ON DIFFERENTIAL PRIVATE ℓ_1, ℓ_2 AND ℓ_p^p DISTANCE QUERIES

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a refined differentially private (DP) data structure for kernel density estimation (KDE) with ℓ_1, ℓ_2 and ℓ_p^p distance kernels. This new DP data structure offers not only improved privacy-utility tradeoff but also better query efficiency over prior results. Specifically, we study the mathematical problem: given a similarity function f (or DP KDE) and a private dataset $X \subset \mathbb{R}^d$, our goal is to preprocess X so that for any query $y \in \mathbb{R}^d$, we approximate $\sum_{x \in X} f(x, y)$ in a differentially private fashion. The best previous algorithm for $f(x, y) = \|x - y\|_1$ is the node-contaminated balanced binary tree by [Backurs, Lin, Mahabadi, Silwal, and Tarnawski, ICLR 2024]. Their algorithm requires $O(nd)$ space and time for preprocessing with $n = |X|$. For any query point, the query time is $\alpha^{-1} d \log^2 n$, with an multiplicative error guarantee of $(1 + \alpha)$-approximation and an additive error guarantee of $O(\epsilon^{-1} \alpha^{-0.5} d^{1.5} R \log^{1.5} n)$.

In this paper, we use the same space and pre-processing time, improve the best previous result [Backurs, Lin, Mahabadi, Silwal, and Tarnawski, ICLR 2024] in three aspects

- We reduce query time by $\alpha^{-1} \log n$ factor
- We improve the approximation ratio from $1 + \alpha$ to 1
- We reduce the error dependence by a factor of $\alpha^{-0.5}$

From a technical perspective, our method of constructing the search tree differs from previous work [Backurs, Lin, Mahabadi, Silwal, and Tarnawski, ICLR 2024]. In prior work, for each query, the answer is split into $\alpha^{-1} \log n$ numbers, each derived from the summation of $\log n$ values in interval tree countings. In contrast, we construct the tree differently, splitting the answer into $\log n$ numbers, where each is a smart combination of two distance values, two counting values, and y itself. We believe our tree structure may be of independent interest.

1 INTRODUCTION

We propose a refined differentially private (DP) data structure for DP kernel density estimation, offering improved privacy-utility tradeoff over prior results without compromising efficiency. Let $X \subset \mathbb{R}^d$ be a private dataset, and let $f(x, y) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a similarity function[1], such as a kernel or distance function, between a user query $y \in \mathbb{R}^d$ and a private data point $x \in X$.

Problem 1.1 (DP Kernel Density Estimation Query)**.** *The DP Kernel Density Estimation (KDE) query problem aims for an algorithm outputting a private data structure $D_X : \mathbb{R}^d \to \mathbb{R}$, that approximates the map $y \mapsto \sum_{x \in X} f(x, y)$ in a DP fashion[2]. Especially, we require D_X to remain private with respect to X, regardless of the number of queries.*

Problem 1.1 is well-studied (Hall et al., 2013; Huang & Roth, 2014; Wang et al., 2016; Aldà & Rubinstein, 2017; Alman et al., 2020; Coleman & Shrivastava, 2021; Aggarwal & Alman, 2022; Qin et al., 2022; Alman & Song, 2023; Gao et al., 2023; Wagner et al., 2023; Alman & Song, 2024; Hu et al., 2024; Li et al., 2024a; Backurs et al., 2024) due to its broad applicability and its importance in productizing large foundation models (Lin et al., 2024; Xie et al., 2024a). What makes this problem interesting is its generic abstraction of many practical DP challenges in modern machine learning.

[1] In this work, we use "kernel query" and "distance query" interchangeably.

[2] For a given dataset X and a given query y, an KDE query computes an approximation to $\sum_{x \in X} f(x, y)$.

These include generating synthetic data similar to a private dataset (Lin et al., 2020; Li et al., 2022; Yu et al., 2022; Yin et al., 2022), and selecting a similar public dataset for pre-training ML models (Hou et al., 2023; Yu et al., 2024a; Yue et al., 2023). Such problem becomes more prevalent in this era of large foundation models (Lin et al., 2024; Xie et al., 2024a). Essentially, these problems involve computing the similarity between a private dataset (i.e., X) and a processed data point (i.e., query y), thus falling under Problem 1.1.

In this work, we focus on the ℓ_1 kernel[3] (i.e., $f(x, y) = \|x-y\|_1$). By far, the algorithm with the best privacy-utility tradeoff and query efficiency for Problem 1.1 is by Backurs, Lin, Mahabadi, Silwal, and Tarnawski (Backurs et al., 2024). They propose a DP data structure via a node-contaminated balanced binary tree for ℓ_1 kernel. To be concrete, we begin by defining the similarity error between two data structures and then present their results.

Definition 1.2 (Similarity Error between Two Data Structures)**.** *For a fixed query, let A represent the value output by our private data structure, and let A' represent the true value. We say that A has an error of (M, Z) for $M \geq 1$ and $Z \geq 0$, if $\mathbb{E}[|A - A'|] \leq (M-1)A' + Z$. This implies a relative error of $M - 1$ and an additive error of Z. The expectation is taken over the randomness used by the data structure.*

Let $n := |X|$ be the size of the dataset, $X_{i,j}$ be the j-th dimension of the i-th data point, $R := \max_{i,j}(X_{i,j})$ be the max value of each data entries from each dimension, and $\alpha \in [0, 1]$ be a parameter of the data structure selected before calculation. For ℓ_1 kernel, Backurs et al. (2024) give the error bound $(1 + \alpha, \alpha^{-0.5}\epsilon^{-1}Rd^{1.5}\log^{1.5} n)$ through their DP data structure.

