where A_t are the true counts from H in the h timesteps and F_t are the h predicted counts from a forecasting algorithm fitted to the historical data points from $\hat{H}$.

3 VAE-BASED DENSITY RELEASE (VDR)

Given a dataset of location reports over time, we perform a careful sanitization process to create a DP-compliant spatio-temporal histogram. Any number of computations performed on the histogram after it is sanitized with DP are considered safe. The histogram may be thus publicly released and used to answer arbitrary downstream queries such as the ones discussed in Sec 2. Our approach takes advantage of specific characteristics of spatio-temporal datasets to provide a customized mechanism that achieves DP.

3.1 VDR Overview

Our approach consists of three steps: (1) *data collection*, (2) *learned denoising* and (3) *statistical refinement*. We provide a brief overview of each step below, and present their complete technical descriptions in Sections 3.2 - 3.4.

Data Collection creates a noisy DP-compliant histogram of the data using the Laplace mechanism and a sampling technique that reduces sensitivity by bounding the amount of location reports from each individual. It makes no assumptions on the data distribution, and simply adds i.i.d random noise to each cell, with no modeling bias being introduced in the query answers to histogram cells.

Learned Denoising learns a model that captures the spatio-temporal density patterns of the original data from the noisy histogram. Using these patterns, denoising is able to construct a histogram that answers queries more accurately. It acts as a post-processing algorithm, hence the the DP requirement is preserved, and it makes use of variational auto-encoders to offset some of the excessive noise introduced by the Laplace mechanism.

Statistical Refinement scales the denoised histogram according to a DP-compliant statistical estimator to offset the effect of sampling used to decrease sensitivity during data collection.

3.2 Data Collection

Data collection uses a combination of sampling and noise addition to create a differentially private histogram of the data without making any modelling assumptions. In the case of spatio-temporal data, simple noise addition will lead to poor quality results, as the amount of noise needed will destroy any meaningful signal in the data. We first discuss the naive solution and its specific challenge for spatiotemporal data; subsequently, we show how sampling is

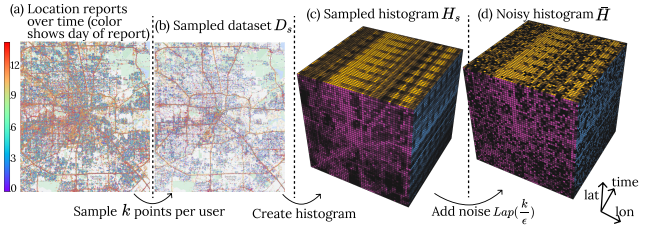

Figure 1: (a) and (b): real-world complete and sampled dataset of location reports over time in Houston. (c) and (d): exact and noisy 3-d histograms created from the sampled dataset, higher brightness shows higher density.

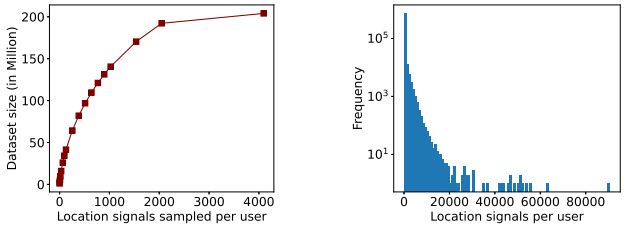

Figure 2: Veraset Houston Data Statistics

able to improve the accuracy; finally we present the details of the data collection mechanism. We use as running example a real world dataset of location reports from Houston, TX, USA (see Section 5 for exact details of dataset). This dataset is shown in Fig. 1 (a).

DP Histogram Release. Given dataset D with location reports from different users over time, the goal is to create a histogram of the data while preserving ε-DP. One way to do this (without making any modeling assumptions) is to first create the true histogram of the data H, and then add independent Laplace noise, $Lap(\frac{\Delta}{\varepsilon})$ to each cell of the histogram, where Δ is the sensitivity of the query of number of data points falling inside a cell. This sensitivity is equal to the maximum number of points, $k_{\max}$, a user contributes to the dataset. Thus, the DP histogram of the data can be written as $\bar{H} = H + Lap(\frac{k_{\max}}{\varepsilon})$, where independently generated random noise is added to each cell of the histogram.

Challenge for Spatiotemporal Histograms. In spatiotemporal datasets, the number of data points contributed to the dataset by each user varies significantly across users. While most users contribute few points, some prolific users may contribute a very large number of points. In fact, the number of points a user contributes often follows a power law [41, 63]. For our running example, Fig. 2 shows such dataset characteristics, where the maximum number of points contributed by a user is $k_{\max} = 90,676$ (in real datasets, location updates from mobile apps reporting user movements often leads to large number of contributions, with the amount of user contributions varying due to different app utilization across users). In this dataset, for ranges of 30 meters and 1 hour time periods, only 1 percent of the histogram cells have values more than 25. Thus, Laplace noise scaled to $k_{\max}/\varepsilon$, for any reasonable value of ε, wipes out any meaningful information in all but a few cells.

Sampling to Bound Sensitivity. Instead of using all the data points of users when creating the histogram, we sample a maximum of k points per user, for a user parameter $k < k_{\max}$. Specifically, we sample a subset of points $D_s \subseteq D$ as follows: for any user with more than k points, we sample k of their points uniformly at random.

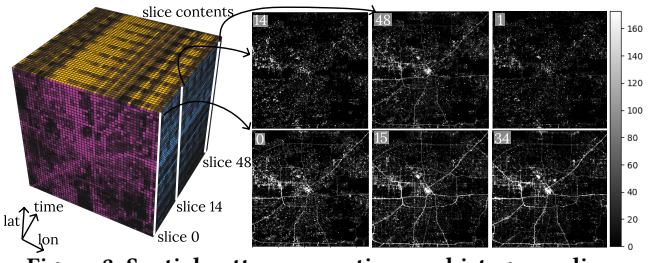

Figure 3: Spatial patterns over time on histogram slices

For users with at most k points, we keep all their points. This reduces the sensitivity of releasing histogram to k, requiring that we add only noise $Lap(\frac{k}{\epsilon})$ to each cell of the histogram. In this way, we can exploit the skewness in user contributions to the dataset, because by setting k to a small value, we are able to retain most of the original data. In our running example in Fig. 2 (left), setting k to 128 captures nearly 25% percent of the data, while reducing sensitivity by 700%. Consequently, we bias the data distribution in order to reduce variance in query reporting. Nonetheless, sampling introduces *sampling error* in the answers over histogram D_s. In our statistical refinement step (Sec. 3.4), we discuss how we can counter this source of error. We further discuss trade-offs arising in our method based on the choice of k in Sec. 4.

Summary and Example. Data collection is depicted in Fig. 1 for our running example, where we release a noisy 50×50×50 histogram of the dataset of location reports in Houston. We sample up to k for each user from the complete dataset D to create the sampled dataset D_s. Then, we create a 3-dimensional histogram, H_s of D_s. Finally, we create the histogram $\bar{H} = H_s + Lap(\frac{k}{\epsilon})$ so that the data collection process satisfies ϵ-DP. The output of the data collection step is the noisy histogram $\bar{H}$. The output in Fig. 1 (d) shows the noisy histogram, where each cell corresponds to noisy density in a 800×800 meter cell for a 4 hour time period.

3.3 Learned Denoising

Denoising uses machine learning, specifically VAEs, to identify spatial-temporal patterns, and utilizes them to improve histogram accuracy. Our main observations are: (1) spatio-temporal histograms are similar in nature to a sequence of images, thus methods from image representation learning can be applied to capture data patterns; (2) regularized representation learning can ensure the model learns a denoised representation of the data while not over-fitting noise; and (3) multi-resolution learning can capture spatio-temporal patterns at different granularities.

3.3.1 *Design Principles*

Denoising with Regularized Representation Learning. We want to derive a denoised histogram $\hat{H}$ from $\bar{H}$ that is similar to H, where similarity is measured as norm $\|H - \hat{H}\|$, i.e., the sum of squared differences across all cells of the histograms. To achieve this, consider a function `encoder`($\bar{H}$) that creates an encoding, z, of the noisy histogram, and a function `decoder`(z), that outputs a histogram, $\hat{H}$, from the encoding z. Consider the problem of learning an encoding z (i.e., by learning functions `encoder`(.) and `decoder`(.)), so that $\|\hat{H} - \bar{H}\|$ is minimized, where we call $\|\hat{H} - \bar{H}\|$ the *reconstruction error*. Our goal is to obtain an encoding z that summarizes the patterns in $\bar{H}$, since such patterns will also exist

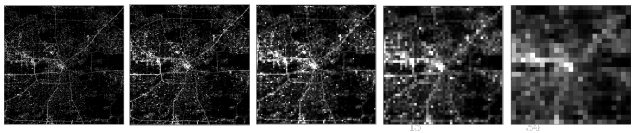

Figure 4: A histogram slice with varying coarseness

in the true histogram H. To see why this is possible, observe that a constraint on z limits its representation power. For instance, by setting the dimensionality of z to be lower than that of $\bar{H}$, z cannot contain as much information as $\bar{H}$. Thus, a regularized encoding, z, that minimizes the reconstruction error cannot contain all the information in $\bar{H}$. By learning a regularized representation, z, a model is able to capture the patterns in $\bar{H}$ that best summarize the histogram. Such a summary cannot include noise information, as it does not generalize across the histogram (noise is independently added to each cell) and can only be memorized individually per cell. Thus, by properly regularizing the encoding, we can find an encoding that is denoised, i.e., contains the patterns in the data, but not the noise. Subsequently, by decoding such a representation, we can obtain a denoised histogram. That is, the regularization ensures that even though we try to minimize the reconstruction error $\|\hat{H} - \bar{H}\|$, we obtain a histogram such that $\|\hat{H} - H\|$ is smaller than $\|\bar{H} - H\|$.

