A Neural Database for Differentially Private Spatial Range Queries

Sepanta Zeighami*
University of Southern California
zeighami@usc.edu

Ritesh Ahuja*
University of Southern California
riteshah@usc.edu

Gabriel Ghinita
UMass Boston
gabriel.ghinita@umb.edu

Cyrus Shahabi
University of Southern California
shahabi@usc.edu

ABSTRACT

Mobile apps and location-based services generate large amounts of location data. Location density information from such datasets benefits research on traffic optimization, context-aware notifications and public health (e.g., disease spread). To preserve individual privacy, one must sanitize location data, which is commonly done using differential privacy (DP). Existing methods partition the data domain into bins, add noise to each bin and publish a noisy histogram of the data. However, such simplistic modelling choices fall short of accurately capturing the useful density information in spatial datasets and yield poor accuracy. We propose a machine-learning based approach for answering range count queries on location data with DP guarantees. We focus on countering the sources of error that plague existing approaches (i.e., noise and uniformity error) through learning, and we design a neural database system that models spatial data such that density features are preserved, even when DP-compliant noise is added. We also devise a framework for effective system parameter tuning on top of *public* data, which helps set important system parameters without expending scarce privacy budget. Extensive experimental results on real datasets with heterogeneous characteristics show that our proposed approach significantly outperforms the state of the art.

PVLDB Reference Format:
Sepanta Zeighami, Ritesh Ahuja, Gabriel Ghinita, and Cyrus Shahabi. A Neural Database for Differentially Private Spatial Range Queries. PVLDB, 15(5): 1066 - 1078, 2022.
doi:10.14778/3510397.3510404

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/szeighami/SNH.

1 INTRODUCTION

Mobile apps collect large amounts of individual location data used to optimize traffic, study disease spread, or improve point-of-interest placement. When using such data, preserving location privacy is essential, since even aggregate statistics can leak details about individual whereabouts. Existing solutions publish a noisy version of the dataset, transformed according to differential privacy (DP) [13], the de-facto standard for releasing statistical data. The goal of DP mechanisms is to ensure privacy while keeping the query answers as accurate as possible. For spatial data, *range* queries are the most popular query type, used as building blocks in most processing tasks. A DP-compliant *representation* of a spatial dataset is created by partitioning the data domain into bins, and then publishing a *histogram* with the noisy count of points that fall within each bin. *Domain partitioning* is commonly adopted [11, 18, 25, 34, 40, 48], e.g., uniform and adaptive grids [34] or hierarchical partitioning [11, 48].

At query time, the noisy histogram is used to compute answers, by considering the counts in all bins that overlap the query. When a query partially overlaps with a bin, the *uniformity assumption* is used to estimate what fraction of the bin's count should be added to the answer. Since DP mechanisms release only the (noisy) count for each bin, it is assumed that data points are distributed uniformly within the partition, hence the estimate is calculated as the product of the bin count and the ratio of the overlapping area to the total area of the bin. This is often a poor estimate, since location datasets tend to be highly skewed in space (e.g., a shopping mall in a suburb increases mobile user density in an otherwise sparse region). Thus, in addition to DP sanitization noise, *uniformity error* is a major cause of inaccuracy for existing work on DP release of spatial data.

We propose a paradigm shift towards *learned representations* of data, which have been shown to accurately capture data distribution in non-private approximate query processing [19, 27, 46]. Such results show that learning exploits data patterns to accurately and compactly represent the data. As such, learning can be used to combat data modelling errors, also present in DP setting. Nonetheless, due to the impact of DP noise on the process of learning, creating learned differentially private data representations is non-trivial.

Recent attempts at creating learned DP data representations [29, 47] propose the use of learned models to answer queries in non-spatial domains (e.g., categorical data). While these approaches perform well in the case of categorical data, they cannot model the intrinsic properties of location datasets, which exhibit both high skewness, as well as strong correlation among regions with similar designations. For instance, two busy city areas (e.g., a stadium and a street bar area) will exhibit similar density patterns, while the regions in between may be sparse. These busy areas may also be correlated, since people are likely to congregate at bars after they see

Proceedings of the VLDB Endowment, Vol. 15, No. 5 ISSN 2150-8097.
doi:10.14778/3510397.3510404

*Equal contribution.

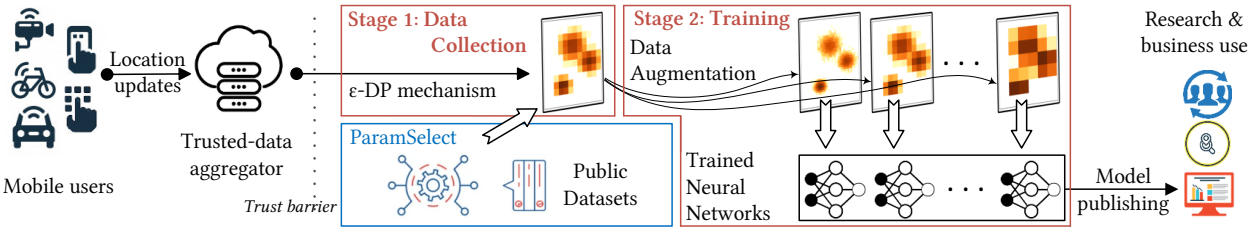

Figure 1: Spatial Neural Histogram System

a game at the stadium. Models with strong representational power in the continuous domain are necessary to learn such patterns.

Meanwhile, training complex models while preserving DP is difficult. For neural networks, existing techniques [4] utilize *gradient perturbation* to train differentially private models. However, the *sensitivity* of this process, defined as the influence a single input record may have on the output (see Section 2 for a formal definition), is high. DP-added noise is proportional to sensitivity, and as a result meaningful information encoded in the gradients is obliterated. The learning process has to be carefully crafted to the unique properties of spatial data, or accuracy will deteriorate.

We propose Spatial Neural Histograms (SNH), a neural network system specifically designed to answer differentially private spatial range queries. SNH models range queries as a *function approximation* task, where we learn a function approximator that takes as input a spatial range and outputs the number of points that fall within that range. Training SNH consists of two stages (Figure 1): the first perturbs training query answers according to DP, while the second trains neural networks from noisy answers. The first stage is called *data collection*. It prepares a differentially private training set for our model while ensuring low sensitivity, such that the signal-to-noise ratio is good. However, due to the privacy constraints imposed by DP, we can only collect a limited amount of training data. Thus, in the second stage, we synthesize more training samples based on the collected data to boost learning accuracy, in a step called *data augmentation*. Then, we employ a *supervised learning* training process with a carefully selected set of training samples comprising of spatial ranges and their answers. SNH learns from training queries *at varying granularity and placement* to capture subtle correlations present within the data. Finally, an extensive private parameter tuning process (ParamSelect) is performed using publicly available data, without the need to consume valuable privacy budget.

The fully trained SNH can then be released publicly and only requires a single *forward pass* to answer a query, making it highly efficient at runtime. SNH is able to learn complex density variation patterns that are specific to spatial datasets, and reduces the negative impact of noise and uniformity assumption when answering range queries, significantly boosting accuracy.

Use of machine learning when answering test queries (i.e., at runtime) is beneficial because, through learning, SNH combines evidence from *multiple* training queries over distinct regions. In fact, gradient computation during training can be seen as a novel means of aggregating information across the space. We show that neural networks can learn the underlying patterns in location data from imprecise observations (e.g., observations collected with noise and uniformity error), use those patterns to answer queries accurately and thereby mitigate noise and uniformity errors. In contrast, existing approaches are limited to using *imprecise local* information only (i.e., within a single bin). When the noise introduced by differential privacy or the error caused by the uniformity assumption are large for a particular bin, the answer to queries evaluated using that bin will be inaccurate.

Contributions and organization. In this paper, we

- Formulate the problem of answering spatial range count queries as a function approximation task (Sec. 2);
- Propose a novel system that leverages neural networks to represent spatial datasets while accurately capturing location-specific density and correlation patterns (Sec. 3, 4);
- Introduce a comprehensive framework for tuning system parameters on *public* data (Sec. 5); and
- Conduct an extensive experimental evaluation on a broad array of public and private real-world location datasets with heterogeneous properties and show that SNH outperforms all the state-of-the-art solutions (Sec. 6).

We survey related work in Section 7 and conclude in Section 8.