Compared to the results of (Backurs et al., 2024), we provide a refined DP data structure and improves both privacy-utility tradeoff and query efficiency. Specifically, we not only improve the ℓ_1 error bound to $(1, \epsilon^{-1}Rd^{1.5}\log^{1.5} n)$, but also the query time from $O(\alpha^{-1}d\log^2 n)$ to $O(d\log n)$. This makes our new DP data structure the latest best algorithm for Problem 1.1:

Theorem 1.3 (Informal Version of Theorem 3.11)**.** *Given a dataset $X \subset \mathbb{R}^d$ with $|X| = n$. There is an algorithm that uses $O(nd)$ space to build a data-structure which supports the following operations:*

- **INIT**(X, ϵ)**.** *It takes dataset X and privacy parameter ϵ as input and spends $O(nd)$ time to build a data-structure.*
- **QUERY**$(y \in \mathbb{R}^d)$**.** *It takes y as input and spends $O(d\log n)$ time to output a scalar A such that*
$$\mathbb{E}[|A - A'|] \leq O(\epsilon^{-1}Rd^{1.5}\log^{1.5} n).$$

Notably, our improvements stem from a key observation that — in the balanced binary tree data structure proposed by Backurs et al. (2024), each query answer is split into $\alpha^{-1}\log n$ values, derived from the summation of $\log n$ values in interval tree counts. We deem that this splitting is not essential and can be achieved through simple and efficient preprocessing steps.

To this end, we introduce a novel data representation in the tree nodes, where each node stores the sum of distances from one point to multiple points. This allows us to split the answer into only $\log n$ values, each being a smart combination of two distance values, two count values, and y itself. This additional information enhances each query calculation, leading to an improved privacy-utility tradeoff (Theorem 3.11 and Lemma 3.5) and faster query time (Lemma 3.3).

Lastly, we remark that our ℓ_1 results are transferable to other kernels via the provably dimensional reduction recipe of (Backurs et al., 2024). To showcase this, we generalize our results to ℓ_2 and ℓ_p^p kernels following (Backurs et al., 2024). Notably, these results also improve upon the best previous results reported in (Backurs et al., 2024) for ℓ_2 and ℓ_p^p with $p \in [1, n]$. For details, see Appendix E and F, particularly the comparison Tables 2 and 3.

Related Work. We defer related work discussion to Appendix B due to page limits.

Organization. Section 2 includes the preliminaries, summarizes the prior best DP data structure by (Backurs et al., 2024), and provides a high-level overview of our main results. Section 3 presents

[3]We remark that ℓ_1 query distance is fundamental. Namely, it is easy to generalize the ℓ_1 results to a wide range of kernels and distance functions following the provably dimensional reduction recipe of (Backurs et al., 2024). See Appendix E and F for improved DP KDE results with ℓ_2 and ℓ_p^p kernels.

our new data structure for DP ℓ_1 kernel estimation along with its theoretical analysis. Appendices E and F extends the ℓ_1 results to ℓ_2 and ℓ_p^p kernels. Appendix A includes proof-of-concept experiments. Lastly, Section 4 concludes the paper.

References	Approx. Ratio	Error	Query Time	Init time	Space
(Backurs et al., 2024)	$1+\alpha$	$\alpha^{-0.5}\epsilon^{-1}Rd^{1.5}\log^{1.5} n$	$\alpha^{-1}d\log^2 n$	nd	nd
Theorem 1.3	1	$\epsilon^{-1}Rd^{1.5}\log^{1.5} n$	$d\log n$	nd	nd

Table 1: Comparison of $\|x-y\|_1$ Results with Best Known Algorithm by Backurs et al. (2024). We ignore the big-O notation in the table, for simplicity of presentation. We use n to denote the number of points in dataset. We use d to denote the dimension for each point in the dataset. We assume all the points are bounded by R. We use the ϵ to denote ϵ-DP.

2 PRELIMINARIES

Notation. For a positive integer n, we use $[n]$ to denote $\{1, 2, \cdots, n\}$. For a vector x, we use $\|x\|_1 := \sum_{i=1}^n |x_i|$ to denote its ℓ_1-norm. We use $\mathbb{E}[\cdot]$ to denote the expectation. We use $\mathrm{Var}[\cdot]$ to denote the variance. We use $\Pr[\cdot]$ to denote the probability. We use $\mathsf{Laplace}(\lambda)$ random variable with parameter λ. It is known that $\mathbb{E}[\mathsf{Laplace}(\lambda)] = 0$ and $\mathrm{Var}[\mathsf{Laplace}(\lambda)] = 2\lambda^2$.

2.1 DIFFERENTIAL PRIVACY

For completeness, we state the definitions of differential privacy (Dwork et al., 2006; 2010b).