Spatial Patterns as Visual Patterns. Denoising with regularized representation learning will be beneficial only if the model is able to extract the patterns in the histogram. To facilitate this, we observe that a 3-d histogram of the data can be seen as a sequence of images, as shown in Fig. 3 for our running example. The left side of Fig. 3 plots the 3-d histogram which represents a time-series of two-dimensional histograms, one per each timestamp. We call each of these 2-d histograms a *slice*. On the right side of the figure, we plotted various slices, each corresponding to a different timestamp. We can see that spatial patterns in the histogram are in fact visual patterns. For instance, patterns corresponding to roads or busy areas can be seen as lines or blobs in the image. Furthermore, such patterns are consistent and repeating over time, suggesting that representation learning can be achieved effectively using techniques from image feature learning.

Multi-Resolution Learning at Varying Granularity. Spatial patterns in the data exist at various granularities of the input histogram. Fig. 4 illustrates several degrees of histogram resolution, from finest (left) to coarsest (right). We observe that patterns at finer resolutions feature roads more prominently, while patterns at coarser granularities feature primarily neighbourhood densities. Furthermore, the patterns in coarser granularity histograms are less affected by noise, which allows the model to still infer spatial density. Thus, we propose to train a single model based on data configured at multiple granularities to improve denoising accuracy.

3.3.2 *Denoising with Convolutional VAEs.*

Based on the above principles, we utilize convolutional VAEs to denoise the histogram $\bar{H}$. We first provide an overview of our algorithm, then provide more details on the role of regularization in our methodology, and finally present the algorithm pseudocode.

Our method consists of three stages: (1) training data preparation, (2) model training and (3) model inference. We discuss each below.

Training data preparation. Recall that we are given a noisy 3-d histogram, $\bar{H}$, where the 2-d histograms resulting from each

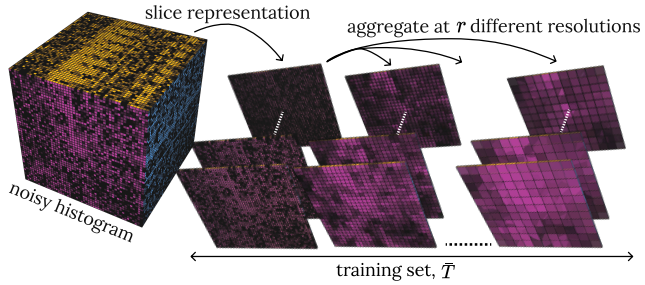

Figure 5: Training set preparation

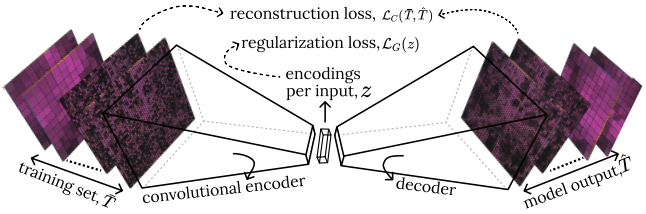

Figure 6: Model Training

slice contain density information for different locations. Thus, we view the 3-dimensional histogram $\bar{H}$ as a set of two dimensional histograms, where the i-th element in this set, $\bar{H}_i$, is a 2-dimensional slice corresponding to the i-th timestamp. This is shown on the left side of Fig. 5. Then, as shown on the right side of Fig. 5, to utilize multi-resolution learning, we aggregate each of the slices at various resolutions. For instance, every block of 2×2 cells in $\bar{H}_i$ are aggregated to obtain a new 2-d histogram which has a coarser granularity. This aggregation is done at r different resolutions, and all the aggregated histograms are put together in a training set T.

Model Training. We use a convolutional VAE to perform regularized representation learning. Specifically, an encoder, which is a CNN, takes as input each 2-d slice and outputs a representation for it. Denote by $\texttt{encoder}(.;\theta_e)$ the network whose parameters are θ_e, and let ℓ be the dimensionality of the representation output. The representation is then fed to a decoder, which is another neural network, denoted by $\texttt{decoder}(.;\theta_d)$, where θ_d are the parameters of the decoder. The output of the decoder is a 2-d histogram, as shown in Fig. 6. To simplify notation, we directly input a set of 2-d histograms to the encoder, in which case the output is also a set of representations (similarly for decoder). The model is trained to minimize a reconstruction loss, which is the difference between input slice and the output slice, and a regularization loss, which ensures that the learned representation follows some regularization constraints.

Model Inference. Each slice, $\bar{H}_i$, is passed through the convolutional VAE, first encoded and then decoded, to obtain the denoised representation for $\bar{H}_i$. This is done for all timestamps, i, which allows us to obtain a denoised 3-d histogram, $\hat{H}$. Note that, inference is not performed on any aggregated histogram (via multi-resolution), but only on the original noisy histogram $\bar{H}$. In other words, the output of learned denoising is a single 3-d histogram $\hat{H}$, which is at the same resolution as the noisy input histogram $\bar{H}$.

VAE and Regularization Details. We discuss parts of the VAE design relevant to the problem of denoising. Further details of our model can be found in Sec. 5. We utilize the Vector Quantized variant of VAE (VQ-VAE), where the encoding is forced to follow a certain discrete structure (other variants, e.g., Gaussian latent distribution [35] are also possible, but we found VQ-VAE to perform the best). A discrete set, Υ, called a *codebook*, of $\mathcal{B}$ different encodings, $\Upsilon = \{e_1, ..., e_{\mathcal{B}}\}$, where each e_i is ℓ-dimensional, is learned, and VAE training process forces the encoder to output an encoding that is similar to an element in the codebook. Recall that $\texttt{encoder}(.;\theta_e)$ is a convolutional neural network that takes as input a 2-d histogram. For an input 2-d histograms in our training set, $\bar{T}$, $\texttt{encoder}(\bar{T};\theta_e)$ provides a set of representations z. These representations are then input to the decoder to obtain reconstructions $\hat{T} = \texttt{decoder}(z;\theta_d)$. VQ-VAE defines a distance function between z and Υ, $d(z,\Upsilon)$, that measures how similar the encodings are to the codebook. $d(z,\Upsilon)$ is then minimized in the training process to ensure the encoder learns representations that are similar to the codebook. We call $\mathcal{L}_G(z) = d(z,\Upsilon)$ the regularization loss and define $\mathcal{L}_C(\hat{T}) = \sum_{i=1}^{|\bar{T}|} \|\bar{T}_i - \hat{T}_i\|^2$ as the reconstruction loss, where T_i is the i-th slice in the training set and $\hat{T}_i$ is the output of the VQ-VAE on the i-th training slice. We then train VQ-VAE to minimize $\alpha \times \mathcal{L}_G(z) + \mathcal{L}_C(\hat{T})$, where parameter α is introduced to control the amount of emphasis on the regularization.

Parameters $\mathcal{B}$ and α control how regularization benefits denoising. (1) $\mathcal{B}$ controls the representation power of the encoding space: the smaller $\mathcal{B}$ is, the less information can be captured by different encodings, as the encodings for different slices are forced to be similar. On the other hand, when $\mathcal{B}$ is large, different slices are allowed to take different representations, as the codebook allows for more variability. (2) α controls how much the encoder is forced to adhere to the codebook. When α is small, the encoder can learn representations that do not follow the discretized structure. It allows learning different encodings for different slices, thus memorizing the information within slices instead of learning patterns that generalize.

We empirically find significant benefit in invoking both of the above regularization aspects of VQ-VAEs. Specifically, we observed worse performance when setting α to a small value, emphasizing the need for regularizing the encoding space. We also observed worse performance when $\mathcal{B}$ is too small or too large, the former because not enough information can be stored in the learned encoding, and the latter because the encoding can become noisy (due to insufficient regularization). Furthermore, privacy budget affects the extent of regularization required. For instance, when privacy budget is large, noise of smaller magnitude is added and thus finer grained patterns are preserved in the data. By increasing $\mathcal{B}$ in such a scenario, the model can learn more detailed features of the data. On the other hand, for smaller privacy budgets, $\mathcal{B}$ needs to be small so that the model does not overfit the noise. We further investigate these trade-offs in Sec. 5.

Complete Denoising Algorithm. The complete denoising process is shown in Alg. 1. Lines 1-5 show how the training set is augmented with histograms at varying granularities. Lines 6-7 create a CNN as an encoder and a Transposed CNN as the decoder. The model is trained in Lines 8-16, where at Line 9 the encoder outputs encodings of the histograms in the training set and the encodings are then decoded by the decoder to reconstruct the histograms in Line 10. The model is then optimized with stochastic gradient descent to minimize the reconstruction loss and the regularization

loss. Finally, after convergence, a forward pass of the model yields the denoised histogram.

We have kept the discussion of convolutional VAEs at a high level and only provided details for ideas that pertain to the problem of denoising, without discussing in-depth the technical details in VAE and VQ-VAE design such as commitment and alignment loss [57], the reparametrization trick [61] and their relation to regularization. We provide implementation specific details of VQ-VAE in Sec. 5.