2 PRELIMINARIES

2.1 Differential Privacy

ε-differential privacy [13] provides a rigorous privacy framework with formal protection guarantees. Given *privacy budget* parameter $\varepsilon \in (0, +\infty)$, a randomized mechanism $\mathcal{M}$ satisfies ε-differential privacy iff for all datasets D and D', where D' can be obtained from D by either adding or removing one 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 $\mathcal{M}$. $\mathcal{M}$ hides the presence of an individual in the data, since the difference in probability of any set of outcomes obtained on two datasets differing in a single tuple never exceeds 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 Z_f, is the maximum amount the value of f can change when adding or removing a single individual's records from the data. The ε-DP guarantee can be achieved by adding random noise derived from the Laplace distribution $\text{Lap}(Z_f/\varepsilon)$. For a query $f : D \rightarrow \mathbb{R}$, the Laplace mechanism $\mathcal{M}$ returns $f(D) + \text{Lap}(Z_f/\varepsilon)$, where $\text{Lap}(Z_f/\varepsilon)$ is a sample drawn from the probability density function $\text{Lap}(x|(Z_f/\varepsilon)) = (\varepsilon/2Z_f)\exp(-|x|\varepsilon/Z_f)$ [13]. The *composability* property of DP helps quantify the amount of privacy attained when

Table 1: Summary of Notations

Notation	Definition
ε	DP Privacy Budget
$\mathcal{Q}, Q_W$	Query distribution and workload query set
Q_D, Y_D	Data collection query set and its answers
Q_A, Y_A	Augmented query set and its answers
R, k	Set and number of query sizes for training
l, u	Lower and upper bound on query sizes
$f(q)$, ($\hat{f}(q;\theta)$)	Count of records in q calculated from D (estimated from θ)
$\tilde{f}(q)$	$f(q) + Lap(1/\varepsilon)$
ρ, C	Grid granularity, Set of bottom-left corners of grid cells
ψ	Smoothing factor in relative error
Φ,ϕ	ParamSelect Model, Dataset features
$\mathcal{D}, D_T, D_I$	All public datasets, ParamSelect training and inference datasets
$\pi_\alpha(D,\varepsilon)$	Function denoting best value of system parameter α for dataset D and budget ε
$\hat{\pi}_\alpha(D,\varepsilon)$	Empirical estimate of $\pi_\alpha(D,\varepsilon)$

multiple functions are evaluated on the data. Specifically, when mechanisms $\mathcal{M}_1$, $\mathcal{M}_2$ with privacy budgets ε_1, ε_2 are applied in succession on *overlapping* data partitions, the *sequential* composition property [13] states that the budget consumption is $(\varepsilon_1 + \varepsilon_2)$. Conversely, when $\mathcal{M}_1$, $\mathcal{M}_2$ are applied on *disjoint* data partitions, the *parallel* composition property states that the resulting budget consumption is $\max(\varepsilon_1, \varepsilon_2)$. The *post-processing property* of differential privacy [13] states that given any arbitrary function h and an ε-DP mechanism $\mathcal{M}$, the mechanism $h(\mathcal{M})$ is ε-DP. Lastly, we note that DP is robust to side-channel information [13], that is, the privacy guarantee on the DP-release of D is irrespective of any publicly available information about the users in D.

2.2 Problem Definition

Consider a database D that covers a spatial region $SR \subseteq \mathbb{R}^2$, and contains n records each describing an individual's geo-coordinate. Given a privacy budget ε, the problem studied in this paper is to return the answer to an unbounded number of spatial range count queries (RCQs). An RCQ consists of a spatial range predicate and its answer is the number of records in D that satisfy the range predicate. We consider spatial range queries that are axis-parallel and square-shaped, defined by their bottom-left corner c (where c is a vector in SR), and their side length r. An RCQ, q, is then defined by the pair $q = (c, r)$. We say r is the query size and c is its location coordinate. For a database D, the answer to the RCQ $q = (c, r)$ can be written as a function $f(q) = |\{p | p \in D, c[i] \leq p[i] < c[i]+r, \forall i \in \{0,1\}\}|$, where $z[0]$ and $z[1]$ denote the latitude and longitude of any coordinate z, respectively. We assume RCQs follow a distribution $\mathcal{Q}$ and for any RCQ q, we measure the utility of its estimated answer, y, using the *relative error metric*, defined as $\Delta(y, f(q)) = \frac{|y-f(q)|}{\max\{f(q),\psi\}}$, where ψ is a smoothing factor necessary to avoid division by zero.

The typical way to solve the problem of answering an unbounded number of RCQs is to design an ε-DP mechanism $\mathcal{M}$ and a function $\hat{f}$ such that *(1)* $\mathcal{M}$ takes as an input the database D and outputs a differentially private representation of the data, θ; and *(2)* the function $\hat{f}(q;\theta)$ takes the representation θ, together with any input query q, and outputs an estimate of $f(q)$. In practice, $\mathcal{M}$ is used exactly once to generate the representation θ. Given such a representation, $\hat{f}(q;\theta)$ answers any RCQ, q, without further access to the database. For instance, in [34], $\mathcal{M}$ is a mechanism that outputs noisy counts of cells of a 2-dimensional grid overlaid on D. Then, to answer an RCQ q, $\hat{f}(q;\theta)$ takes the noisy grid, θ, and the RCQ, q, as inputs and returns an estimate of $f(q)$ using the grid. The objective is to design $\mathcal{M}$ and $\hat{f}$ such that the relative error between $\hat{f}(q;\theta)$ and $f(q)$ is minimized, that is, to minimize $E_{\theta\sim\mathcal{M}} E_{q\sim\mathcal{Q}}[\Delta(\hat{f}(q;\theta), f(q))]$.

Let $\hat{f}$ be a function approximator and define $\mathcal{M}$ to be a mechanism that learns its parameters. The learning objective of $\mathcal{M}$ is to find a θ such that $\hat{f}(q;\theta)$ closely mimics $f(q)$ for different RCQs, q. The representation of the data, θ, is the set of learned parameters of a function approximator. Mechanism $\mathcal{M}$ outputs a representation θ, and any RCQ, q, is answered by evaluating the function $\hat{f}(q;\theta)$. However, $\mathcal{M}$ is now defined as a learning algorithm and $\hat{f}$ as a function approximator. Our problem is formally defined as follows:

Problem 1. *Given a privacy budget ε, design a function approximator, $\hat{f}$, (let the set of possible parameters of $\hat{f}$ be Θ) and a learning algorithm, $\mathcal{M}$, such that $\mathcal{M}$ satisfies ε-DP and finds*

$$\arg\min_{\theta\in\Theta} E_{q\in\mathcal{Q}}[\Delta(\hat{f}(q;\theta), f(q))]$$

3 SPATIAL NEURAL HISTOGRAMS (SNH)

Our goal is to utilize models that can learn patterns within the data in order to answer RCQs accurately. We employ neural networks as the function approximator $\hat{f}$, due to their ability to learn complex patterns effectively. Prior work [4] introduced a differentially private stochastic gradient descent (DP-SGD) approach to privately train a neural network. Thus, a seemingly straightforward solution to Problem 1 is using a simple fully connected neural network and learning its parameters with DP-SGD. Sec. 3.1 discusses this trivial approach and outlines the limitations of using DP-SGD in our setting, which leads to poor accuracy. Next, in Sec.3.2, we discuss how we improve the training process to achieve good accuracy. In Sec.3.3 we provide an overview of our proposed Spatial Neural Histogram (SNH) solution. Table 1 summarizes the notations.

3.1 Baseline Solution using DP-SGD

Learning Setup. We define $\hat{f}(.;\theta)$ to be a fully connected neural network with parameter set θ. We train the neural network so that for an RCQ q, its output $\hat{f}(q;\theta)$ is similar to $f(q)$. A training set, T, is created, consisting of $(q, f(q))$ pairs, where q is the input to the neural network and $f(q)$ is the training label for the input q (we call RCQs in the training set *training RCQs*). To create the training set, similar to [25, 28], we assume we have access to a set of *workload RCQs*, Q_W, that resembles RCQs a query issuer would ask (e.g., are sampled from $\mathcal{Q}$ or a similar distribution) and is assumed to be public. Thus, we can define our training set T to be $\{(q, f(q)) | q \in Q_W\}$. We define the training loss as

$$\mathcal{L} = \sum_{q\in Q_W} (\hat{f}(q;\theta) - f(q))^2 \quad (2)$$

In a non-private setting, a model can be learned by directly optimizing Eq. (2) using a gradient descent approach. The model can answer any new RCQ q similar to the ground truth $f(q)$.

Incorporating Privacy. DP-SGD [4] incorporates differential privacy for training neural networks. It modifies SGD by clipping each sample gradient to have norm at most equal to a given *clipping*

threshold, B, and obfuscating them with Gaussian noise. Intuitively, the clipping threshold, B, disallows learning more information than a set quantity from any given training sample (no matter how different it is from the rest) and the standard deviation of the Gaussian noise added is scaled with B to ensure obfuscation is proportional to the amount of information gained per sample. Specifically, in each iteration: (1) a subset, S, of the training set is sampled; (2) for each sample, $s = (x, y) \in S$, the gradient $g_s = \nabla_\theta(\hat{f}(x;\theta) - y)^2$ is computed, and clipped (i.e., truncated) to a maximum ℓ_2-norm of B as $\bar{g}_s = \min(\|g_s\|_2, B)\frac{g_s}{\|g_s\|_2}$; (3) the average clipped gradient value for samples in S is obfuscated with Gaussian noise as

$$g = \sum_{s\in S}(\bar{g}_s) + \mathcal{N}(0, \sigma^2 B^2) \tag{3}$$

(4) the parameters are updated in the direction opposite to g.

DP-SGD Challenges. In our problem setting, the training set is created by querying D to obtain the training labels, and our goal is to ensure the privacy of records in D. On the other hand, DP-SGD considers the training set itself to be the dataset whose privacy needs to be secured. This changes the sensitivity analysis of DP-SGD. In our setting, to compute the sensitivity of the gradient sum $\sum_{s\in S}(\bar{g}_s)$ in step (3) of DP-SGD, we have to consider the worst-case effect the presence or absence of a single geo-coordinate record p can have on the sum (as opposed to the worst-case effect of the presence or absence of a single training sample). Removing p can potentially affect every $\bar{g}_s$ for all $s \in S$, so sensitivity of the gradient sum is $|2S| \times B$ and Gaussian noise of $\mathcal{N}(0, \sigma^2 4|S|^2 B^2)$ must be added to the gradient sum to achieve DP (cf. noise in step (3) above). After this adjustment, per-iteration and total privacy consumption of DP-SGD is amplified, impairing learning. We experimentally observed that, for any reasonable privacy budget, training loss does not improve at all during training due to the large added noise.

3.2 A different learning paradigm for RCQs

Next, we introduce *three design principles (P1-P3)* we follow when training neural networks to answer RCQs. These principles are then used in Sec. 3.3 to build our solution.