Definition 2.1 (Pure/Approximate Differential Privacy)**.** *A randomized algorithm $M : \mathcal{X}^n \to \mathcal{Y}$ satisfies (ϵ, δ)-differential privacy if, for all $x, x' \in \mathcal{X}^n$ differing on a single element and for all events $E \subset \mathcal{Y}$, we have* $\Pr[M(x) \in E] \leq e^\epsilon \cdot \Pr[M(x') \in E] + \delta$

For special case of (ϵ, 0)-differential privacy, we use pure or pointwise ϵ-differential privacy to denote it. For the other case of $\delta > 0$, we denote it as approximate differential privacy.

Next, we introduce a common tool in DP proofs. For multiple independent DP functions, the next lemma provides an estimation on the privacy of their composition:

Lemma 2.2 (Advanced Composition Starting from Pure DP (Dwork et al., 2010b))**.** *Let $M_1, ..., M_k : X^n \to Y$ be randomized algorithms, each of which is (ϵ, δ)-DP. Define $M : X^n \to Y^k$ by $M(x) = (M_1(x), ..., M_k(x))$ where each algorithm is run independently. Then M is (ϵ', δ)-DP for any $\epsilon, \delta > 0$ and*

$$\epsilon' = k\epsilon^2/2 + \epsilon\sqrt{2k\log(1/\delta)}.$$

For $\delta = 0$, M is $k\epsilon$-DP.

2.2 (BACKURS ET AL., 2024)'S DP DATA STRUCTURE: NODE-CONTAMINATED BALANCED TREE

Backurs et al. (2024) propose a data structure for general high-dimensional (e.g., d-dimensional) ℓ_1 kernel via 1-dimensional decomposition:

$$\sum_{x\in X} \|x-y\|_1 = \sum_{i=1}^d \sum_{x\in X} |x_i - y_i|.$$

Namely, they create a DP data structure for a d-dimensional dataset X by considering d copies 1-dimensional DP data structures (i.e., $\sum_{x\in X} |x_i - y_i|$ for $i \in [d]$). For each of these 1-dimensional DP data structures, they employ a node-contaminated balanced tree algorithm as follows.

Let $n = |X|$ be the size of the dataset X. Considering all input values are integer multiples of R/n in $[0, R]$,[4]

1. They use a classic balance binary tree of L layers in one dimension: a binary tree where each non-leaf node has exactly two children, and all leaf nodes are at the same depth, L. The tree has $2^L - 1$ total nodes, with 2^{L-1} leaf nodes and L levels from the root (level 1) to the leaves (level L). Each level k contains 2^{k-1} nodes.

[4]For general continuous data, they achieve this by rounding all data points to some integer multiples of R/n.

2. They assign the leaves to correspond to these multiples (of R/n in $[0, R]$) and store the number of dataset points at each position, while internal nodes store the sum of their children.
3. It's well-known that any interval query can be answered by summing values from $O(\log n)$ tree nodes. This provide an efficient (*noise-less*) query operation.
4. To ensure DP, they propose a noisy version of the tree by injecting noise onto each node, i.e., a *node-contaminated* balanced tree. By above, changing any data point affects only $O(\log n)$ counts in the tree, each by at most one (from leaf to root), which bounds the sensitivity of the data structure.

To highlight the significance of this work — removing the α dependence in the result and eliminating a $\log n$ factor in the query (see Table 1) — we make the following remarks on (Backurs et al., 2024).

- **Query Time.** For each query y, in order to report the answer, they need to take summation of $O(\alpha^{-1} \log n)$ numbers where each number from each region $(y+(1+\alpha)^i, y+(1+\alpha)^{i+1}]$ (see line 7 in Algorithm 3 in (Backurs et al., 2024)). In particular, for each region, they need to run a interval query which consists of $\log n$ numbers (each number is from a tree node, see Algorithm 2 in (Backurs et al., 2024)). That's why the their algorithm has $\alpha^{-1} \log^2 n$ dependence.
- **Error Bound and Accuracy.** Due to the region $(y + (1 + \alpha)^i, y + (1 + \alpha)^{i+1}]$ can only approximate the number within $(1 + \alpha)$ approximation. That is why the final error is proportional to α^{-1} and accuracy is losing a $(1 + \alpha)$ relative error. Since the values are in range $[0, R]$, thus the error bound will be linearly depend on R according to how they construct the tree and alayze the error.
- **Sensitivity and DP Guarantee.** Since the tree is analyzing the counting problem, thus the sensitivity is $O(\log n)$ because change one point, will affect $O(\log n)$-levels stored counts.

2.3 High-Level Overview of Our DP Data Structure

We improve (Backurs et al., 2024) by providing a refined balanced tree for DP 1-dimensional ℓ_1 distance query. Specifically, we introduce a new data representation on each node of the balanced binary tree as follows.

Let $n = |X|$ be the size of the dataset X and $\{x_k\} \in [0, R)$ for $k \in [n]$. (Note that unlike (Backurs et al., 2024), we don't assume x_k is a multiple of R/n, because our improved algorithm is compatible with floating-point values of x_k.)