3.4 Statistical Refinement

Given that the values in the denoised histogram are based on the sampled dataset, they will be an underestimation of the true counts. In this section, we study how the values can be scaled to accurately represent the true counts. We first discuss how differential privacy complicates this process of statistical refinement, then present notations and assumptions in our method and finally discuss the statistical refinement step.

3.4.1 Estimation with Differential Privacy. Recall that we sampled a dataset D_s from the true dataset D, and created a noisy histogram $\bar{H}$ from the sampled set. We retained up to k points per user, hence the size of D_s is smaller than D. Thus, the number of data points that fall inside the histogram created based on D_s will be an underestimation of the true number of data points. To adjust the observed answers based on sampled data points we need to scale them, so that they accurately represent the true numbers. However, DP affects how this scaling can be done.

Noisy Observations. Scaling the values in $\bar{H}$ scales both the added noise and the observed values, thus amplifying the random noise. In other words, by scaling the observed values, we reduce the bias in our estimation (i.e., account for underestimation), but this scaling increases the variance in our estimation because the random noise gets amplified. Thus, in the case of sampling with differential privacy, it is important for our method to account for both bias and variance in the estimation.

Private Sampling Procedure. The sampling procedure is data dependent, and its specific details may be unknown, due to privacy requirements. Therefore, we aim to derive a refinement approach that is agnostic to the sampling performed during data collection. For instance, the probability of sampling a point in a particular cell does not only depend on the total number of points in that cell, but also on which user they belong to. If all users in a cell have exactly one point in D, then all the points in that cell will be preserved and thus the number of points in that cell in D_s will be the same as number of points in the corresponding cell in D. However, if users in a cell have more than k points, then the number of points in the cell in D_s is less than D. Due to differential privacy, information about the number of points per user in a cell can only be known by spending privacy budget, which is undesirable.

3.4.2 Estimation algorithm. Taking into account the above observations, we use mean square error minimization to decide how the answer should be scaled, which accounts for both bias and the variance, and thus ensures that if the noise is too severe it is not amplified. Moreover, rather than spending privacy budget to estimate the sampling procedure, we make simplifying assumptions to create a tractable sampling model that can be mathematically analyzed.

Algorithm 1 Learned Denoising

Require: A set of noisy 2-dimensional histograms, $\bar{H}$
Ensure: A set of denoised 2-dimensional histogram, $\hat{H}$

$\bar{T} \leftarrow \bar{H}$
for $j \leftarrow 2$ to r **do**
  for $\bar{H}_i$ in $\bar{H}$ **do**
    $\bar{H}_i^j \leftarrow$ Histogram from aggregating $j \times j$ blocks of $\bar{H}_i$
    $\bar{T} \leftarrow \bar{T} \cup \bar{H}_i^j$
`encoder`$(.;\theta_e) \leftarrow$ CNN encoder with params. θ_e
`decoder`$(.;\theta_d) \leftarrow$ TransposedCNN decoder with params. θ_d
repeat
  $z \leftarrow$ `encoder`$(\bar{T};\theta_e)$
  $\hat{T} \leftarrow$ `decoder`$(z;\theta_d)$
  $\mathcal{L}_C(\hat{T}) \leftarrow \sum_i \|\bar{T}_i - \hat{T}_i\|^2$
  $\mathcal{L}_G(z) \leftarrow d(z, \Upsilon)$
  $\mathcal{L} \leftarrow \alpha \times \mathcal{L}_G(z) + \mathcal{L}_C(\hat{T})$
  $\theta \leftarrow \theta_e \cup \theta_d$
  Update θ in direction $-\nabla_\theta \mathcal{L}$
until convergence
return `decoder(encoder`$(\bar{H};\theta_e)\theta_d$`)`

In the remainder of this section, we first describe our sampling model and then show how mean square error minimization can be used to decide how the observed noisy answers should be scaled to accurately represent the true data.

Notation and Modeling Assumptions. Let $N = |D|$ be the total number of data points and $n = |D_s|$ be the observed number of data points after sampling. We make simplifying assumptions about the sampling procedure for the purpose of our analysis. Specifically, we consider the case when the n points are sampled independently and uniformly at random. Let X_i^c be the indicator random variable equal to 1 if the i-th point falls in a cell c. Furthermore, let μ_c be the proportion of data points in the complete dataset that are in the cell c, so that $N \times \mu_c$ is the total number of data points in cell c. We assume that the i-th point is sampled uniformly at random across all data points, so that $\mathbb{P}(X_i^c = 1) = \mu_c$.

Algorithm. Our goal is to design an estimator to estimate $N \times \mu_c$ for all cells, c, in the histogram. Our estimator needs to be accurate, but at the same time has to preserve differential privacy. We consider the estimator $\theta_c = \gamma(\sum_i X_i^c + Lap(\frac{k}{\epsilon}))$. θ_c obtains a differentially private estimate of the observed number of points in the cell c and scales it by a parameter γ. We find the parameter γ by minimizing the mean squared error of our estimator θ_c. To do so we first calculate bias and variance of our estimator.

$$\text{Bias}(\theta_c) = \mathbb{E}[\theta_c - N\mu_c] = \mu_c(\gamma n - N)$$
$$\text{Var}(\theta_c) = \gamma^2(n\mu_c(1-\mu_c) + 2k^2\epsilon^{-2})$$

Thus given the mean squared error of an estimator, $\text{MSE}(\theta_c) = \text{Bias}(\theta_c)^2 + \text{Var}(\theta_c)$, we obtain

$$\text{MSE}(\theta_c) = \gamma^2(n\mu_c(1-\mu_c) + 2k^2\epsilon^{-2}) + \mu_c^2(\gamma n - N)^2.$$

Next, we find the γ value that minimizes error across all cells. Let $m = \mathbb{M} \times \mathbb{M} \times \mathbb{T}$ be the number of cells in the histogram. We minimize $\sum_{c=1}^m \text{MSE}(\theta_c)$ by taking the derivative of $\sum_{c=1}^m \text{MSE}(\theta_c)$ with

Algorithm 2 VDR algorithm

Require: A dataset D, privacy budget ϵ, spatial, $\mathbb{M}$, and temporal, $\mathbb{T}$, discretization granularity, sampling parameter k, and refinement factor C.

Ensure: Differentially private 3d-histogram $\hat{H}$ of D

1: $D_s \leftarrow$ sample k points per user in D
2: $H_s \leftarrow \mathbb{M}\times\mathbb{M}\times\mathbb{T}$ histogram of D_s
3: $\bar{H} \leftarrow H_s + Lap(\frac{k}{\epsilon})$
4: $\hat{H} \leftarrow \text{Denoise}(\bar{H})$
5: **return** $\gamma_C \times \hat{H}$

respect to γ and setting it to zero. We obtain that

$$\gamma = \frac{nNC}{2mk^2\epsilon^{-2} + (1-C)n + Cn^2} \tag{2}$$

minimizes $\sum_{c=1}^{m} \text{MSE}(\theta_c)$, where $C = \sum_{c=1}^{m} \mu_c^2$ is a data-dependent constant. It is left to determine the value of C, but doing so on the private data itself may require spending privacy budget. However, due to inherent properties of location datasets, in practice, C can be treated as a system parameter and set—in a data-independent manner thus not consuming valuable privacy budget—to a fixed value that works. We further discuss how C can be set in Sec. 4.3.

4 SYSTEM DESIGN AND ANALYSIS

4.1 Privacy Analysis

Alg. 2 shows our proposed end-to-end algorithm. Lines 1-3 correspond to the data collection step, line 4 calls Alg. 1 to perform learned denoising and line 5 uses the value of γ_C calculated in Eq. 2 to scale the results (we write it as γ_C to make explicit the dependence on the factor C). Alg. 2 only accesses the data in the data collection step. Thus, lines 4-5 can be seen as a post processing step and do not consume any privacy budget. Data collection is ϵ-DP, and thus the entire VDR provides ε-DP, as proved by the following:

Theorem 4.1. *Algorithm 2 is ϵ-DP*

The proof is in our extended technical report [12].

4.2 VDR Design Choices

Real world use-case of spatio-temporal data extends beyond simple range count queries that are commonly studied and optimized-for in common approaches to location data release, which are typically partitioning-based [25, 45, 45, 46, 64, 65, 67]. Other common query types such as forecasting POI visits, or finding hotspots, are sensitive to biases that such approaches introduce, causing them to perform poorly. VDR's approach of denoising a histogram created by Laplace mechanism (LPM) offers significant benefits across different spatio-temporal queries by avoiding such biases.

Forecasting Queries. Forecasting methods are often robust to random noise present in real data, some even explicitly incorporating its effects in their models. Thus, a DP mechanism that only introduces random noise, such as LPM, can perform well, whereas those that approximate the density of regions by cleverly grouping and partitioning the domain introduce additional bias and obliterate trends and seasonal effects present in the timeseries.

Hotspot Queries. For hotspots discovery, if a DP mechanism underestimates counts in the region of the hotspot, then it will receive a distance penalty due to not having found the correct spot, and may in addition incur a large *regret*, up to the maximum of the density threshold. This happens if an approach creates coarse partitions of the data, thus underestimating the density for 'hot' peaks. Selectively creating finer partitions can improve the result, since some 'hot' peaks may be preserved. Nonetheless, modelling errors in deciding where to create fine partitions can cause underestimation in some regions, resulting in large distance penalty. On the other hand, a bias-free approach such as LPM performs better since it is not affected by a systematic reduction in data utility that partitioning approaches incur.