P1: Separation of noise addition from training. The main reason DP-SGD fails in our problem setting is that too much noise needs to be added when calculating gradients privately. Recall that DP-SGD uses the quantity g, defined in Eq. (3), as the differentially private estimate of the gradient of the loss function. Here, we investigate the private gradient computation in more details to provide an alternative method to calculate the gradient with differential privacy. Recall that the goal is to obtain the gradient of the loss function, $\mathcal{L}$, defined in Eq. (2) with respect to the model parameters. We thus differentiate $\mathcal{L}$ and obtain:

$$\nabla_\theta \mathcal{L} = \sum_{q\in Q_W} 2 \times (\underbrace{\hat{f}(q;\theta)}_{\substack{\text{data}\\\text{indep.}}} - \underbrace{f(q)}_{\substack{\text{data}\\\text{dep.}}}) \times \underbrace{\nabla\hat{f}(q;\theta)}_{\substack{\text{data}\\\text{indep.}}} \tag{4}$$

In Eq. (4), only $f(q)$ accesses the database. This is because the training RCQs in Q_W (i.e., the inputs to the neural network), are created independently of the database. The data dependent term requires computing private answers to $f(q)$ for an RCQ q, hence must consume budget, while the data-independent terms can be calculated without spending any privacy budget. This decomposition of the gradient into data dependent and independent terms is possible because, different from typical machine learning settings, the differential privacy is defined with respect to the database D and not the training set (as discussed in Sec. 3.1).

Instead of directly using g (Eq. (3)) as the differentially private estimate of the gradient (where the gradients are clipped and noise is added to the clipped gradients), we calculate a differentially private value of the training label $f(q)$, called $\bar{f}(q)$, by adding noise to the label (define $\bar{f}(q) = f(q) + \text{Lap}(1/\varepsilon)$) and calculate the gradient from that. The differentially private estimate of the gradient is then

$$g = \sum_{q\in Q_W} 2 \times (\hat{f}(q;\theta) - \bar{f}(q)) \times \nabla\hat{f}(q;\theta) \tag{5}$$

A crucial benefit is that $\bar{f}(q)$, does not change over successive learning iterations. That is, the differentially private value $\bar{f}(q)$ can be computed once and used for all training iterations. This motivates our first design principle of separating noise addition and training. This way, training becomes a two step process: first, for all $q \in Q_W$, we calculate the differentially private training label $\bar{f}(q)$. We call this step *data collection*. Then, we use a training set consisting of pairs $(q, \bar{f}(q))$ for all $q \in Q_W$ for training. Since DP-compliant data measurements are obtained, all future operations that use as input these measurements are also ε-differentially private according to the *post-processing property* of differential privacy [13]. Thus, the training process is done as in a non-private setting, where a conventional SGD algorithm can be applied (i.e., we need not add noise to gradients), and differential privacy is still satisfied.

P2: Spatial data augmentation through partitioning. Following principle P1, privacy accounting is only needed when answering training queries to collect training labels. Meanwhile, in our experiments, we observed that training accurate neural networks requires a training set containing queries of different sizes (see Sec. 6.3.2). Such queries may overlap and, if we answer them directly from the database, *sequential composition* theorem would apply to account for the total privacy budget consumption. This way, the more such queries we answer, the more budget needs to be spent.

Instead, to avoid spending extra privacy budget while creating more training samples with multiple query sizes, we propose *spatial data augmentation* through partitioning. First, we use a *data collection query set*, Q_D, chosen such that RCQs in Q_D don't overlap (i.e., a space partitioning). This ensures *parallel composition* can be used for privacy accounting, instead of sequential composition, which allows answering all RCQs in Q_D by spending budget equal to one RCQ. Then, using the partitioning Q_D, we create and answer new queries, q, of different sizes without spending any more privacy budget but by making uniformity assumption across cells in Q_D that partially overlap q. Even though this approach introduces uniformity error in our training set, it avoids adding the otherwise required large scale noise, and boosts accuracy. Thus, it allows us to optimize the uniformity/noise trade-off [11, 34] when creating our training set (we present experiments in Sec. B.2 of our technical report [45] to show that data augmentation reduces error).

P3: Learning at multiple granularities. We employ in our solution multiple models that learn at different granularities, each designed to answer RCQs of a specific size. Intuitively, it is more difficult for a model to learn patterns when both query size and

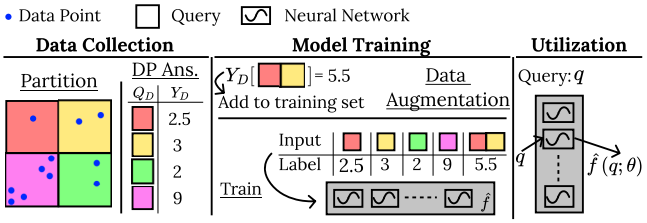

Figure 2: SHN Overview

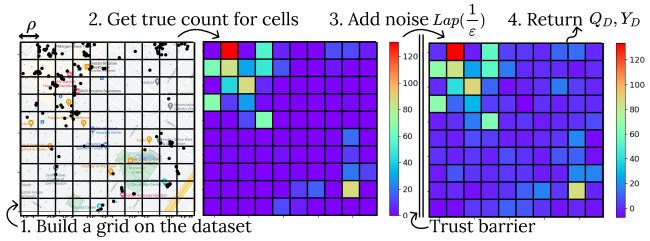

Figure 3: Data Collection: map view (left), true cell count heatmap (middle), ε-DP heatmap with noisy counts (right)

locations change. Using multiple models allows each model to learn the patterns relevant to the granularity they operate on.

3.3 Proposed approach: SNH

Our Spatial Neural Histograms (SNH) design, illustrated in Figure 2, consists of three steps: (1) Data collection, (2) Model Training, and (3) Model Utilization. We provide a summary of each step below, and defer details until Sec. 4.

Data Collection. This step partitions the space into non-overlapping RCQs that are directly answered with DP-added noise. The output of this step is a *data collection query set*, Q_D, and a set Y_D which consists of the differentially private answers to RCQs in Q_D. This is the only step in SNH that accesses the database. In Fig. 2 for example, the query space is partitioned into four RCQs, and a differentially private answer is computed for each.

Training. Our training process consists of two stages. First, we use *spatial data augmentation* to create more training samples based on Q_D. An example is shown in Fig. 2, where an RCQ covering both the red and yellow squares is not present in the set Q_D, but it is obtained by aggregating its composing sub-queries (both in Q_D). Second, the augmented training set is used to train a function approximator $\hat{f}$ that captures f well. $\hat{f}$ consists of a set of neural networks, each trained to answer different query sizes.

Model Utilization. This step decides how any previously unseen RCQ can be answered using the learned function approximator, and how different neural networks are utilized to answer an RCQ.

4 TECHNICAL DETAILS

4.1 Step 1: Data Collection

This step creates a partitioning of the space into non-overlapping bins, and computes for each bin a differentially private answer. We opt for a simple equi-width grid of cell width ρ as our partitioning method. As illustrated in Fig. 3, *(1)* we overlay a grid on top of the data domain; *(2)* we calculate the true count for each cell in the grid, and *(3)* we add noise sampled from $Lap(\frac{1}{\varepsilon})$ to each cell count. We represent a cell by the coordinates of its bottom left corner, c, so that getting the count of records in each cell is an RCQ, $q = (c, \rho)$. Let C be the set of bottom left coordinates of all the cells in the grid. Furthermore, recall that for a query q, $\tilde{f}(q) = f(q) + Lap(\frac{1}{\varepsilon})$. Thus, the data collection query set is defined as $Q_D = \{(c, \rho), c \in C\}$, and their answers are the set $Y_D = \{\tilde{f}(c, \rho), c \in C\}$. We use $Y_D[c]$ to refer to the answer for the query located at c in Y_D. The output of the data collection step consists of sets Q_D and Y_D.

Even though more complex partitioning structures have been used previosuly for privately answering RCQs [34, 48], we chose a simple regular grid, for two reasons. First, our focus is on a novel neural database approach to answering RCQs, which can be used in conjunction with *any* partitioning type – using a simple grid allows us to isolate the benefits of the neural approach. Second, using more complex structures in the data collection step may increase the impact of uniformity error, which we attempt to suppress through our approach. The neural learning step captures density variations well, and conducting more complex DP-compliant operations in the data collection step can have a negative effect on overall accuracy. In our experiments, we observed significant improvements in accuracy with the simple grid approach. While it may be possible to improve the accuracy of SNH by using more advanced data collection methods, we leave that study for future work.

The challenge in data collection is choosing the value of ρ to minimize induced errors. We address this thoroughly in Sec. 5.1 and present a method to determine the best granularity of the grid.

4.2 Step 2: SNH Training

Given query set Q_D and its sanitized answers, we can perform any operation on this set without privacy leakage due to the post-processing property of DP. As discussed in Sec. 3.3, we first perform a data augmentation step using Q_D to create an augmented training set Q_A. Then, Q_A is used for training our function approximator.

Data Augmentation is a common machine learning technique to increase the number of samples for training based on the existing (often limited) available samples [24, 49]. We propose spatial data augmentation for learning to answer RCQs. Our proposed data augmentation approach is based on our design principle P2, discussed in Sec. 3.2, where we motivate augmenting the training set through partitioning. In the data augmentation step, we create new queries of different sizes, answer them using the partitioning, and add the answers to our training set, as detailed in the following.