1. We use a balanced binary tree where in each node stores both distance s and the counts c. Here, s denotes the pre-summation of certain distances, and c denotes the pre-counting.
2. We make a key observation (Lemma 3.1) that these values (s and c) are not depending on query y, while they are critical components of the final answer (the ℓ_1 kernel $\sum_{x \in X} \|x - y\|_1$). This motivate us to construct the answer by using $L = \log n$ layers result, where each layer result is a smart combination of y, s_1, s_2, c_1, c_2 (i.e., $(s_1 - s_2 + y \cdot c_1 - y \cdot c_2)$).
3. To ensure DP, we need a noise version of the tree, due to s and c have different sensitivities, thus we need to add different levels of noise for s and c.

A few remarks are in order:

- **Query Time.** For each query y, to report the answer, we first extract four $L = O(\log n)$ numbers along with y and apply a special function: $(s_1 - s_2 + y \cdot (c_1 - c_2))$ (see our key observation in Lemma 3.1).
- **Error Bound and Accuracy.** Since our algorithm does not use the $(1+\alpha)^i$ region concept, we avoid the α^{-1} error. As a result, our error is purely additive, with no relative error.
- **Sensitivity and DP Guarantee.** Since s approximates the summation of distances, the sensitivity is $O(LR)$, where each node in one of the L layers contributes a factor of R. On the other hand, since c counts the points, its sensitivity is $O(L)$, with each node in the L layers contributing an $O(1)$ factor. Previous work by Backurs et al. (2024) only use y to determine which region to count points, but our algorithm uses y to construct the final answer. Since $y \leq R$, this explains the R gap between the noise levels in s and c.

3 A Refined Differentially Private Data Structure

We propose a new data structure with ϵ-differential privacy that achieves $O(\epsilon^{-1}R\log^{1.5} n)$ additive error. Our method improves the balanced binary tree structure proposed by Backur, Lin, Mahabadi, Silawal, and Tarnawsk (Backurs et al., 2024) in both DP guarantee and efficiency. Our data structure adapts a new data representation on each node that simplifies each query without losing privacy guarantees.

3.1 Key Observation and New Data Structure for One Dimensional ℓ_1 Distance Query

For simplicity, we consider the 1-dimensional distance query problem with ℓ_1 kernel distance. Note that any high-dimensional distance query problem can be reduced to this case via the 1-dimensional decomposition discussed in Section 2.2 or in (Backurs et al., 2024).

When doing queries, we observe that

Lemma 3.1. *For a collection of values $\{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}$ and a value y, we define two sets*

$$S_+ := \{k \in [n] \ : \ x_k > y\}$$
$$S_- := \{k \in [n] \ : \ x_k < y\}.$$

It holds

$$\sum_{k=1}^{n} |x_k - y| = \underbrace{(\sum_{k\in S_+} x_k)}_{s_{\mathrm{right}}} - \underbrace{(\sum_{k\in S_-} x_k)}_{s_{\mathrm{left}}} + y \cdot \underbrace{|S_-|}_{c_{\mathrm{left}}} - y \cdot \underbrace{|S_+|}_{c_{\mathrm{right}}}.$$

Proof.

$$\begin{aligned}
&\sum_{k=1}^{n} |x_k - y| \\
=& \sum_{k\in S_+} (x_k - y) + \sum_{k\in S_-} (y - x_k) \\
=& (\sum_{k\in S_+} x_k) - (\sum_{k\in S_-} x_k) + y \cdot (\sum_{k\in S_-} 1) - y \cdot (\sum_{k\in S_+} 1) \\
=& (\sum_{k\in S_+} x_k) - (\sum_{k\in S_-} x_k) + y \cdot |S_-| - y \cdot |S_+|.
\end{aligned}$$

This completes the proof. □

Lemma 3.1 provides a neat decomposition of the ℓ_1 distance query into four components. This motivates us to design a new DP data structure that pre-computes these terms and storing them in a balanced tree for possible algorithmic speedup. Surprisingly, this preprocessing not only accelerates query time but also improves the privacy-utility tradeoff. The following discussion illustrates this.

To calculate each part efficiently, we define our data representation on a balanced binary tree as follows:

- **Dataset**: Given a dataset $X := \{x_k\}_{k=1}^n$ containing n values in the range $[0, R)$, we build a balanced binary tree using X.
- **Tree Structure**:
  - Let L denote the total number of layers in the tree.
  - At the l-th layer, there are exactly 2^l nodes.
- **Node Representation**:
  - Each node represents a consecutive interval of X.
  - The interval for the j-th node at the l-th layer is defined as $I_{l,j} := [(j-1) \cdot \frac{R}{2^{l-1}}, j \cdot \frac{R}{2^{l-1}})$.
  - The left and right children of $I_{l,j}$ are $I_{l+1,2j-1}$ and $I_{l+1,2j}$, respectively.

Each node $I_{l,j}$ stores two values: $c_{l,j}$ and $s_{l,j}$. Here:

- $c_{l,j} := |\{x_k : x_k \in I_{l,j}\}|$ represents the count of data points in the interval.

- $s_{l,j} := \sum_{x_k \in I_{l,j}} x_k$ represents the sum of the data points' values in the interval.

After calculating each $c_{l,j}$ and $s_{l,j}$, we add independent noise drawn from $\mathsf{Laplace}(L/\epsilon)$ to each c value and from $\mathsf{Laplace}(LR/\epsilon)$ to each s value. The INIT algorithm (Algorithm 1) outlines the construction of our data structure.