Range Count Queries. Answering larger query ranges over LPM requires aggregating more histogram cells, each contributing additional error to the answer. Therefore, VDR is specifically designed to denoise (reduce variance) of a bias-free mechanism, smoothing out the noise by exploiting the inductive bias that spatial patterns exist in location datasets. In this way, it can improve forecasts significantly by preserving timeseries specific factors and discovering hotspots that likely meet the threshold, while not sacrificing the quality of results for range count queries.

4.3 System Parameter Selection

We discuss the impact of system parameters k, C and collection granularity on the performance of the system, and provide guidelines on how they should be set in practice.

4.3.1 Refinement Factor and Sampling Parameter. Recall that in our data collection step (Alg. 2 line 1), we sample up to k points per user and in the statistical refinement step we scale our result by a factor γ_C (Alg. 2 line 5), which depends on the refinement factor C. Both parameters, as discussed below, depend on data skewness as well as distribution of user contributions. However, due to DP, measuring data dependent properties requires spending privacy budget, which is scarce. Next, we discuss potential trade-offs in the values of these system parameters, and heuristics to set each.

Sampling parameter, k. For accuracy, the sampled dataset should retain density characteristics of the original dataset. After scaling with our statistical refinement step, the obtained query answers should be close to original counts. In our real-world datasets, the true data size, N, plays an important role in the interplay between true data characteristics and sampled ones. Specifically, when N increases, most cells in the true 3-d histogram, H, remain empty or retain small values, due to data sparsity, while the number of reported locations in dense cells increases. This results in a more skewed true dataset. Thus, for the sampled dataset D_s to capture this skewness, we need a larger number of samples, or otherwise our estimation will have a very large variance. We conclude that the value of k should grow with data size. Our experiments in Section 5.3 corroborate this heuristic, showing that the growth ratio λ, defined as $\frac{k^*}{N}$, where k^* is the best possible sampling rate, stays almost constant across datasets of various sizes. In fact, we observe that this value remains constant across different cities, suggesting that due to similarity in skewness, inherent to location datasets, we can set the value of k to be a constant fraction of N. Details of these observation are presented in Sec. 5.

Refinement Factor, C. Recall that C determines how query answers are scaled to obtain the final histogram. Our theoretical

model suggests that $C = \sum_c \mu_c^2$, but we do not have access to μ_c due to differential privacy, thus we treat C as a system parameter. Note that, C depends on data distribution. For instance, if the data are uniformly distributed, μ_c will be equal to $\frac{N/m}{N} = \frac{1}{m}$, so that $\sum_c \mu_c^2 = m \times \frac{1}{m^2} = \frac{1}{m}$. On the other hand, if all the N points are in a single cell, then $\sum_c \mu_c^2 = 1$. For location datasets, we expect similar skewness across the space, and as a result, similar values of C should perform well across datasets. Our results in Sec. 5.4 confirm that the same value of C can be used with distinct datasets and sampling rates. We suggest setting the value of C to one that performs well on a public dataset. Having C as a system parameter is advantageous because it allows for correction of errors that have been introduced due to our theoretical modeling. For instance, our analysis in Sec. 3.4.2 does not take into account the impact of denoising, which we expect to be consistent across datasets. By setting C as a system parameter, we can avoid any adverse impacts of modeling errors in practice. Moreover, since the modelling error is consistent across datasets, the same value of C can be set for all datasets.

4.3.2 Discretization Granularity. We assume throughout our discussion that a 3-d histogram of a predefined granularity is required. This is often the case since the choice of discretization tends to be domain specific. In the case of location datasets, high-resolution density maps are preferred [5], and VDR is the first attempt at releasing high resolution spatio-temporal density counts, where we choose to release counts at the same granularity as done by industry data release projects, e.g., [5]. This has been typically challenging since publicly available location datasets are small, and thus not suitable for studying at high resolution. The primary concern is that a fine grained histogram will have small true count values per cell, and since scale of DP-added noise is proportional to sensitivity and not actual data counts, the resulting signal-to-noise ratio will be low, compromising accuracy. On the other hand, if a coarse discretization is used, it induces an implicit bias in the results, since fine resolution queries need to estimate answers by assuming uniform distribution of points within the coarse cell, and will thus be inaccurate.

4.4 Data Release over Time

So far we have considered the release of a static dataset D. In practice, spatiotemporal data is released over time, with new data coming in continuously. In such a setting, privacy budget is often allocated per time period, e.g., a budget of ε_i would be allocated for the i-th week (ε_i is typical set to go to zero so that $\sum_i \varepsilon_i$ is bounded). Thus, the release consists of a sequence of datasets $D_1, D_2, \ldots$, where each D_i covers a fixed period of time. Let τ denote the duration covered by each D_i, which we call release duration. To use VDR in this setting, Alg. 2, can be called for every release, where in the i-th release, the input dataset is D_i and privacy budget is ϵ_i. However, an important characteristic of VDR is that the model does not need to be retrained for every data release. That is, rather than retraining the model in the learned denoising step for every release, after the model is trained once, it is still able to denoise the input histograms. We verify this empirically in our experiments. This also shows that our model is learning recurring patterns from data, which generalize well to unseen data points.

5 EXPERIMENTAL EVALUATION

Sec. 5.1 describes the experimental testbed. Sec. 5.2 compares VDR with state-of-the-art approaches. Sec. 5.3 ablates design choices. Sec. 5.4 presents a case-study of a user-level density release of spatiotemporal data, and the effect of statistical refinement on answer quality. Sec. 5.5 showcases VDR's effectiveness for non-uniform datasets. Sec. B of our technical report [12] contains supplemental experimental results.

5.1 Experimental Settings

Dataset Description. All datasets comprise of user check-ins specified as tuples of: user identifier, latitude and longitude of check-in location, and timestamp. Our primary dataset is proprietary, obtained from Veraset [3] (VS), a data-as-a-service company that collects anonymized movement data from 10% of the cellphones in the U.S [4]. For a single day in Jan 2020, there were 2.4 billion readings from 27.2 million distinct users. Where applicable, we also present our results on public datasets. Public datasets typically contain sporadic check-ins made over a relatively long period of time, as opposed to real longitudinal trajectories of users that the proprietary dataset offers. Our first public dataset from the Foursquare geo-social network (4SQ) [62] is collected during a time period of 22 months from April 2012 to 2014. There are 22M checkins by 114k users at 3.8 M POIs. Our second public dataset is a subset of the user check-ins in the US collected from the *Gowalla* (GW) network by the SNAP project [19]. It contains 6.4 million records from 196k unique users during a time period between February 2009 and October 2010. Our third public dataset is the San Francisco taxi dataset (CABS_SF) [44] derived from the GPS coordinates of approximately 500 taxis collected over 24 days in May 2008.

To simulate a realistic environment of a city and its suburbs, we consider urban areas in the US covering 20km^2. For the Veraset data, we select cities based on their population density [2]. In particular we have Salt Lake (VS_SL) city in Utah as a *low density* city (41M points from 600k users), Los Angeles (VS_LA) in California as *medium density* city (80M points from 852k users), and Houston (VS_HT) in Texas as high density city (221M points from 1.28M users). For all primary datasets, we discretize the temporal domain to 3 hours, giving a total of $\mathbb{T} = 96$ slices for the 12 day period from Jan 7 to 19, 2020. For each of the secondary datasets, we discretize the temporal dimension such that each slice covers the duration of one month for a total of $\mathbb{T} = 24$ slices. From the Foursquare dataset we consider Tokyo, Japan (4SQ_TKY). The subset contains a total of 755k location updates from 8k unique users. From the Gowalla dataset we select the San Francisco (GW_SF) city with 568k location updates from 14k users and New York (GW_NY) city with 520k location updates from 16k users. For the CABS dataset, following [27, 45], we keep only the start and end points of the mobility traces of the 500 taxis, for a total of 846k records.

Parameter Settings. Following recommendations in the industry [5] and research literature [38, 66], we partition the space into a 576×576 (i.e. $\mathbb{M} = 576$) grid to obtain $30m \times 30m$ cells. As described above, the temporal granularity is set specific to each dataset. The default value of privacy budget ε is set to 0.2.

Query Evaluation Metrics. Evaluation metrics are defined in Sec. 2. To evaluate the range count query metric, we construct

query sets of 5,000 RCQs centered at cells of randomly selected data records. Each query has side length that varies uniformly from 30 meters to 120 meters. To set the smoothing factor ψ we extend the recommended value of 0.1% of the cardinality n of the spatial dataset for snapshot queries [20, 45, 66] to 0.1% of the cardinality $n/\mathbb{T}$ of the average slice of the spatio-temporal dataset. When comparing multiple datasets with each other the smallest smoothing constant among them is used to remain consistent.

We evaluate the forecasting query only on subsets of the Veraset data, since other datasets do not contain timeseries of sufficient length. We generate 100 forecastable timeseries by sampling positions of POIs in the city, extracting count data of between 8-12 days (random lengths), selecting only those series that satisfy a Autocorrelation Seasonality test[39] (90% confidence) at seasonal period of 8 (meaning a daily seasonality according to 3 hour temporal discretization of the 24 hour period) To make forecasts, we use the winning algorithm of the M3 forecasting competition [39], Theta [13], which is a variant of the Simple Exponential Smoother. We use the data of all-but-last day to fit the forecaster and evaluate its predictions for last day (i.e., a forecasting horizon of $h = 8$). We report the sMAPE error as defined in Sec. 2.