We use the partitioning defined by Q_D and corresponding answers Y_D to answer queries at the same locations as in Q_D but of *other* sizes. Consider a query location $c \in C$ and a query size r, $r \neq \rho$. We estimate the answer for RCQ $q = (c, r)$ as $\sum_{c' \in C} \frac{|(c,r) \cap (c', \rho)|}{\rho^2} \times Y_D[c]$, where $|(c, r) \cap (c', \rho)|$ is the overlapping area of RCQs (c, r) and (c', r). In this estimate, noisy counts of cells in Q_D fully covered by q are added as-is (since $|(c, r) \cap (c', \rho)| = \rho^2$), whereas fractional counts for partially-covered cells are estimated using the uniformity assumption. Fig. 4 shows how we perform data augmentation for a

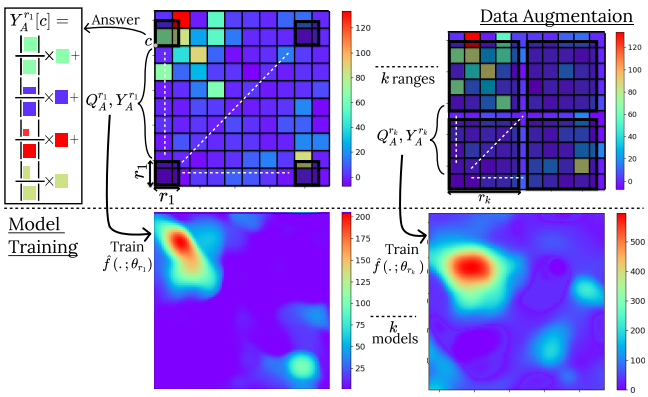

Figure 4: Model Training: Augmented query sets of size r_1 to r_k (top) are used to learn neural network models (bottom)

Algorithm 1 Spatial data augmentation

Input: Query set Q_D with answers Y_D, k query sizes
Output: Augmented training set Q_A with labels Y_A

1: $R \leftarrow \{l + \frac{(u-l)}{k} \times (i + \frac{1}{2}), \forall i, 0 \leq i < k\}$
2: **for all** $r \in R$ **do**
3: $\quad Q_A^r, Y_A^r \leftarrow \emptyset$
4: $\quad$ **for** $(c, \rho) \in Q_D$ **do**
5: $\quad\quad Q_A^r.append((c, r))$
6: $\quad\quad Y_A^r[c] \leftarrow \sum_{(c',\rho) \in Q_D} \frac{|(c,r) \cap (c',\rho)|}{\rho^2} \times Y_D[c']$
7: **return** $Q_A, Y_A \leftarrow \{Q_A^r, \forall r \in R\}, \{Y_A^r, \forall r \in R\}$

query (c, r_1) with size r_1 at location c. Also observe that, by using queries at the same locations as in Q_D, the bottom-left corners of all queries in the augmented query set are aligned with the grid.

We repeat this procedure for k different query sizes to generate sufficient training data. To ensure coverage for all expected query sizes, we define the set of k sizes to be uniformly spaced. Specifically, assuming the test RCQs have size between l and u, we define the set R as the set of k uniformly spaced values between l and u, and we create an augmented training set for each query size in R. This procedure is shown in Alg.1. We define Q_A^r for $r \in R$ to be the set of RCQs located at C but with query size r, that is $Q_A^r = \{(c, r), c \in C\}$, and define Y_A^r to be the set of the estimates for queries in Q_A^r obtained from Q_D and Y_D. The output of Alg. 1 is the augmented training set containing training samples for different query sizes. Note that, as seen in the definition above, Q_A^r, for any r, only contains queries whose bottom-left corner is aligned with the grid used for data collection to minimize the use of the uniformity assumption. However, uniformity errors can still be present in our answers in Y_A^r. We discuss in Sec. 4.3 how training of neural networks on top of these answers allows us to mitigate the uniformity error through learning.

Model architecture. We find that using multiple neural networks, each trained for a specific query size, performs better than using a single neural network to answer queries of all sizes. Thus, we train k different neural networks, one for each $r \in R$. Meaning that a single neural network trained for query size r can only answer queries

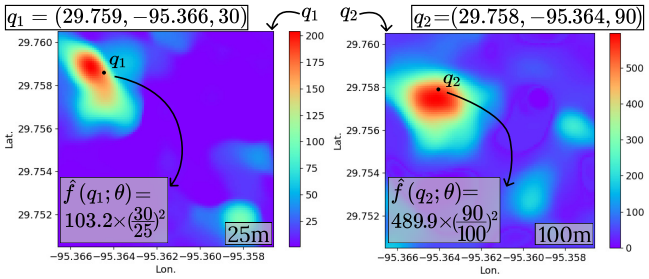

Figure 5: Model utilization: 30m query answered from 25m network (left), 90m query from 100m network (right)

of size r (we discuss in Sec. 4.3 how the neural networks are used to answer other query sizes), accordingly the input dimensionality of each neural network is two, i.e., lat. and lon. of the location of the query. We use k identical fully-connected neural networks (specifics of the network architecture are discussed in Sec. 6).

Loss function and Optimization. We train each of the k neural networks independently. We denote by Q_A^r the training set for a neural network $\hat{f}(.;\theta_r)$, trained for query size r, and we denote the resulting labels by Y_A^r. We use a mean squared error loss function to train the model, but propose two adjustments to capitalize on the workload information available. First, note that for a query size $r \in R$, Q_A^r is comprised of queries at uniformly spaced intervals, which may not follow the query distribution $\mathcal{Q}$. However, we can exploit properties of the workload queries, Q_W to tune the model for queries from $\mathcal{Q}$. Specifically, for any $(c, r) \in Q_A^r$, let $w_{(c,r)} = |\{q' \in Q_W, (c, r) \cap q' \neq \emptyset\}|$, that is, $w_{(c,r)}$ is the number of workload queries that overlap a training query. In our loss function, we weight every query (c, r) by $w_{(c,r)}$. This workload-adaptive modification to the loss function emphasizes the regions that are more popular for a potential query issuer. Second, we aim at answering queries with low relative error, whereas a mean square loss puts more emphasis on absolute error. Thus, for a training query (c, r), we also weight the sample by $1/\max\{Y_A^r[c], \psi\}$. Putting these together, the loss function optimized for each neural network is

$$\sum_{(c,r) \in Q_A^r} \frac{w_{(c,r)}}{\max\{Y_A^r[c], \psi\}} (\hat{f}(c, \theta_r) - Y_A^r[c])^2 \quad (6)$$

4.3 Model Utilization

To answer a *new* query (c, r), the model that is trained to answer queries with size most similar to r is accessed. That is, we find $r^* = \arg\min_{r' \in R} |r - r'|$ and we answer the query using network $\hat{f}(c, \theta_{r^*})$. The output answer is scaled to r according to a uniformity assumption, and the scaled answer is returned, i.e., $(\frac{r}{r^*})^2 \hat{f}(c, \theta_{r^*})$. Fig. 5 shows this procedure for two different RCQs.

It is important to differentiate the use of uniformity assumption before learning (i.e., in data augmentation), called uniformity assumption *pre-learning*, from the use of uniformity assumption after learning (during model utilization), called uniformity assumption *post-learning*. The parameter k allows exploring the spectrum between the two cases. Specifically, when k is larger, we train more models and each model is trained for a different query size. For

each query size, data augmentation uses uniformity assumption to generate training samples. Thus, more training samples are created using uniformity assumption. We call this *increasing* uniformity assumption pre-learning. On the other hand, since more models are trained, the output of each model will be scaled by a factor closer to one (i.e., in the above paragraph, r^* becomes closer to r so that $(\frac{r}{r^*})^2$ becomes closer to 1). We call this *decreasing* uniformity assumption post-learning. Our experimental results in Sec. 6.3.2 show that increasing k improves accuracy, and k should be set as large as possible so that uniformity assumption post-learning becomes negligible in practice. This follows the SNH motivation (and observations in Sec. 6.3.4) that learning can mitigate the uniformity error. That is, the uniformity assumption should be made pre-learning so that its impact on final accuracy can be reduced through learning.

5 END-TO-END SYSTEM ASPECTS

5.1 System Tuning with ParamSelect

Choosing a good grid granularity, ρ, is crucial for achieving high accuracy for DP spatial data publishing, and studied in previous work [17, 34]. Discretizing continuous domain geo-coordinates creates uniformity errors, and hence the granularity of the grid must be carefully tuned to compensate for the effect of discretization. Existing work [17, 34] makes simplifying assumptions to analytically model the impact of grid granularity on the accuracy of answering queries. However, modelling data and query specific factors is difficult and the simplifying assumptions are often not true in practice, as our experiments show (see Sec. B.3 of our technical report [45]). Instead, we learn a model that is able to predict an advantageous grid granularity for the specific dataset, query distribution and privacy budget. Sec. 5.1.1, discusses *ParamSelect*, our approach to determine ρ. In Sec. 5.1.2 we show how to extend ParamSelect to tune other system parameters.

5.1.1 ParamSelect for ρ. The impact of grid granularity on privacy-accuracy trade-offs when answering queries is well-understood in the literature [34]. In SNH, the grid granularity in *data collection* phase impacts the performance as follows. On the one hand, smaller grid cells increase the resolution at which the data are collected, thereby reducing the uniformity error. Learning is also improved, due to more training samples being extracted. On the other hand, creating too fine grids can diminish the signal-to-noise ratio for cells with small counts, since at a given ε the magnitude of noise added to any cell count is fixed. Moreover, during data augmentation, aggregating multiple cells leads to increase in the total noise variance, since the errors of individual cells are summed. SNH is impacted by cell width in multiple ways, and determining a good cell width, ρ, is important to achieve good accuracy.