In the QUERY algorithm (Algorithm 2), we sum the s and c values at the relevant nodes. Here:

- s_{left} represents $\sum_{k \in S_-} x_k$
- c_{left} represents $|S_-|$
- s_{right} represents $\sum_{k \in S_+} x_k$
- c_{right} represents $|S_+|$

For each query, we first locate the leaf that y belongs to, then traverse the tree from the bottom up, summing the values of the left and right nodes.

Next, we prove the time complexity, differential privacy property and error of results of out algorithm respectively.

Algorithm 1 Building Binary Tree

data structure FASTERTREE ▷ Theorem 3.8
members
  $T := \{c, s\}$ ▷ $c_{i,j}$ and $s_{i,j}$ represents the count and sum of node defined above
end members
procedure INIT(X, n, ϵ, L) ▷ Lemma 3.2, Lemma 3.5
  for each x_k in X **do**
    find the leaf node j of layer L that x_k is in the interval $I_{L,j} = [(j-1) \cdot \frac{R}{2^{L-1}}, j \cdot \frac{R}{2^{L-1}})$
    $c_{L,j} \leftarrow c_{L,j} + 1$
    $s_{L,j} \leftarrow s_{L,j} + x_k$
  end for
  for each layer l from $L-1$ to 1 **do**
    for each node j in layer l **do**
      $c_{l,j} \leftarrow c_{l+1,2j-1} + c_{l+1,2j}$
      $s_{l,j} \leftarrow s_{l+1,2j-1} + s_{l+1,2j}$
    end for
  end for
  Add independent noises drawn from $\mathsf{Laplace}(LR/\epsilon)$ to each $s_{l,j}$
  Add independent noises drawn from $\mathsf{Laplace}(L/\epsilon)$ to each $c_{l,j}$
  $T \leftarrow \{c, s\}$
end procedure
end data structure

3.2 TIME COMPLEXITY

Here we compute the init and query time of Algorithm 1 and 2, respectively.

Lemma 3.2 (Init Time)**.** *If the total layer $L = \log(n)$, the running time of* INIT *(Algorithm 1) is $O(n)$.*

Proof. By definition, on l-th layer, there are 2^l nodes. Therefore, the total number of nodes is $\sum_{l=1}^{L} 2^l = 2^{L+1} - 1$. When $L = \log(n)$, the total number of nodes on the tree is $O(n)$. In INIT (Algorithm 1), we iterate over n data points, then iterate all nodes. Therefore the total time complexity of pre-processing is $O(n)$. □

Lemma 3.3 (Query Time)**.** *If the total layer $L = \log(n)$, the time* QUERY *function (Algorithm 2) $O(\log(n))$ time.*

Proof. For each single query, we iterate the node that y belongs to on each layer. The total layer number is $\log(n)$, so the time complexity of each query is $O(\log(n))$. □

Remark 3.4 (Comparing with Prior Work)**.** *For a dataset $X \subset \mathbb{R}^d$ of size $n = |X|$ and a parameter $\alpha \in [0, 1]$ selected before calculation, Lemma 3.3 improves the prior best query time in (Backurs et al., 2024) by a factor of $\alpha^{-1} \log n$.*

Algorithm 2 One Dimensional Distance Query

data structure FASTERTREE ▷ Theorem 3.8
procedure QUERY($y \in \mathbb{R}$) ▷ Lemma 3.3, Lemma 3.6
  $s_{\text{left}}, c_{\text{left}}, s_{\text{right}}, c_{\text{right}} \leftarrow 0$ ▷ As previously defined
  for each layer l from 2 to L **do**
    find the node $j = \lceil \frac{2^{l-1}}{R} y \rceil$ in layer l so that y is in the interval $I_{l,j} = [(j-1) \cdot \frac{R}{2^{l-1}}, j \cdot \frac{R}{2^{l-1}})$
    if $j \mod 2 = 0$ **then** ▷ if node j is the right child of its parent
      $c_{\text{left}} \leftarrow c_{\text{left}} + c_{l,j-1}$
      $s_{\text{left}} \leftarrow s_{\text{left}} + s_{l,j-1}$
    else ▷ if node j is the left child of its parent
      $c_{\text{right}} \leftarrow c_{\text{right}} + c_{l,j+1}$
      $s_{\text{right}} \leftarrow s_{\text{right}} + s_{l,j+1}$
    end if
  end for
  return $s_{\text{left}} - s_{\text{right}} + y \cdot c_{\text{right}} - y \cdot c_{\text{left}}$ ▷ Lemma 3.1
end procedure
end data structure

3.3 PRIVACY GUARANTEES

Here we provide our DP guarantee of our proposed DP data structure (i.e., Algorithm 1).

Lemma 3.5 (Differential Privacy)**.** *The data structure* FASTERTREE *returned by* FASTERTREE.INIT *(Algorithm 1) is ϵ-DP.*

Proof. For binary tree of L layers, there are $2^{L+1} - 1$ nodes. On each node, there are 2 values s and c. We consider c and s separately.

Let the functions $F_c(X)$ and $F_s(X) : [0, R)^n \to \mathbb{R}^{2^{L+1}-1}$ represent the mappings from dataset X to c and s, respectively. Next, we prove that both $F_c(X)$ and $F_s(X)$ are $\epsilon/2$-DP.

When changing each data point, only the nodes containing the point change their values. On each layer, there is at most one such node. Therefore, only $O(L)$ values change.