Lastly, the hotspot query is evaluated at a specified threshold ν. The query, originating from the cells of 1000 randomly selected data points, is answered using an expanding search within the 3-d spatial region SR with lengths 5km in lat and lon and no bounds in time dimensions.

Implementation. All algorithms were implemented in Python, and executed on Linux machines with Intel i9-9980XE CPU, 128GB RAM and a RTX3090 GPU. Neural networks are implemented in JAX [15]. Given this setup, VDR took up to 50 minutes to train for 12 days of the Veraset Houston data. The inference time of VDR is less than 1ms and the model takes 9 MB of space. We publicly release the source code at [11].

Model Training. For Multi-Resolution Learning we augment the training set at $r = 3$ granularities chosen at equal spacing between the minimum (30m) and maximum (120m) query ranges to be evaluated. To train the model, we utilize the Adam [34] optimizer with Exponential Moving Average updates [49]. The encoder and decoder architecture is based on a ResNet[53]. The model takes in batches, the slices of the histogram, each of size 576×576, passes it through the ConvNet encoder θ_e and decoder θ_d. The EMA version trains much faster than the non-EMA version, especially when using Multi-Resolution Learning. In all our experiments, we utilize hyperparameters consistent with those utilized in previous work [49, 56]; i.e., $\ell = 64$, $\mathcal{B} = 128$ and a batch size of $b = 8$.

5.2 Comparison with Baselines

Baselines. We use as benchmarks Uniform Grid (UG) [45], Adaptive Grid (AG) [45], HB_striped [46, 64], PrivBayes [65], AHP [67] and MWEM [25]. Brief summaries of each method are provided in Sec. 6. We utilize Ektelo [26, 64], an operator-based framework for implementing privacy algorithms. To extend approaches designed to originally support range queries in two-dimensional data (spatial-only) to the 3-d case, we partition the temporal domain into parallel 'stripes/slices' of that domain for each fixed value of the rest of dimensions, so that the measurements are essentially the 2D histograms resulting from each slice. For example, HB_striped[64] performs on each slice the HB algorithm [46], which builds an optimized hierarchical set of queries. We similarly implement Uniform Grid (UG)[45] and Adaptive Grid (AG)[45]. We use as-is algorithms that are designed to extend to the multi-dimensional setting such as PrivBayes [65], AHP [67] and MWEM [25]. Lastly, we also considered QuadTree [20], but the results were far worse than the above approaches and thus are not reported. We were unable to run the DAWA [38] algorithm directly on such a large domain due to memory and computational constraints. DAWA is designed for 1D-inputs and extended to at best the 2-d setting using a Hilbert Curve based domain reduction.

Privacy Model. Since none of the existing baselines consider user-level privacy, to allow for a fair comparison with the baselines, we disable the user-level features of VDR and present experiments with event-level privacy. That is, for the results in this section, VDR does not perform sampling or statistical refinement and we assume each data record belongs to a separate user, hence the privacy protection offered degrades to event-level. We also note that since the public datasets have very limited number of points (orders of magnitude fewer than Veraset), user-level privacy evaluation on them is not feasible. Thus, presenting event-level results here also allows for reproducibility of our results on public datasets. We evaluate thoroughly the application of user-level privacy using Veraset data in Sec. 5.4.

5.2.1 Range Count Query. Figure 7 presents the error of VDR and the compared approaches when varying ε. Recall that a smaller ε means stronger privacy protection (hence larger noise magnitude). Unsurprisingly, the error reduces as noise magnitude decreases, with VDR consistently outperforming all competitor approaches. This shows that VDR is effective in capturing spatial patterns present in real-world locations datasets and smoothing excessive noise by exploiting such information. We also evaluate the impact of query size on accuracy of range count queries by considering test queries of three different sizes for datasets GW_NY and VS_HT. Figures 8(a) and 8(b) show that the error for all the algorithms increases when query size grows. Answering larger query ranges requires aggregating more grid cells, each contributing additional error to the answer.

5.2.2 Forecasting Query. Figs. 9(a)-(c) show results on Veraset data subsets VS_HT, VS_LA and VS_SL. Due to noise, all DP mechanisms produce worse forecasts (compared to 'No Noise'), as a direct result of poor fitting of the Theta forecasting algorithm. DP mechanisms that only introduce random noise, such as LPM, can perform well, whereas those that infer the density of regions by cleverly grouping and partitioning the domain introduce a bias and obliterate trends and seasonal effects present in the timeseries. VDR, with its ability to smooth out the noise, improves significantly these forecasts by preserving timeseries specific factors. Lastly, in some instances, such as for $\varepsilon = 0.05$ of Fig 9(c), UG can perform well by learning an average value for the timeseries period and making *naive* forecasts that predict the last period's actuals as next period's value, without establishing causal factors.

5.2.3 Hotspot Query. Fig.10 reports the accuracy for hotspot queries on various Veraset subsets at the fixed density threshold of $\nu = 20$. We report the results for varying thresholds in Sec. B of our technical report [12]. Mechanisms that model directly the data distribution

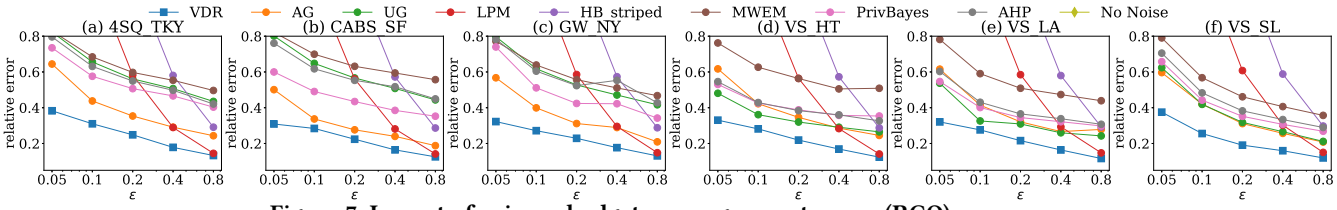

Figure 7: Impact of privacy budget on range count query (RCQ) accuracy.

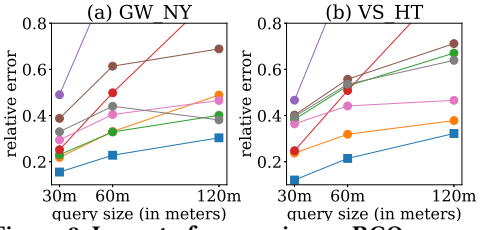

Figure 8: Impact of query size on RCQ accuracy.

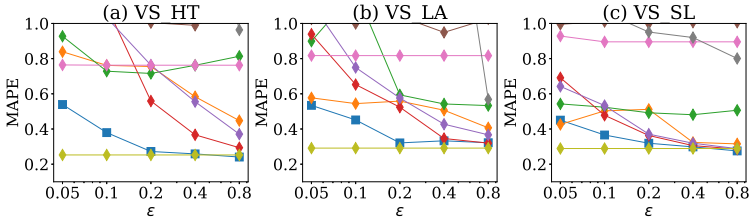

Figure 9: Impact of privacy budget on forecast query error (sMAPE)

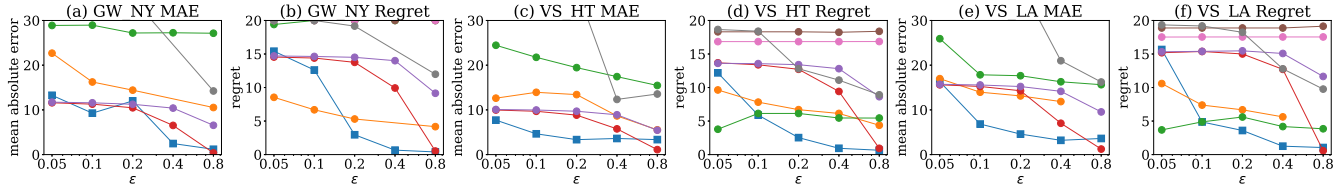

Figure 10: Impact of privacy budget on hotspot query accuracy.

such as MWEM, AHP and PrivBayes tend to underestimate density globally, and incur a large MAE and regret penalty, up to the maximum of the density threshold. To the same effect, UG, due to its coarse partitioning of the data domain, underestimates the 'hot' peaks that the query searches for, also experiencing both a large MAE and regret. AG improves these estimates to some extent by building a finer domain partitioning in the lower level of its hierarchy, and while it may not locate the closest hotspot (high MAE), it still finds one that meets the density threshold (lower regret). LPM fares well for this query as it is not affected by the biases that partitioning approaches bring about. VDR further improves on LPM in both metrics. In all instances, VDR finds an effective balance between the MAE and regret error metrics.

Our results show VDR clearly outperforms the existing state-of-the-art mechanisms. In particular, by starting with an unbiased estimate of the density counts and denoising them, our approach has clear advantages when used for answering range count queries, finding hotspots or forecasting POI visit counts. In the rest of this section, we no longer consider competitor approaches, and we focus solely on analyzing the parameters of our system.