Capturing an analytical dependence may not be possible, since numerous data, query and modelling factors determine the ideal cell width. If data points are concentrated in some area where the queries fall, a finer grid can more accurately answer queries for the query distribution (even though signal-to-noise ratio may be poor for parts of the space where queries are not often asked). This factor can be measured only by looking at the actual data and the distribution of queries, and would require spending privacy budget.

The best value of ρ depends on the privacy budget ε, the distribution of points in D and the query distribution $\mathcal{Q}$. Define $\delta(\rho, D, \varepsilon)$ to be the error of SNH with cell width ρ and define $\pi(D, \varepsilon) = \arg\min_{\rho \in \mathbb{R}} \delta(\rho, D, \varepsilon)$, that is, the function that outputs the ideal cell width. We learn a model, Φ, to approximate $\pi(D, \varepsilon)$. We refer to Φ as *regressor* to distinguish it from the SNH model, $\hat{f}$, discussed in Sec. 4. The learning process is similar to any supervised learning task, where for different dataset and privacy budget pairs, (D, ε), we use the label $\pi(D, \varepsilon)$ to train Φ. The input to the regressor is (D, ε) and the training objective is to get the output, $\Phi(D, \varepsilon)$, to be close to the label $\pi(D, \varepsilon)$.

Feature engineering. Learning a regressor that takes a raw database D as input is infeasible, due to the high sensitivity of learning with privacy constraints. Instead, we introduce a feature engineering step that, for the dataset D, outputs a set of features, ϕ_D. Training then replaces D with ϕ_D. Let the spatial region of D be SR_D. First, as one of our features, we measure the skewness in the spread of individuals over SR_D, since this value directly correlates with the expected error induced by using the uniformity assumption. In particular, we (1) discretize SR_D using an equi-width partitioning, (2) for each cell, calculate the probability of a point falling into a cell as the count of points in the cell normalized by total number of points in D, and (3) take the Shannon's Entropy h_D over the probabilities in the flattened grid. However, calculating h_D on a private dataset violates differential privacy. Instead, we utilize *publicly available* location datasets as an auxiliary source to approximately describe the private data distribution for the same spatial region. We posit that there exist high-level similarities in distribution of people's locations in a city across different private and public datasets for the same spatial regions and thus, the public dataset can be used as a surrogate. Let $\mathcal{D}$ be the set of public datasets that we have access to, and let $D_I \in \mathcal{D}$ be a public dataset covering the same spatial region as D. We estimate h_D for a dataset with h_{D_I}. We call D_I public *ParamSelect Inference dataset.*

Second, we use data-independent features: ε, $\frac{1}{n \times \varepsilon}$ and $\frac{1}{\sqrt{n \times \varepsilon}}$, where the product of $n \times \varepsilon$ accounts for the fact that decreasing the scale of the input dataset and increasing epsilon have equivalent effects on the error. This is also understood as epsilon-scale exchangeability [17]. We calculate $\phi_{D,\varepsilon} = (n, \varepsilon, \frac{1}{n\varepsilon}, \frac{1}{\sqrt{n\varepsilon}}, h_{D_I})$ as the set of features for the dataset D without consuming any privacy budget in the process. Lastly, we remark that for regions where an auxiliary source of information is unavailable, we may still utilize the data-independent features to good effect. In our technical report [45], we show that our proposed features achieve reliable accuracy across datasets; particularly, we chose h_D amongst several alternative data-dependent features for that reason.

Training Sample Collection. Generating training samples for Φ is not straightforward since we do not have an analytical formulation for $\delta(\rho, D, \varepsilon)$ and thus $\pi(D, \varepsilon)$. Since the exact value of $\pi(D, \varepsilon)$ is unknown, we use an empirical estimate. We run SNH with various grid granularities of *data collection* and return the grid size, $\rho_{D,\varepsilon}$, for which SNH achieves the lowest error. Our experimental results in Sec. 6.3 show that $\delta(\rho, D, \varepsilon)$ is only marginally affected with small changes in ρ (so evaluating $\delta(\rho, D, \varepsilon)$ at different values of ρ five meters apart and selecting the best ρ provides a good estimate of $\pi(D, \varepsilon)$). Intuitively, one expects the error in the training set to remain the same if the cell width of data collection grid changes by a few meters, since the uniformity errors induced are similar.

Algorithm 2 ParamSelect training

Input: A set of public training datasets $D_T \subseteq \mathcal{D}$ and privacy budgets $\mathcal{E}$ for training to predict a system parameter α
Output: Regressor Φ_α for system parameter α

procedure $\phi(D, n, \varepsilon)$
  $h_D \leftarrow$ entropy of D
  return $(n, \varepsilon, \frac{1}{n\varepsilon}, \frac{1}{\sqrt{n\varepsilon}}, h_D)$
procedure TRAIN_PARAMSELECT(D_T, $\mathcal{E}$)
  $T \leftarrow \{(\phi(D, |D|, \varepsilon), \hat{\pi}_\alpha(D, \varepsilon)) | \varepsilon \in \mathcal{E}, D \in D_T\}$
  $\Phi_\alpha \leftarrow$ Train regressor using T
  return Φ_α

Algorithm 3 ParamSelect usage

Input: Spatial extent SR and size n of a sensitive dataset D and privacy budget ε
Output: System parameter value α for private dataset D

procedure PARAMSELECT(SR, n, ε)
  $D_I \leftarrow$ Public dataset with spatial extent SR
  $\alpha \leftarrow \Phi_\alpha(\phi(D_I, n, \varepsilon))$
  return α

Thus, we use this approach to obtain $\rho_{D,\varepsilon}$ as our training label. Note that the empirically determined value of $\rho_{D,\varepsilon}$ is dependent on—and hence accounts for—the query distribution on which SNH error is measured. Moreover, when D contains sensitive data, obtaining $\rho_{D,\varepsilon}$ would require spending privacy budget. Instead, we generate training records from a set of datasets, $D_T \subseteq \mathcal{D}$ that have already been publicly released (see Sec. 6 for details of public datasets). We call datasets in D_T public *ParamSelect Training datasets*. Put together, our training set is $\{(\phi_{D,\varepsilon}, \rho_{D,\varepsilon}) | \varepsilon \in \mathcal{E}, D \in D_T\}$, where $\mathcal{E}$ is the range of different privacy budgets chosen for training.

Predicting Grid Width with ParamSelect. The training phase of ParamSelect builds regressor Φ using the training set described above. We observed that models from the decision tree family perform the best for this task. Once the regressor is trained, its utilization for any unseen dataset is straightforward and only requires calculating the corresponding features and evaluating Φ.

5.1.2 Generalizing ParamSelect to any system parameter. We can easily generalize the approach in Sec. 5.1.1 to any system parameter. Define function $\pi_\alpha(D, \varepsilon)$ that given a query distribution, outputs the best value of α for a certain database and privacy budget. The goal of ParamSelect is to learn a regressor, using public datasets $D_T \in \mathcal{D}$, that mimics the function $\pi_\alpha(.)$.

ParamSelect functionality is summarized in Alg. 2. First, during a pre-processing step, it defines the feature extraction function $\phi(D, n, \varepsilon)$, that extracts the features described in Sec. 5.1.1 from the public dataset D with n records, and a privacy budget ε. Second, it creates the training set $\{(\phi(D, |D|, \varepsilon), \hat{\pi}_\alpha(D, \varepsilon)), \varepsilon \in \mathcal{E}, D \in D_T\}$, where $\hat{\pi}_\alpha(D, \varepsilon)$ estimates the value of $\pi_\alpha(D, \varepsilon)$ with an empirical search (i.e., by trying different values of α and selecting the one with the highest accuracy), and D_T and $\mathcal{E}$ are different public datasets and values of privacy budget, respectively, used to collect training samples. Lastly, it trains a regressor Φ_α that takes extracted features as an input and outputs a value for α.

Table 2: Urban datasets characteristics.

Low Pop. density	Medium Pop. density	High Pop. density
Fargo [46.877, -96.789]	Phoenix [33.448, -112.073]	**Miami** [25.801, -80.256]
Kansas City [39.09, -94.59]	Los Angeles [34.02, -118.29]	Chicago [41.880, -87.70]
Salt Lake [40.73, -111.926]	Houston [29.747, -95.365]	SF [37.764, -122.43]
Tulsa [36.153, -95.992]	**Milwaukee** [43.038, -87.910]	Boston [42.360, -71.058]

At inference stage (Alg. 3) ParamSelect uses a public dataset D_I that covers the same spatial region as D, as well as size of D, n, and privacy budget ε to extract features $\phi(D_I, n, \varepsilon)$. The predicted system parameter value for D is then $\Phi_\alpha(\phi(D_I, n, \varepsilon))$.

5.2 Privacy and Security Discussion

Let D be a private dataset covering a spatial region SR and $\mathcal{D}$ be a set of public datasets. The SNH end-to-end privacy mechanism $\mathcal{M}$ is comprised of two parts that compose sequentially: mechanism $\mathcal{M}_f$, that models range count queries using the neural networks, and mechanism $\mathcal{M}_\Phi$, that trains a regressor to determine the system parameters. $\mathcal{M}_f$ operates over D, ε, SR and $\mathcal{D}$. $\mathcal{M}_\Phi$ operates over $\mathcal{D}$ and SR for ParamSelect training and inference. Hence, we write the end-to-end system as the SNH mechanism $\mathcal{M}(D|\varepsilon, SR, \mathcal{D}) = \mathcal{M}_f(D|\varepsilon, \mathcal{D}, SR, \mathcal{M}_\Phi(\mathcal{D}, SR))$.

THEOREM 5.1. *Mechanism $\mathcal{M}(D|\varepsilon, SR, \mathcal{D})$ satisfies ε-DP.*

Sec. A of our technical report [45] contains a proof of the above theorem and a qualitative discussion on DP privacy guarantees.