Sensitivity for Storing Counts. For F_c, each c value changes by at most 1, so the sensitivity is L. Adding coordinate-wise Laplace noise with magnitude $\eta = O(2L/\epsilon)$ suffices to ensure $\epsilon/2$-DP using the standard Laplace mechanism.

Sensitivity for Storing Distances. For F_s, changing one data entry affects at most $O(L)$ values, and each value can be affected by at most R. Therefore, the sensitivity is RL. Adding coordinate-wise Laplace noise with magnitude $\eta = O(2RL/\epsilon)$ ensures $\epsilon/2$-DP.

By Lemma 2.2, the differential privacy parameter ϵ of the tree $T := (F_c(X), F_s(X))$ equals $2 \cdot \epsilon/2 = \epsilon$. This completes the proof. □

3.4 ERROR GUARANTEE

Here we provide the (data-structure) similarity error of our proposed DP data structure.

Lemma 3.6 (Error Guarantee)**.** *Let $X = \{x_i\}_{i \in [n]}$ be a dataset of n one dimensional values $x_i \in [0, R)$. Let $A' = \sum_{i=1}^{n} |x_i - y|$ be the true distance query value. Let A be the output of* QUERY *(Algorithm 2) with $L = \log(n)$. Then we have $\mathbb{E}[|A - A'|] \leq O(\log^{1.5}(n) R/\epsilon)$.*

Proof. We analyze the additive error of our algorithm. We notice that the difference between true distance and our output can be divided into two parts.

Part I. The first part is the data in the leaf node which contains query y. In QUERY(Algorithm 2), we ignore these data points. Let the leaf node j of layer L be the node that query point y belongs to. The error is

$$\sum_{x_k \in [(j-1) \cdot R/2^L, j \cdot R/2^L)} |x_k - y|.$$

Since the distance between each data point and y is no more than the length of the interval $R/2^L$, and their total number is no more than n. When $L = \log(n)$, this error is $O(R)$.

Part II. The second part is the Laplace noises. In our query, we add up c and s from $O(L)$ intervals. In these intervals, each contains a Laplace noise. Therefore, the total error is the sum of these noises:

$$\begin{aligned}\mathbb{E}[|A - A'|] \leq & \mathbb{E}[|\sum_{i=1}^{L} \mathsf{Laplace}(RL/\epsilon) + y \cdot \sum_{i=1}^{L} \mathsf{Laplace}(L/\epsilon)|] \\ \leq & \mathbb{E}[|\sum_{i=1}^{L} \mathsf{Laplace}(RL/\epsilon)|] + |y| \cdot \mathbb{E}[|\sum_{i=1}^{L} \mathsf{Laplace}(L/\epsilon)|],\end{aligned}$$

where the second step follows from triangle inequality.

For a random variable Q with $\mathbb{E}[Q] = 0$, using Fact D.1 we have $\mathbb{E}[|Q|] \leq \sqrt{\mathrm{Var}[Q]}$, thus

$$\begin{aligned}\mathbb{E}[|A - A'|] \leq & (\mathrm{Var}[\sum_{i=1}^{L} \mathsf{Laplace}(RL/\epsilon)])^{1/2} + |y| \cdot (\mathrm{Var}[\sum_{i=1}^{L} \mathsf{Laplace}(L/\epsilon)])^{1/2} \\ \leq & \sqrt{L \cdot \mathrm{Var}[\mathsf{Laplace}(RL/\epsilon)]} + |y| \cdot \sqrt{L \cdot \mathrm{Var}[\mathsf{Laplace}(L/\epsilon)]} \\ = & \sqrt{L \cdot 2R^2L^2/\epsilon^2} + |y| \cdot \sqrt{L \cdot 2L^2/\epsilon^2} \\ = & \sqrt{2} \cdot (R + |y|)L^{1.5}/\epsilon \\ = & O(\epsilon^{-1} R \log^{1.5}(n)).\end{aligned}$$

where the third step follows from $\mathrm{Var}[\mathsf{Laplace}(x)] = 2x^2$.

The last step is because $y \in [0, R)$ and $L = \log(n)$. □

Remark 3.7 (Comparing with Prior Work)**.** *For a dataset $X \subset \mathbb{R}^d$ of size $n = |X|$ and a parameter $\alpha \in [0, 1]$ selected prior to calculation, Lemma 3.6 improves the approximation ratio from α to 1 and reduces the error dependence by a factor of $\alpha^{-0.5}$ compared to (Backurs et al., 2024).*

3.5 One Dimensional Differentially Private Data Structure

Collectively, we present the init time, space, query time, approximation error, and privacy guarantees for the 1-dimensional DP data structure (Algorithm 1, 2) in the next theorem. As a reminder, we address the 1-dimensional distance query problem (Problem 1.1 with $d = 1$) using the ℓ_1 kernel distance.

Theorem 3.8 (1-Dimensional ℓ_1 DP Distance Query)**.** *Given a dataset $X \subset \mathbb{R}$ with $|X| = n$, there is an algorithm that uses $O(n)$ space to build a data structure (Algorithm 1, 2) which supports the following operations*

- **Init**(X, ϵ)**.** *It takes dataset X and privacy parameter ϵ as input and spends $O(n)$ time to build a data-structure.*
- **Query**$(y \in \mathbb{R})$**.** *It takes y as input and spends $O(\log n)$ time to output a scalar A such that*
$$\mathbb{E}[|A - A'|] \leq O(\epsilon^{-1} R \log^{1.5} n).$$

Furthermore, the data structure is ϵ-DP.