5.3 System Analysis

5.3.1 Modeling choices

Effect of model regularization on performance. We evaluate the effectiveness of *Variational-AutoEncoder* in denoising DP histograms. Recall that, by training a lower dimension representation of the data, we wish to learn patterns without overfitting to the noise. In Fig. 14 we evaluate two methods: 'VDR' is our approach of using a VAE as a regularization-enabled AutoEncoder, while 'R. focus' simulates an AutoEncoder by over-emphasizing reconstruction loss (i.e. by setting α to a very small value). We consider both public and proprietary datasets while varying the bottleneck size. We notice that a small bottleneck performs poorly due to having limited representation power to learn the input data. When increasing the bottleneck, we see polar effects in the presence and absence of regularization. In the case of 'R. Focus', the model quickly overfits to the noise, decreasing accuracy. Whereas if the AutoEncoder is sufficiently regularized, accuracy remains good even for large models due to the learning of generalizable patterns, emphasizing the need for regularizing the encoding space.

Effect of learning period. Fig. 15 shows the accuracy of denoising when we train the VAE with a varying number of slices. When the number of learning slices is one, we have in essence a snapshot dataset in 2D. As we add more slices to training, the learning is stabilized and the the learned patterns help achieve better denoising performance in the entire dataset.

Effect of Multi Resolution Learning on accuracy. Fig. 13 shows that across all datasets, by augmenting the training set with coarser granularity histograms, we learn a model that can answer queries more accurately. This is also due to the smoothing effect of the ConvNet, as learned information from one slice helps denoise another within the same dataset.

5.3.2 Data release over time. We study the effectiveness of VDR when releasing data over time. Specifically, we measure how often VDR needs to be retrained when new data arrive. For this experiment we utilize the Veraset Houston data for a period of 24 days, with each slice representing a one hour time period. We consider a training period, T_b to T_e, where T_b is the beginning of the training period and T_e is the end, and a testing period that starts at T_e and ends at T_t. We refer to the period T_b to T_e as in-sample and T_e to T_t as out-of-sample. We evaluate the performance of the model in two scenarios. In Figure 11, we test the denoising performance of VDR on an out-of-sample period of 3.5 days (84 slices). VDR in-sample and out-of-sample show the accuracy of VDR on the

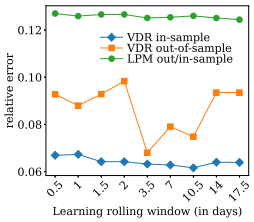

Figure 11: Impact of varying learning period

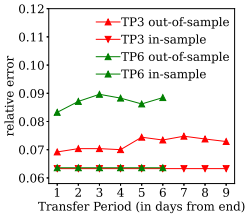

Figure 12: Out-of-sample denoising.

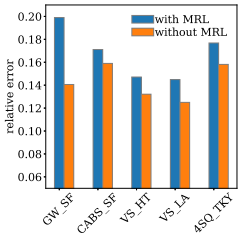

Figure 13: Multi Resolution Learning

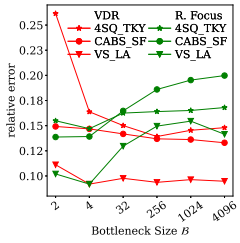

Figure 14: Model regularization analysis

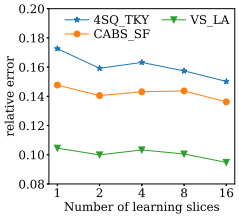

Figure 15: Learning Period Analysis

in-sample and out-of-sample period, respectively. The training data period is varied by moving T_b forward in time but keeping T_e and T_t the same (so training period ranges from 420 to 12 slices). We see that the performance first improves when the training period is up to 3.5 days, as having more data helps the model denoise via better generalizabilty. But as even more data is used in the training, specificity of the patterns is reduced, hence accuracy suffers. In the second setting (Fig. 12), we train VDR over slices of 'TP' (*Training period*) number of days, and use the trained model to denoise. For in-sample testing the *Transfer Period* refers to the accuracy of VDR on the training data itself. We see that the model accuracy is mostly unaffected. For out-of-sample testing *Transfer* period refers to the number of days from T_e to T_t. We increase T_t so that transfer period ranges from 1 day to 9 days. We see that performance degrades when denoising out-of-sample periods far from the training periods. We recommend retraining of the VAE model every couple of days.

5.4 User-level privacy and statistical refinement

We consider several Veraset data subsets, and set $\varepsilon = 6$, analogous to [10, 17]. Recall that, in order to release data without consuming excessive privacy budget, we bound the maximum number of contributions of a user to k, consequently having access to only the sampled subset D_s to learn our model. Note that in Figs. 16, 17, 18 and 19 the relative error metric is evaluated w.r.t to the true data D, while the query computations are performed over D_s.

Bounding user contribution. In Fig 16, we evaluate the accuracy of answering range count queries when varying the sampling rate k. Sampling error (denoted as SE) measures the error induced purely due to bounding the contribution of each user to k. As expected, SE decreases as the sampled subset D_s comes closer to representing the true dataset D (also refer to Fig. 2). We also report the error in the DP-preserving answer using LPM. Denoted by $SE + NE$ is LPM which contains errors induced to DP-compliant noise in addition to those due to sampling. As k increases, the benefits from increased sampling dominate the total effect (see $k = 2$ to 16), however the sensitivity of the query (hence, the noise) eventually exceeds the rate at which sampling error decreases, causing a sharp increase in the error of the reported answer (see $k = 32$ to 256). We discuss how to set the ideal value of k at the end of this section.

Analysing the effects of brute-force debiasing. Recall that the answer to the range count query reported on D_s can be scaled according to N/n to potentially debias the result. However, since the data is skewed and with added noise, such a scaling can affect the results negatively. We vary the degree of scaling as $g \times N/n$ for values of g from 0.1 to 1. Figure 17 shows that for sampling induced error SE, scaling the answer can be useful. However, if there is in addition noise error SE+NE, then upscaling also amplifies the noise in the reported counts and almost always yields poor outputs. Therefore, to utilize any form of scaling it is important to capture accurately the model of the data, such as by denoising with VDR.

Statistical refinement. The refinement constant C determines the degree of scaling α that is applied to the query answer. For example, at $C = 1$, γ approaches N/n, equivalent to a basic scaling. In Figure 18, we evaluate the accuracy of reporting for VDR while varying C at various degrees of sampling k. Remarkably, among all settings the lowest error is achieved at C = 5e-5, substantiating our claim that a fixed value of C is sufficient to refine answers. In Fig. 19 we compare the accuracy of VDR (at C = 5e-5) with the approach that reports the answer computed on D_s as-is. In all datasets, there is a clear benefit to using statistical refinement, improving the error by up to 40% in the case of VS_HT, a high density city.

We conclude with recommendations on how to set k, continuing our discussion from Sec. 4.3. We determine the value of growth ratio λ empirically from the data of another city (assumed public for the sake of privacy accounting), where we find λ = 2.5e-7 to give the best accuracy. For this value our heuristic ($k = \lambda N$) recommends setting $k = 10$ for VS_SL, $k = 20$ for VS_LA and $k = 53$ for VS_HT. As we see in Fig. 19, these values of k achieve close to the best accuracy for their corresponding cities. This suggests that due to similarity in skewness, inherent to location datasets, we can set the value of k to be a constant fraction of N.

5.5 Learning Ability on Non-Uniform Datasets

Setup. We synthesize $2M$ points from a Gaussian Mixture Model (GMM) [50] with 50 components positioned at random in the 3D integer lattice $\mathbb{Z}^3$ of size 9×9×9. All components are equally weighted and have the covariance matrix $\vec{I}\times\sigma^2$, where $\vec{I}$ is the identity matrix. To control the variation around its mean value, we adjust the parameter σ. The synthesized data is partitioned into a 3D histogram of 100×100×100 cells. We train and denoise with VDR on the 100 slices. We report σ in terms of the number of such cells, with a smaller variance implying a data spread tighter around the mean of each GMM component, thus mimicking the skewed data distributions typically present in spatio-temporal location datasets. Fig. 20 ($\sigma = 3$) and Fig. 21 ($\sigma = 7$) visualize a single slice of this dataset with its true values (left), noisy data collections (middle) and denoised reconstructions (right). VDR has a strong ability to recover the underlying patterns of GMMs from even highly distorted observations. Moreover, Figure 22 ($\varepsilon = 0.2$) shows that as we increase the variance σ^2 of the GMM components, the model performance suffers, since at a large variance (such as the one depicted in Fig. 21) data is more uniformly distributed and lacks the spatial

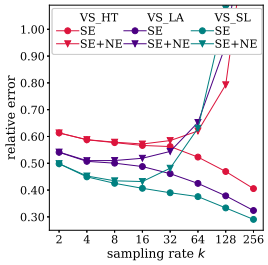

Figure 16: Sampling Error and and Noise Error at varying sampling rate.

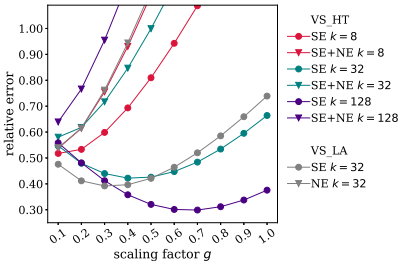

Figure 17: Effect of Scaling factor g on SE and SE+NE

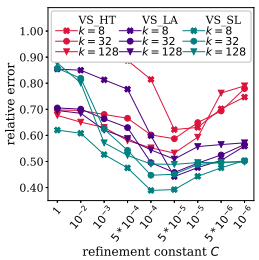

Figure 18: Varying refinement constant C for VDR.

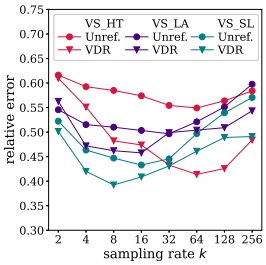

Figure 19: Benefit of VDR refinement with C = 5e-5.