6 EXPERIMENTAL EVALUATION

Sec. 6.1 describes the experimental testbed. Sec. 6.2 evaluates SHN in comparison with state-of-the-art approaches. Sec. 6.3 provides an ablation study of various design choices. Sec. B of our technical reports [45] contains complementary experimental results.

6.1 Experimental Settings

6.1.1 Datasets. We first describe all the datasets and then specify how they are utilized in our experiments.

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 first dataset is a subset of the user check-ins collected by the SNAP project [10] from the *Gowalla* (GW) network. It contains 6.4 million records from 200k unique users during a time period between February 2009 and October 2010. Our second dataset, SF-CABS-S (CABS) [33], is derived from the GPS coordinates of approximately 250 taxis collected over 30 days in San Francisco. Following [17, 34], we keep only the start point of the mobility traces, for a total of 217k records. The third dataset is proprietary, obtained from Veraset [2] (VS), a data-as-a-service company that provides anonymized movement data from 10% of the cellphones in the U.S [3]. For a single day in December 2019, there were 2.6 billion readings from 28 million distinct devices. From VS we generate the fourth dataset called SPD-VS. We perform Stay Point Detection (SPD) [42] on the data to remove location signals when a person is moving, and to extract POI visits when a user is stationary. SPD is useful for POI services [32], and results in a data distribution consisting of user visits (i.e., fewer points on roads and

more at POIs). Following [42], we consider as location visit a region 100 meters wide where a user spends at least 30 minutes.

To simulate a realistic urban environment, we focus on check-ins from several cities in the U.S. We group cities into three categories based on their population densities [1], measured in people per square mile: *low density* (lower than 1000/sq mi), *medium density* (between 1000 and 4000/sq mi) and *high density* (greater than 4000/sq mi). A total of twelve cities are selected, four in each population density category as listed in Table 2. For each city, we consider a large spatial region covering a $20 \times 20\text{km}^2$ area centered at [lat, lon]. From each density category we randomly select a *test city* (highlighted in bold in Table 2), while the remaining cities are used as *training cities*. We use the notation <*city*> (<*dataset*>) to refer to the subset of a *dataset* for a particular *city*, e.g., Milwaukee (VS) refers to the subset of VS datasets for the city of Milwaukee.

Experiments on VS. *Private dataset*: Our experiments on Veraset can be seen as a case-study of answering RCQs on a proprietary dataset while preserving differential privacy. We evaluate RCQs on the Veraset dataset for the *test cities*. Due to the enormous volume of data, we sample at random sets of n check-ins, for $n \in \{25k, 50k, 100k, 200k, 400k\}$ for the test cities and report the results on these datasets. *Auxiliary Datasets*: For each test city in VS, we set Q_W and D_I to be the GW dataset from the corresponding city. GW and VS datasets are completely disjoint (they are collected almost a decade apart). The public datasets D_T are the set of all the training cities of the GW dataset.

Experiments on GW. *Private dataset*: We present the results on the complete set of records for the test cities of Miami, Milwaukee and Kansas City with 27k, 32k and 54k data points, respectively. *Auxiliary Datasets*: For each test city, we set Q_W and D_I to be the VS counterpart dataset for that city. D_T contains all the training cities in the GW dataset. None of the test cities, which are considered sensitive data, are included in D_T.

Experiments on CABS. *Private dataset*: Since CABS consists of 217k records within the city of San Francisco only, we treat it as the sensitive test city for publishing. *Auxiliary Datasets*: We set Q_W and D_I to be the GW dataset for San Francisco. D_T contains all the training cities in the GW dataset. Once again, collecting auxiliary information from an entirely different dataset ensures no privacy leakage on the considered private dataset.

6.1.2 SNH system parameters. We use the GW dataset to train the ParamSelect regression model. For the nine training cities and five values of privacy budget ε, we obtain 45 training samples. We utilize an AutoML *pipeline* (such as [15, 41]) to find out a suitable model from among a wide range of ML algorithms. The pipelines use cross-validation to evaluate goodness-of-fit for possible algorithm and hyperparameter combinations. The final model is an Extremely Randomized Tree (ExtraTrees) [16]. ExtraTrees create an ensemble of random forests [21], where each tree is trained using the whole learning sample (rather than a bootstrap sample). The model ensembles 150 trees having a maximum depth of 7.

For other system parameters, we observed that their best value for SNH remain stable over various dataset and privacy budget combinations. Sec. 6.3.2 and Sec. B.4 of our technical report [45] present this result for parameter k and Sec. 6.3.4 and Sec. B.4 of our technical report [45] for the model depth. We observed no benefit in using ParamSelect to set these parameters and merely selected a value that performed well on our public datasets for the system parameter k and neural network hyper-parameters. The fully connected neural networks contain 20 layers of 80 unit each and are trained with Adam [23] optimizer with learning rate 0.001.

6.1.3 Other experimental settings.

Evaluation Metric. We construct query sets of 5,000 RCQs centered at uniformly random positions. Each query has side length that varies uniformly from 25 meters to 100 meters. We evaluate the relative error for a query q as defined in Sec. 2, and set smoothing factor ψ to 0.1% of the dataset cardinality n, as in [11, 34, 48].

Baselines. We evaluate our proposed SNH approach in comparison to state-of-the-art DP solutions: PrivTree [48], Uniform Grid (UG) [34], Adaptive Grid (AG) [34] and Data and Workload Aware Algorithm (DAWA) [25]. Brief summaries of each method are provided in Sec. 7. DAWA requires the input data to be represented over a discrete 1D domain, which can be obtained by applying a Hilbert transformation. To this end, we discretize the domain of each dataset into a uniform grid with 2^{20} cells, following the work of [25, 48]. DAWA also uses the workload query set, Q_W, as specified in Sec. 6.1.1. For PrivTree, we set its fanout to 4, following [48]. We also considered Hierarchical methods in 2D (HB2D) [18, 35] and QuadTree [11], but the results were far worse than the above approaches and thus are not reported (we report the results of all the baselines in Sec. B.1 of our technical report [45]). As an additional baseline, we modify STHoles [8], a non-private workload-aware algorithm, to satisfy DP. STHoles builds nested buckets in regions where the workload requires finer granularity. We incorporate differential privacy by (1) adding the required sanitization noise to the frequency counts in STHoles' buckets and (2) implementing the algorithm so that it avoids asking overlapping queries from the database to minimize the magnitude of noise added. Details of our DP-compliant adoption of STHoles are available in the Appendix C of our technical report [45] and our implementation is publicly available at [43]. Similar to DAWA and SNH, STHoles uses the workload query set, Q_W, as specified in Sec. 6.1.1.

Implementation. All algorithms were implemented in Python, and executed on a Linux machine with an Intel i9-9980XE CPU, 128GB RAM and a RTX2080 Ti GPU. Neural networks are implemented in JAX [7]. Given this setup, SNH took up to 20 minutes to train in our experiments, depending on the value of ρ. The average query time of SNH is $329\mu s$ and a model takes 4 MB of space. We publicly release the source code at [44].

Default Values. Unless otherwise stated, we present the results on the *medium* population density city, Milwaukee (VS), with data cardinality $n = 100k$. Privacy budget ε is set to 0.2.

6.2 Comparison with Baselines

Impact of privacy budget. Figs. 6 and 7 present the error of SNH and competitors when varying ε for test datasets VS, SPD-VS, CABS and GW. Recall that a smaller ε means stronger privacy protection.

For our proprietary datasets, VS and SPD-VS, we observe that SNH outperforms the state-of-the-art by up to 50% at all privacy levels (Fig. 6 (a)-(d)). This shows that SNH is effective in utilizing machine learning and publicly available data to improve accuracy of privately releasing proprietary datasets. Fig. 6 (e) and Fig. 7 show