Proof. By Lemma 3.2, we prove the storage space, and the initialization time. By Lemma 3.3, we prove the query time of the data-structure. By Lemma 3.5, we prove the differential privacy guarantee. By Lemma 3.6, we prove the error guarantee after adding the noise. □

3.6 High Dimensional ℓ_1 Distance Query

We now generalize our 1-dimensional DP data structure to d-dimensional DP distance queries. Specifically, we apply our 1-dimensional DP data structure independently to each dimension to compute high-dimensional queries.

Lemma 3.9 (Differential Privacy)**.** *The data structure* HighDimFasterTree *returned by* HighDimFasterTree.Init *(Algorithm 3) is ϵ-DP.*

Proof. The proof directly follows from calling Lemma 3.5 for d times with replacing ϵ by ϵ/d. Then using the DP composition Lemma. □

Algorithm 3 High Dimensional Distance Query

Input: n d-dimensional points in the box $[0, R)^d$, privacy parameter ϵ, query y.
data structure HIGHDIMFASTERTREE
members
  T_i for $i \in [d]$ ▷ T_i represents a FasterTree in Algorithm 1 for $i-$th dimension.
end memebers
procedure INIT(X)
  Initialize d independent data structure T_i for $i \in [d]$ with Algorithm 1. T_i builds on i-th coordinate projections of the data points. Each data structure is ϵ/d-private.
end procedure
procedure QUERY(y)
  Return the sum of the outputs of Algorithm 2 from T_i with query y_i for $i \in [d]$.
end procedure
end data structure

Lemma 3.10 (Error Guarantee)**.** *Let $X = \{x_i\}_{i\in[n]}$ be a dataset of n d-dimensional vectors $x_i \in [0, R)^d$. Let $A' = \sum_{i=1}^n |x_i - y|_1$ be the true distance query value. Let A be the output of* QUERY *(Algorithm 3). Then we have $\mathbb{E}[|A - A'|] \leq O(\epsilon^{-1} R d^{1.5} \log^{1.5} n)$.*

Proof. The output of Algorithm 3 is the sum of each dimension. Let A'_i be the output of D_i, A_i be the true distance of the i-th dimension. The total error of Algorithm 3 is $|\sum_{i=1}^d (A_i - A'_i)|$. From the proof of Algorithm 2, we know $A_i - A'_i$ is the sum of $O(L)$ Laplace noises, where $L = O(\log n)$. Therefore, we have $\mathbb{E}[A_i - A'_i] = 0$, $\mathrm{Var}[A_i - A'_i] = O(\epsilon^{-2} d^2 R^2 \log^3(n))$. Using the independence property of $A_i - A'_i$ for all i, we bound $|\sum_{i=1}^d (A_i - A'_i)|$ by

$$\begin{aligned}
\mathbb{E}[|\sum_{i=1}^d (A_i - A'_i)|] &\leq (\mathrm{Var}[\sum_{i=1}^d (A_i - A'_i)])^{1/2} \\
&= (d \,\mathrm{Var}[A_i - A'_i])^{1/2} \\
&= O(d \cdot \epsilon^{-2} d^2 R^2 \log^3 n)^{1/2} \\
&= O(\epsilon^{-1} d^{1.5} R \log^{1.5}(n)).
\end{aligned}$$

where the first step follows from Fact D.1. This completes the proof. □

Theorem 3.11 (d-Dimensional ℓ_1 DP Distance Query, Formal version of Theorem 1.3)**.** *Given a dataset $X \subset \mathbb{R}^d$ with $|X| = n$, there is an algorithm that uses $O(nd)$ space to build a data-structure (Algorithm 1, 2) which supports the following operations*

- **INIT**$(X \subset \mathbb{R}^d, \epsilon \in (0, 1))$**.** *It takes dataset X and privacy parameter $\epsilon \in (0, 1)$ as input and spends $O(nd)$ time to build a data structure.*
- **QUERY**$(y \in \mathbb{R}^d)$**.** *It takes y as input and spends $O(d \log n)$ time to output a scalar $A \in \mathbb{R}$ such that $\mathbb{E}[|A - A'|] \leq O(\epsilon^{-1} R d^{1.5} \log^{1.5} n)$.*

Proof. By Lemma 3.9, we prove the privacy guarantees. By Lemma 3.10, we prove the error guarantee of the QUERY. This completes the proof. □

4 CONCLUDING REMARKS

We present a refined data structure for differentially private kernel density estimation over the ℓ_1 kernel. Our new DP data structure significantly improves the state of the art in both query efficiency and approximation quality. Specifically, it enhances the best-known node-contaminated balanced binary tree method (Backurs et al., 2024) in three aspects: faster queries (Lemmas 3.2 and 3.3), exact ratios (rather than approximate), and reduced error dependence (Lemma 3.6). Moreover, our approach achieves a strictly better privacy-utility tradeoff without incurring additional preprocessing overhead (Theorem 3.11). Empirically, our theory aligns with numerical experiments, especially Figures 1 and 2, which highlight the improved privacy-utility tradeoff and efficiency of our DP data structure, benchmarked against (Backurs et al., 2024). We also extend our results to ℓ_2, and ℓ_p^p kernels in Appendices E and F. We defer numerical validations to Appendix A due to page limits.