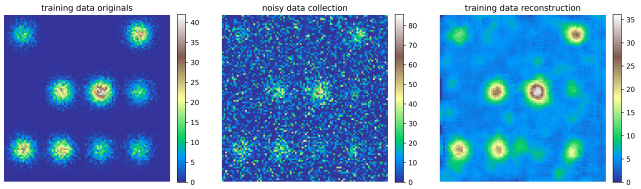

Figure 20: Learning behavior at GMM $\sigma = 3$

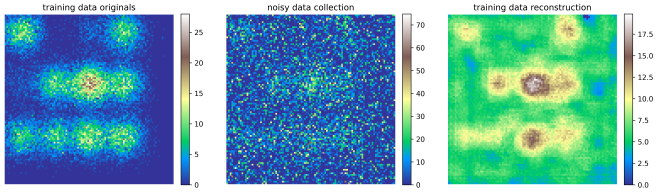

Figure 21: Learning behavior at $\sigma = 7$

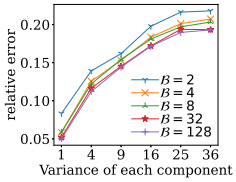

Figure 22: Varying σ^2

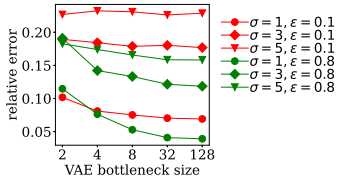

Figure 23: Varying bottleneck.

patterns typically exhibited in location datasets. Lastly, we evaluate the effect of varying the bottleneck size of the VAE on the learning ability of VDR. Figure 23 shows that, for a given privacy budget, a larger bottleneck is required to capture more skewed datasets. Moreover, when the data is skewed (compare lines for $\varepsilon = 0.1$ and $\varepsilon = 0.8$ at $\sigma = 1$), less DP noise in the data collection step helps further emphasize the data spread, thus benefiting from having a larger model capacity to learn precisely such patterns.

6 RELATED WORK

Private Data Release. Longitudinal release of individual location updates increase risk of attack [16, 59], and requires more stringent privacy settings, e.g., user-level privacy. The work in [8] models disjoint regions of the space as separate 1-d time series. However, this limits supported query types, and cannot answer range or hotspot queries. Moreover, the granularity used is very coarse. PrivBayes [65] is a mechanism that privately learns a Bayesian network over the data, and then returns a matrix used for fitting the parameters of the Bayes net. This can be used to then generate a synthetic dataset which can consistently answer workload queries. Budget allocation is equally split between learning the Bayesian network structure and learning its parameters. Multiplicative-Weights Exponential Mechanism (MWEM) [25] maintains an approximating distribution over the data domain, scaled by the number of records. It updates this distribution by posing a workload of linear queries (e.g., RCQs), finding poorly answered ones, and using the multiplicative update rule to revise its estimates. AHP [67] seeks to group a histogram's adjacent bins with close counts to trade for smaller noise. It utilizes LPM, and sets noisy counts below a threshold to zero. The counts are then sorted and clustered using a global clustering scheme to form a partition.

Noise reduction techniques. Most deep-learning based denoising methods [31, 37, 43] rely on many pairs of clean/noisy images. Denoising autoencoders attempt to learn original data distributions that have been corrupted according to some noise distribution, (e.g., by maximizing the log probability of the clean input, given a noisy input). Recent work in [36] trains a model from noisy/noisy image pairs, by extracting noisy versions of the same image repeatedly. Such a training process is not viable under DP since it would require additional privacy budget for each noisy extraction. Some mild noise from images can also be removed in an unsupervised fashion [47, 68]. No approach studied denoising in the presence of DP.

Privacy preserving machine learning. A learned model may leak information about the data it was trained on [29, 52]. Application of DP to empirical risk minimization [18, 33] and deep neural networks [6, 51] has been recently explored. Existing approaches add noise to the output of the trained model [58], add a random regularization term to the objective function [18, 33], or add noise to the gradient of the loss function during training [6]. Our approach sanitizes the training data *before* learning. Furthermore, the work of [6] achieves (ε, δ)-DP [7, 23, 42], a weaker privacy guarantee.

7 CONCLUSION

We proposed a technique for accurate DP-compliant release of spatio-temporal histograms that uses a combination of sampling to reduce sensitivity, VAE-based learning to counter the effect of DP-added noise, and statistical estimators to offset the effect of sampling. The resulting approach captures well spatio-temporal data patterns, and significantly outperforms existing approaches. In future work, we plan to extend our work by creating DP-compliant synthetic datasets based on spatio-temporal histograms. This is more challenging, since it needs to take into account any type of downstream processing that may be performed. One direction to achieve this goal is to sample from the compressed latent space conditioned on the time-of-day, and train a conditional image generation model such as PixelCNN [55] over the latent pixel values.

REFERENCES

[1] [n.d.]. Safegraph Weekly Patterns. https://docs.safegraph.com/docs/weekly-patterns. Accessed: 2022-04-11.

[2] 2021. List of United States cities by population. https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population. Accessed July 2021.

[3] 2021. Veraset. https://www.veraset.com/about-veraset. Accessed: 2021-05-10.

[4] 2021. Veraset Movement Data for the OCONUS. https://datarade.ai/data-products/veraset-movement-data-for-the-oconus-the-largest-deepest-and-broadest-available-movement-dataset-veraset. Accessed: 2021-07-20.

[5] 2022. *Facebook Data for Good: High Resolution Density Maps.* https://dataforgood.facebook.com/dfg/tools/high-resolution-population-density-maps

[6] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[7] John M. Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery, Data Mining* (London, United Kingdom) *(KDD '18)*. 2867.

[8] Gergely Acs and Claude Castelluccia. 2014. A case study: Privacy preserving release of spatio-temporal density in paris. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1679–1688.

[9] Gergely Acs, Claude Castelluccia, and Rui Chen. 2012. Differentially private histogram publishing through lossy compression. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 1–10.

[10] Ahmet Aktay, Shailesh Bavadekar, Gwen Cossoul, John Davis, Damien Desfontaines, Alex Fabrikant, Evgeniy Gabrilovich, Krishna Gadepalli, Bryant Gipson, Miguel Guevara, et al. 2020. Google COVID-19 community mobility reports: anonymization process description (version 1.1). *arXiv preprint arXiv:2004.04145* (2020).

[11] Anonymous. 2022. VDR Implementation. https://anonymous.4open.science/r/paper_code-C76E.

[12] Anonymous. 2022. VDR Technical Report. https://anonymous.4open.science/r/paper_code-C76E/vdr_technical_report.pdf.

[13] Vassilis Assimakopoulos and Konstantinos Nikolopoulos. 2000. The theta model: a decomposition approach to forecasting. *International journal of forecasting* 16, 4 (2000), 521–530.

[14] Aleix Bassolas, Hugo Barbosa-Filho, Brian Dickinson, Xerxes Dotiwalla, Paul Eastham, Riccardo Gallotti, Gourab Ghoshal, Bryant Gipson, Surendra A Hazarie, Henry Kautz, et al. 2019. Hierarchical organization of urban mobility and its connection with city livability. *Nature communications* 10, 1 (2019), 1–10.

[15] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs. http://github.com/google/jax

[16] Yang Cao, Masatoshi Yoshikawa, Yonghui Xiao, and Li Xiong. 2017. Quantifying differential privacy under temporal correlations. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 821–832.

[17] Serina Chang, Emma Pierson, Pang Wei Koh, Jaline Gerardin, Beth Redbird, David Grusky, and Jure Leskovec. 2021. Mobility network models of COVID-19 explain inequities and inform reopening. *Nature* 589, 7840 (2021), 82–87.

[18] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research* 12, 3 (2011).

[19] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1082–1090.

[20] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. 2012. Differentially private spatial decompositions. In *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 20–31.

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

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

[23] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[24] Masoomali Fatehkia, Benjamin Coles, Ferda Ofli, and Ingmar Weber. 2020. The relative value of facebook advertising data for poverty mapping. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 14. 934–938.

[25] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A simple and practical algorithm for differentially private data release. *Advances in neural information processing systems* 25 (2012).

[26] Michael Hay. 2022. Ektelo. https://github.com/ektelo/ektelo.

[27] Michael Hay, Ashwin Machanavajjhala, Gerome Miklau, Yan Chen, and Dan Zhang. 2016. Principled evaluation of differentially private algorithms using dpbench. In *Proceedings of the 2016 International Conference on Management of Data*. 139–154.

[28] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2009. Boosting the accuracy of differentially-private histograms through consistency. *arXiv preprint arXiv:0904.0942* (2009).

[29] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 603–618.

[30] Florimond Houssiau, Luc Rocher, and Yves-Alexandre de Montjoye. 2022. On the difficulty of achieving Differential Privacy in practice: user-level guarantees in aggregate location data. *Nature communications* 13, 1 (2022), 1–3.

[31] Daniel Im Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. 2017. Denoising criterion for variational auto-encoding framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[32] Georgios Kellaris and Stavros Papadopoulos. 2013. Practical differential privacy via grouping and smoothing. *Proceedings of the VLDB Endowment* 6, 5 (2013), 301–312.

[33] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. 2012. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings, 25–1.

[34] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[35] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[36] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. 2019. Noise2void-learning denoising from single noisy images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2129–2137.