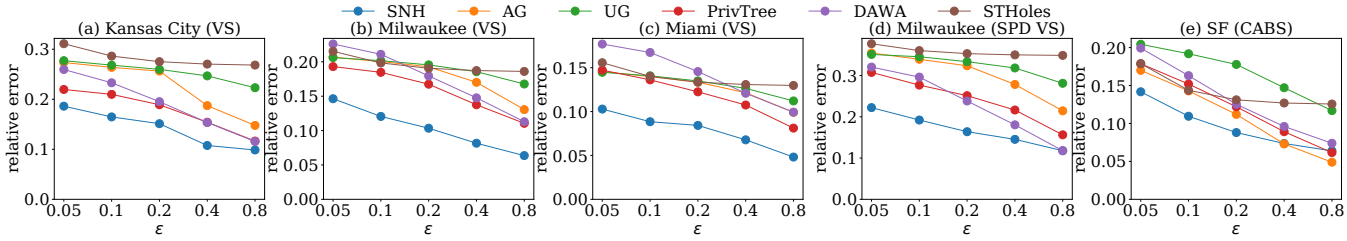

Figure 6: Impact of privacy budget: VS, SPD-VS and CABS datasets

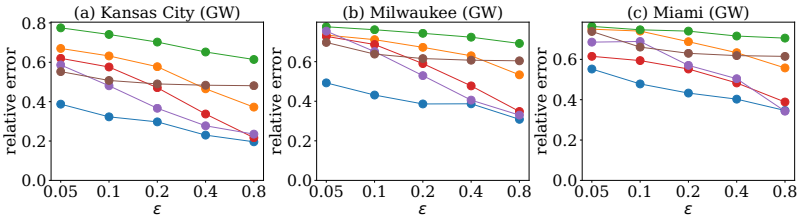

Figure 7: Impact of privacy budget: GW dataset

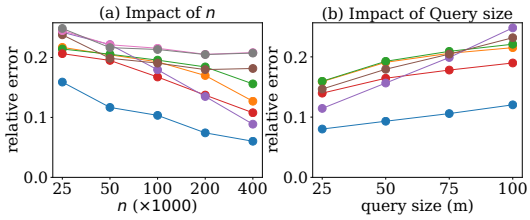

Figure 8: Impact of data and query size

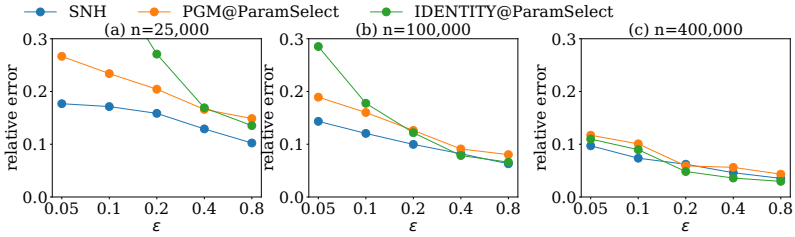

Figure 9: Study of modeling choice

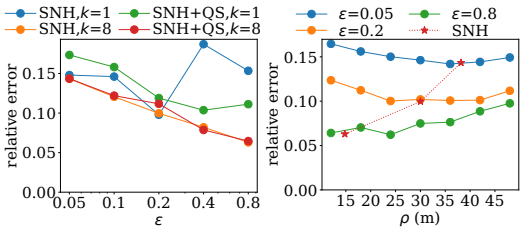

Figure 10: Impact of uniformity assumption

Figure 11: Impact of ρ and ParamSelect

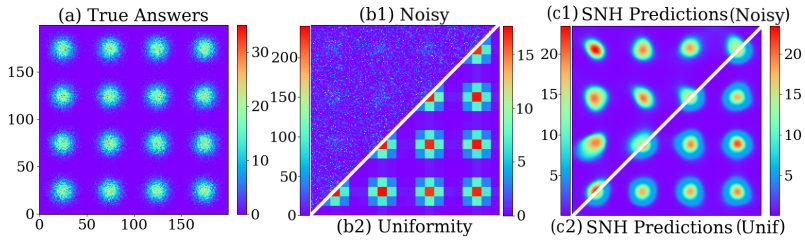

Figure 12: SNH learns patterns on GMM dataset of 16 components. Color shows number of data points.

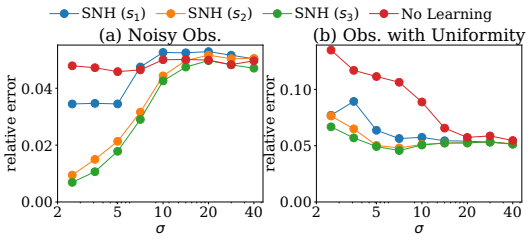

Figure 13: Impact of data skewness ($\varepsilon = 0.2$)

that SNH also outperforms for CABS and GW datasets in almost all settings, the advantage of SNH being more pronounced for smaller ε values. Stricter privacy regimes are particularly important for location data, since such datasets are often released at multiple time instances with smaller privacy budget per release.

Impact of data cardinality. Fig. 8 (a) shows the impact of data cardinality on relative error for Milwaukee (VS). For all algorithms, the accuracy improves as data cardinality increases. This is a direct consequence of the signal-to-noise ratio improving as cell counts are less impacted by DP noise. SNH consistently outperforms competitor approaches at a wide range of data cardinality settings.

Impact of query size. We evaluate the impact of query size on accuracy by considering test queries of four different sizes in Milwaukee (VS). Fig. 8 (b) shows that the error for all the algorithms increases when query size grows, with SNH outperforming the baselines at all sizes. There are two competing effects when increasing query size: on the one hand, each query is less affected by noise, since actual counts are larger; on the other hand, the error from more grid cells is aggregated in a single answer. The second effect is stronger, so the overall error steadily increases with query size.

6.3 Ablation Study for SNH

6.3.1 Modeling choices. Recall that SNH first creates a uniform grid, with granularity decided by ParamSelect. It then performs data augmentation and learning using the data collected on top of the grid. Next, we study the importance of each component of SNH to its overall performance. We create two new baselines to show how our choice of using neural networks to learn the patterns in the data improves performance. The first, called IDENTITY@ParamSelect, ablates SNH, utilizing only the uniform grid created in SNH at data collection. The second baseline, called PGM@ParamSelect, employs Private Probabilistic Graph Models (PGM) [29], a learning algorithm specifically designed for high-dimensional categorical data. We extend PGM to 2D spatial datasets by feeding it a DP uniform grid at the granularity selected by ParamSelect.

Fig. 9 (a) shows SNH outperforming both these baselines. SNH outperforming IDENTITY shows the benefit of learning, since both SNH and IDENTITY use the same grid for data collection but SNH learns neural networks using data generated from the grid, while IDENTITY directly uses the grid to answer queries. This benefit diminishes when the privacy budget and the data cardinality increase (note that both n and ε are in log scale), where a simple uniform grid chosen at the *correct* granularity outperforms all existing methods (comparing Fig. 9 (b) with Fig. 6 (b) shows IDENTITY@ParamSelect outperforms the state-of-the-art for $\varepsilon = 0.4$ and 0.8). For such ranges of privacy budget and data cardinality, ParamSelect recommends a very fine grid granularity. Thus, the uniformity error incurred by IDENTITY@ParamSelect becomes lower than that introduced by the modelling choices of SNH and PGM. This also shows the importance of a good granularity selection algorithm, as UG in Fig. 6 performs worse than IDENTITY@ParamSelect for larger ε.

6.3.2 Balancing Uniformity Errors. We discuss how the use of the uniformity assumption at different stages of SNH impacts accuracy. Recall from Sec. 4.3 that the value of k balances the use of the uniformity assumption pre- and post-learning. We empirically study how uniformity assumption pre- and post-learning influence SNH's accuracy by varying k. Furthermore, we study how removing the uniformity assumption post-learning and replacing it with a neural network affects accuracy. Specifically, we consider a variant of SNH where we train the neural networks to also take as an input the query size. Each neural network is still responsible for a particular set of query sizes, $[r_l, r_u]$, where we use data augmentation to create query samples with different query sizes falling in $[r_l, r_u]$. Instead of scaling the output of the trained neural networks, now each neural network also takes the query size as an input, and thus, the answer to a query is just the forward pass of the neural network. We call this variant *SNH with query size*, or SNH+QS .

Fig. 10 shows that, first, removing the uniformity assumption post-learning has almost no impact on accuracy when k is large. However, for a small value of k, it provides more stable accuracy. Note that when $k = 1$, SNH trains only one neural network for query size r^* and answers the queries of size r by scaling the output of the neural network by $\frac{r}{r^*}$. The error is expected to be lower when ρ and r^* have similar values, since there will be less uniformity error when performing data augmentation. This aspect is captured in Fig. 10, where at $\varepsilon = 0.2$, r^* and ρ are almost the same values and thus, the error is the lowest. Sec. B.4 of our technical report [45] evaluates more comprehensively the impact of k.

6.3.3 ParamSelect and ρ. Fig.11 shows the performance of SNH with varying cell width ρ at multiple values of ε. A coarser grid first improves accuracy by improving signal-to-noise ratio at each cell, but a grid too coarse hampers accuracy by reducing the number of samples extracted for training SNH. This creates a U-shaped trend which shifts to smaller values of ρ for larger values of ε as the lower DP noise impacts the cell counts less aggressively. The red line in Fig.11 labelled SNH shows the result of SNH at the granularity chosen by ParamSelect. SNH performing close to the best possible proves that ParamSelect finds an advantageous cell width for SNH.

6.3.4 SNH Learning Ability in Non-Uniform Datasets. We study the ability of neural networks to learn patterns from skewed datasets through imprecise observations, where imprecision is due to noise or uniformity assumption.

Setup. We synthesize $100k$ points from a Gaussian Mixture Model (GMM) [36] with 16 components. The means of the components are placed uniformly over the data space. All components are equally weighted and have the covariance matrix $\boldsymbol{I} \times \sigma^2$, where $\boldsymbol{I}$ is the identity matrix. GMMs allow controlling data skewness via the parameter σ. We partition the data space into a grid of 200×200 cells and report σ in terms of number of cells. The query set, Q, consists of queries asking for the number of points inside each cell. Fig. 12(a) plots the true answers to this query set when $\sigma = 7$.

Learning from Noisy Observations. We consider two scenarios. First, we obtain the DP answers, $\tilde{A}$, to the queries in Q by adding noise to the true answers. We call this algorithm *No Learning*. For $\varepsilon = 0.05$, Fig. 12 (b1) shows the noisy answers reported by No Learning. Comparing Figs. 12 (a) and 12 (b1) we observe that the sanitization noise severely distorts the existing patterns in the data. Second, we train a neural network using only the noisy answers shown in Fig. 12 (b1), that is, the inputs to the neural network are queries in Q and training labels are the answers in $\tilde{A}$. After training, we ask the same queries, Q. The result in Fig. 12 (c1) shows the output of the neural network. SNH has a strong ability to recover the underlying patterns of GMMs from even highly distorted observations. Additional visualizations for several values of ε and σ can be found in Sec. B.5 of our technical report [45].