[37] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. 2018. Noise2Noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189* (2018).

[38] Chao Li, Michael Hay, Gerome Miklau, and Yue Wang. 2014. A Data- and Workload-Aware Algorithm for Range Queries under Differential Privacy. *Proc. VLDB Endow.* 7, 5 (Jan. 2014), 341–352.

[39] Spyros Makridakis and Michele Hibon. 2000. The M3-Competition: results, conclusions and implications. *International journal of forecasting* 16, 4 (2000), 451–476.

[40] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2018. Optimizing error of high-dimensional statistical queries under differential privacy. *arXiv preprint arXiv:1808.03537* (2018).

[41] Lev Muchnik, Sen Pei, Lucas C Parra, Saulo DS Reis, José S Andrade Jr, Shlomo Havlin, and Hernán A Makse. 2013. Origins of power-law degree distribution in the heterogeneity of human activity in social networks. *Scientific reports* 3, 1 (2013), 1–8.

[42] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 75–84.

[43] Tongyao Pang, Huan Zheng, Yuhui Quan, and Hui Ji. 2021. Recorrupted-to-Recorrupted: Unsupervised Deep Learning for Image Denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2043–2052.

[44] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. 2009. CRAWDAD data set epfl/mobility (v. 2009-02-24).

[45] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Differentially private grids for geospatial data. In *2013 IEEE 29th international conference on data engineering (ICDE)*. IEEE, 757–768.

[46] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1954–1965.

[47] Yuhui Quan, Mingqin Chen, Tongyao Pang, and Hui Ji. 2020. Self2self with dropout: Learning self-supervised denoising from single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1890–1898.

[48] Sirisha Rambhatla, Sepanta Zeighami, Kameron Shahabi, Cyrus Shahabi, and Yan Liu. 2022. Toward Accurate Spatiotemporal COVID-19 Risk Scores Using High-Resolution Real-World Mobility Data. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 8, 2 (2022), 1–30.

[49] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems* 32 (2019).

[50] Douglas A Reynolds. 2009. Gaussian mixture models. *Encyclopedia of biometrics* 741 (2009), 659–663.

[51] Adam Sealfon and Jonathan Ullman. 2021. Efficiently Estimating Erdos-Renyi Graphs with Node Differential Privacy. *Journal of Privacy and Confidentiality* 11, 1 (2021).

[52] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[53] Sasha Targ, Diogo Almeida, and Kevin Lyman. 2016. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029* (2016).

[54] Hien To, Gabriel Ghinita, Liyue Fan, and Cyrus Shahabi. 2017. Differentially Private Location Protection for Worker Datasets in Spatial Crowdsourcing. *IEEE Trans. Mob. Comput.* 16, 4 (2017), 934–949. https://doi.org/10.1109/TMC.2016.2586058

[55] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. 2016. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems* 29 (2016).

[56] Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems* 30 (2017).

[57] Hanwei Wu and Markus Flierl. 2020. Vector quantization-based regularization for autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 6380–6387.

[58] Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. 2017. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1307–1322.

[59] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2010. Differential privacy via wavelet transforms. *IEEE Transactions on knowledge and data engineering* 23, 8 (2010), 1200–1214.

[60] Yonghui Xiao, Li Xiong, Liyue Fan, and Slawomir Goryczka. 2012. Dpcube: differentially private histogram release through multidimensional partitioning. *arXiv preprint arXiv:1202.5358* (2012).

[61] Ming Xu, Matias Quiroz, Robert Kohn, and Scott A Sisson. 2019. Variance reduction properties of the reparameterization trick. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2711–2720.

[62] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. 2019. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In *The world wide web conference*. 2147–2157.

[63] Dingqi Yang, Daqing Zhang, Longbiao Chen, and Bingqing Qu. 2015. Nationtelescope: Monitoring and visualizing large-scale collective behavior in lbsns. *Journal of Network and Computer Applications* 55 (2015), 170–180.

[64] Dan Zhang, Ryan McKenna, Ios Kotsogiannis, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. 2018. Ektelo: A framework for defining differentially-private computations. In *Proceedings of the 2018 International Conference on Management of Data*. 115–130.

[65] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)* 42, 4 (2017), 1–41.

[66] Jun Zhang, Xiaokui Xiao, and Xing Xie. 2016. Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 International Conference on Management of Data*. 155–170.

[67] Xiaojian Zhang, Rui Chen, Jianliang Xu, Xiaofeng Meng, and Yingtao Xie. 2014. Towards accurate histogram publication under differential privacy. In *Proceedings of the 2014 SIAM international conference on data mining*. SIAM, 587–595.

[68] Dihan Zheng, Sia Huat Tan, Xiaowen Zhang, Zuoqiang Shi, Kaisheng Ma, and Chenglong Bao. 2020. An Unsupervised Deep Learning Approach for Real-World Image Denoising. In *International Conference on Learning Representations*.

A DP PROOF

In our discussion, we use the size of the datasets N and sampled sets n, which we assume are publicly available and, if not, an estimate can be obtained by spending negligible privacy budget. We also use the well known property of DP: the *post-processing property* of differential privacy [22] states that given any arbitrary function h and an ε-DP mechanism $\mathcal{M}$, the mechanism $h(\mathcal{M})$ is ε-DP.

Proof. Alg. 2 shows our proposed end-to-end algorithm. We rewrite the algorithm $\mathcal{M}$ as a composition of mechanisms $\mathcal{M}_s$ and $\mathcal{M}_v$ for the purposes of privacy accounting. M_s is a two step process. The first bounds the number of each user's points to at most k. The second step partitions the domain into a 3-d histogram and for each disjoint cell, queries the sampled dataset for the number of points in its extents, and adds noise sampled from the Laplace distribution of scale Δ_f/ε to this answer. In the mechanism $\mathcal{M}_v$, VDR takes as input the sanitized 3-d histogram, denoises it, and then, scales the counts according to a statistical refinement heuristic.

$\mathcal{M}_s$ prunes each user's data points to at most k, where k is determined in a data-independent manner (as a constant fraction of N, see 5.4 for details). The data domain is partitioned into a 3-d histogram H of granularity $\mathbb{M}\times\mathbb{M}\times\mathbb{T}$. For each cell $c \in H$ we write as f_c the query asking the count of points in the cell c. We consider these queries as a single vector $\mathbf{Q}$, and pose this query set to be answered over the dataset. Therefore, the L_1 sensitivity of $\mathbf{Q}$ is k (i.e., $\Delta_f = k$) and hence the mechanism adding Laplace noise scaled to k/ε to each unit of Q enjoys ε-DP (Theorem 1 of [32]).

Next the mechanism M_v trains the VAE on the sanitized 3-d histogram, and refines the output results *after* the histogram has been sanitized. Therefore the transformation M_v is applied post-sanitization, and due to the post processing property of DP does not consume any privacy budget. Moreover, the statistical refinement step determines scaling factor γ, according to a constant value of C, without running any computation on the private dataset. To conclude, the mechanism $\mathcal{M}$, which is a composition of $\mathcal{M}_s$, $\mathcal{M}_v$ is ε-differentially private. □

B ADDITIONAL EXPERIMENTS

B.1 Additional Visualizations for GMM

Figure 25 and figure 26 visualize additional settings for GMM components.

B.2 HostSpot Query evaluation

Figure 24 reports the Mean Absolute Error (MAE) and Regret metrics of the hotspot query for the Foursquare Tokyo dataset at density thresholds of 10, 20 and 40.

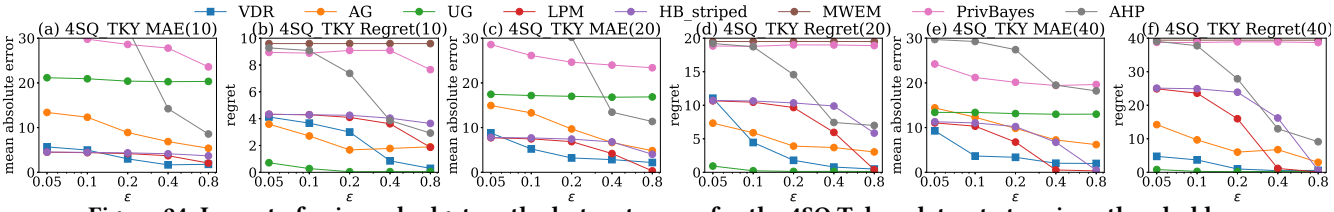

Figure 24: Impact of privacy budget on the hotspot query for the 4SQ Tokyo dataset at various thresholds.

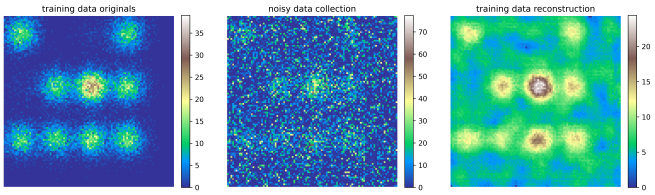

Figure 25: Learning behavior at GMM $\sigma = 5$

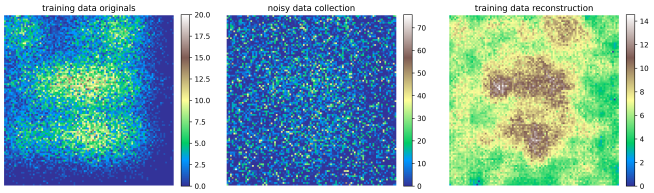

Figure 26: Learning behavior at GMM $\sigma = 9$