Next, we compare the error in the neural network predictions to that in the noisy answers it was trained with. The latter is represented with the line labelled 'No Learning' in Fig. 13 (a) and is the error in $\tilde{A}$. Lines labeled SNH show the error of SNH at varying model sizes (s_1, s_2 and s_3 correspond to models with depth 5, 10 and 20 and width 15, 25 and 80 respectively) on the same query set. When σ is large, the data is closer to being uniformly distributed and there are fewer patterns to learn, whereas when σ is small, the data becomes more skewed towards the mean of each GMM component. The results in Fig. 13 (a) show that when data is skewed, SNH is especially capable of extracting patterns in the data where present, utilizing them to boost accuracy. However, when data is uniform-like, SNH performs similar to 'No Learning' as there are few patterns to be learned. Lastly, by varying model size (lines s_1,

s_2 and s_3) we show that it is beneficial to use a larger neural network for more skewed datasets. A larger network exhibits stronger representation power and hence captures the skewness better.

Learning from Observations with Uniformity Error. We generate the training data by purposefully inducing uniformity error when answering queries in our training set, Q. We first superimpose a coarse partitioning of 20×20 blocks over the original 200×200 cell grid, with each block covering exactly 100 cells. To answer the queries in Q, we first obtain the true answer for each block, and then divide that value by 100 to obtain the answer for each cell within the block (assuming uniformity within the block). The result is shown in Fig. 12 (b2). Note that the set of queries that fall within the same block (in the 20×20 grid) all receive the same answers due to the uniformity assumption. Next, we train a neural network with queries in Q (corresponding to the cells in the 200×200 grid). The result in Fig. 12 (c2) shows that the neural network smoothens the observations and brings them closer to the true answers. In Fig. 13 (b) we evaluate the effect of increasing skewness (i.e., decreasing σ): "No Learning" yields larger errors, whereas SNH, through learning, keeps the error steady for different skewness levels.

7 RELATED WORK

Privacy preserving machine learning. A learned model can leak information about the data it was trained on [20, 38]. Recent efforts have developed differentially private versions of ML algorithms, e.g., empirical risk minimization [9, 22] and deep neural networks [4, 37]. For DP sanitization, existing approaches add noise to the output of the trained model [39], add a random regularization term to the objective function [9, 22], or add noise to the gradient of the loss function during training [4]. Our approach is different in that we sanitize the training data *before* learning. Furthermore, the work of [4] achieves (ε, δ)-DP [5, 14, 31], a weaker privacy guarantee.

Answering RCQs. In the one dimensional case, the data-independent Hierarchical method [18] uses a strategy consisting of hierarchically structured range queries typically arranged as a tree. Similar methods (e.g., HB [35]) differ in their approach to determining the tree's branching factor and allocating appropriate budget to each of its levels. Data-dependent techniques, on the other hand, exploit the redundancy in real-world datasets to boost the accuracy of histograms. The main idea is to first lossily compress the data. For example, EFPA [6] applies the Discrete Fourier Transform whereas DAWA [25] uses dynamic programming to compute the least cost partitioning. The compressed data is then sanitized, for example, directly with Laplace noise [6] or with a greedy algorithm that tunes the privacy budget to an expected workload [25]).

While some approaches such as DAWA and HB extend to 2D naturally, others specialize to answer spatial range queries. Uniform Grid (UG) [34] partitions the domain into a $m \times m$ grid and releases a noisy count for each cell. The value of m is chosen in a data-dependent way, based on dataset cardinality. Adaptive Grid (AG) [34] builds a two-level hierarchy: the top-level partitioning utilizes a granularity coarser than UG. For each bucket of the top-level partition, a second partition is chosen in a data-adaptive way, using a finer granularity for regions with a larger count. QuadTree [11] first generates a quadtree, and then employs the Laplace mechanism to inject noise into the point count of each node. Range-count queries are answered via a top-down traversal of the tree. Privtree [48] is another hierarchical method that allows variable node depth in the indexing tree (as opposed to fixed tree heights in AG, QuadTree and HB). It utilizes the Sparse-Vector Technique [26] to determine a cell's density prior to splitting the node.

The case of high-dimensional data was addressed by [28, 40, 47]. The most accurate algorithm in this class is High-Dimensional Matrix Mechanism (HDMM) [28] which represents queries and data as vectors, and uses optimization and inference techniques to answer RCQs. PrivBayes [47] is a mechanism that privately learns a Bayesian network over the data that generates a synthetic dataset which can consistently answer workload queries. Due to the use of sampling to estimate data distribution, it is a poor fit for skewed spatial datasets. Most similar to our work is PGM [29], which utilizes Probabilistic Graphical Models to measure a compact representation of the data distribution, while minimizing a loss function. Data projections over user-specified subgroups of attributes are sanitized and used to learn the model parameters. PGM is best used in the inference stage of privacy mechanisms (such as HDMM and PrivBayes) that can already capture a good model of the data.

Private parameter tuning. Determining the system parameters of a private data representation must also be DP-compliant. Several approaches utilize the data themselves to tune system parameters such as depth of a hierarchical structure (e.g., in QuadTree or HB) or spatial partition size (e.g. k-d trees), without privacy consideration [18]. Using *public* datasets to tune system parameters is a better strategy [9]. Our strategy to determine a good cell width for a differentially-private grid is similar to that in UG [34]. However, our proposed strategy for parameter selection vastly improves generalization ability over UG [34] by exploiting additional dataset features and their non-linear relationships.

8 CONCLUSION

We proposed SNH: a novel method for answering range count queries on location datasets while preserving differential privacy. To address the shortcomings of existing methods (i.e., over-reliance on the uniformity assumption and noisy local information when answering queries), SNH utilizes the power of neural networks to learn patterns from location datasets. We proposed a two stage learning process: first, noisy training data is collected from the database while preserving differential privacy; second, models are trained using this sanitized dataset, after a data augmentation step. In addition, we devised effective machine learning strategies for tuning system parameters using only *public* data. Our results show SNH outperforms the state-of-the-art on a broad set of input data with diverse characteristics. In future work, we plan to extend SNH to releasing high data dimensional user trajectories datasets.

ACKNOWLEDGMENTS

This research has been funded in part by NSF grants IIS-1910950, IIS-1909806, CNS-2027794, CNS-2125530 and IIS-2128661, and an unrestricted cash gift from Microsoft Research. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of the sponsors such as the NSF.

REFERENCES

[1] 2021. List of United States cities by population. https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population. Accessed July 2021.

[2] 2021. Veraset. https://www.veraset.com/about-veraset. Accessed: 2021-05-10.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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

[8] Nicolas Bruno, Surajit Chaudhuri, and Luis Gravano. 2001. STHoles: A Multidimensional Workload-Aware Histogram. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, New York, NY, USA, 211–222.

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

[10] 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.

[11] 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.

[12] Bolin Ding, Marianne Winslett, Jiawei Han, and Zhenhui Li. 2011. Differentially private data cubes: optimizing noise sources and consistency. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 217–228.

[13] 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.

[14] 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.

[15] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2020. Auto-sklearn 2.0: The next generation. *arXiv preprint arXiv:2007.04074* 24 (2020).

[16] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine learning* 63, 1 (2006), 3–42.

[17] 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.

[18] 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).

[19] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2019. DeepDB: Learn from Data, not from Queries! *Proceedings of the VLDB Endowment* 13, 7 (2019).

[20] 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.

[21] Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, Vol. 1. IEEE, 278–282.

[22] 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.

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

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012), 1097–1105.

[25] 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.

[26] Min Lyu, Dong Su, and Ninghui Li. 2017. Understanding the Sparse Vector Technique for Differential Privacy. *Proc. VLDB Endow.* 10, 6 (Feb. 2017), 637–648.

[27] Qingzhi Ma and Peter Triantafillou. 2019. Dbest: Revisiting approximate query processing engines with machine learning models. In *Proceedings of the 2019 International Conference on Management of Data*. 1553–1570.

[28] 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).

[29] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*. PMLR, 4435–4444.

[30] Henry B Moss, David S Leslie, and Paul Rayson. 2018. Using JK fold cross validation to reduce variance when tuning NLP models. *arXiv preprint arXiv:1806.07139* (2018).

[31] 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.

[32] Rafael Pérez-Torres, César Torres-Huitzil, and Hiram Galeana-Zapién. 2016. Full on-device stay points detection in smartphones for location-based mobile applications. *Sensors* 16, 10 (2016), 1693.

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

[34] 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.

[35] 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.

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

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

[38] 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.

[39] 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.

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

[41] Anatoly Yakovlev, Hesam Fathi Moghadam, Ali Moharrer, Jingxiao Cai, Nikan Chavoshi, Venkatanathan Varadarajan, Sandeep R Agrawal, Sam Idicula, Tomas Karnagel, Sanjay Jinturkar, et al. 2020. Oracle automl: a fast and predictive automl pipeline. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3166–3180.

[42] Yang Ye, Yu Zheng, Yukun Chen, Jianhua Feng, and Xing Xie. 2009. Mining individual life pattern based on location history. In *2009 tenth international conference on mobile data management: Systems, services and middleware*. IEEE, 1–10.

[43] Sepanta Zeighami, Ritesh Ahuja, Gabriel Ghinita, and Cyrus Shahabi. 2021. Private STHoles Implementation. https://github.com/szeighami/stholes.

[44] Sepanta Zeighami, Ritesh Ahuja, Gabriel Ghinita, and Cyrus Shahabi. 2021. SNH Implementation. https://github.com/szeighami/snh.

[45] Sepanta Zeighami, Ritesh Ahuja, Gabriel Ghinita, and Cyrus Shahabi. 2021. SNH Technical Report. https://infolab.usc.edu/DocsDemos/snh.pdf.

[46] Sepanta Zeighami and Cyrus Shahabi. 2021. NeuroDB: A Neural Network Framework for Answering Range Aggregate Queries and Beyond. *arXiv preprint arXiv:2107.04922* (2021).

[47] 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.

[48] 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.

[49] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2020. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 13001–13008.