ETHIC STATEMENT

This paper does not involve human subjects, personally identifiable data, or sensitive applications. We do not foresee direct ethical risks. We follow the ICLR Code of Ethics and affirm that all aspects of this research comply with the principles of fairness, transparency, and integrity.

REPRODUCIBILITY STATEMENT

We ensure reproducibility on both theoretical and empirical fronts. For theory, we include all formal assumptions, definitions, and complete proofs in the appendix. For experiments, we describe model architectures, datasets, preprocessing steps, hyperparameters, and training details in the main text and appendix. Code and scripts are provided in the supplementary materials to replicate the empirical results.

REFERENCES

Amol Aggarwal and Josh Alman. Optimal-degree polynomial approximations for exponentials and gaussian kernel density estimation. In *Proceedings of the 37th Computational Complexity Conference*, pp. 1–23, 2022.

Francesco Aldà and Benjamin I. P. Rubinstein. The bernstein mechanism: Function release under differential privacy. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 1705–1711. AAAI Press, 2017.

Josh Alman and Zhao Song. Fast attention requires bounded entries. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

Josh Alman and Zhao Song. How to capture higher-order correlations? generalizing matrix softmax attention to kronecker computation. In *The Twelfth International Conference on Learning Representations*, 2024.

Josh Alman, Timothy Chu, Aaron Schild, and Zhao Song. Algorithms and hardness for linear algebra on geometric graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 541–552. IEEE, 2020.

Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2023. URL `https://api.semanticscholar.org/CorpusID:268232499`. Accessed: 2024-08-28.

Arturs Backurs, Moses Charikar, Piotr Indyk, and Paris Siminelakis. Efficient density evaluation for smooth kernels. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 615–626, 2018.

Arturs Backurs, Piotr Indyk, and Tal Wagner. Space and time efficient kernel density estimation in high dimensions. *Advances in neural information processing systems*, 32, 2019.

Arturs Backurs, Zinan Lin, Sepideh Mahabadi, Sandeep Silwal, and Jakub Tarnawski. Efficiently computing similarities to private datasets. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Ainesh Bakshi, Piotr Indyk, Praneeth Kacham, Sandeep Silwal, and Samson Zhou. Sub-quadratic algorithms for kernel matrices via kernel density estimation, 2022. URL `https://arxiv.org/abs/2212.00642`.

Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, 1990. URL `https://www.cs.cmu.edu/~guyb/papers/Ble93.pdf`.

Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to noninteractive database privacy. *J. ACM*, 60(2), may 2013. ISSN 0004-5411.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *The Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In Nadia Heninger and Patrick Traynor (eds.), *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pp. 267–284. USENIX Association, 2019.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pp. 1897–1914. IEEE, 2022.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023a.

Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 5253–5270, 2023b.

T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. In *Proceedings of the 38th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 6755 of *Lecture Notes in Computer Science*, pp. 405–417. Springer, 2011. doi: 10.1007/978-3-642-22012-8_32.

Moses Charikar, Mikhail Kapralov, Navid Nouri, and Paris Siminelakis. Kernel density estimation through density constrained near neighbor search. *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 172–183, 2020.

Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. Gan-leaks: A taxonomy of membership inference attacks against generative models. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (eds.), *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pp. 343–362. ACM, 2020.

Benjamin Coleman and Anshumali Shrivastava. A one-pass distributed and private sketch for kernel sums with applications to machine learning at scale. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pp. 3252–3265. ACM, 2021.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

Teddy Cunningham, Graham Cormode, and Hakan Ferhatosmanoğlu. Privacy-preserving synthetic location data in the real world. *17th International Symposium on Spatial and Temporal Databases*, 2021.

Soham De, Leonard Berrada, Jamie Hayes, Samuel L. Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale. *ArXiv*, abs/2204.13650, 2022.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. Jailbreaker: Automated jailbreak across multiple large language model chatbots. *arXiv preprint arXiv:2307.08715*, 2023.

Yichuan Deng, Wenyu Jin, Zhao Song, Xiaorui Sun, and Omri Weinstein. Dynamic kernel sparsifiers. *arXiv preprint arXiv:2211.14825*, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies, Volume 1 (Long and Short Papers)*, 2018.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*, 2022.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, pp. 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pp. 715–724, 2010a. doi: 10.1145/1806689.1806787.

Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 51–60, 2010b.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pp. 1322–1333. ACM, 2015.

Yeqi Gao, Zhao Song, and Xin Yang. Differentially private attention computation. *arXiv preprint arXiv:2305.04701*, 2023.

Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In *Proceedings of the 9th International Conference on Theory of Cryptography*, TCC'12, pp. 339–356, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 9783642289132.

Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Rob Hall, Alessandro Rinaldo, and Larry A. Wasserman. Differential privacy for functions and functional data. *J. Mach. Learn. Res.*, 14(1):703–727, 2013.

Thomas A. Henzinger, Jalaj Upadhyay, and Shubham Upadhyay. Efficient private summation with applications to continual counting. In *Proceedings of the 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2023. (Exact title/venue/year may vary; please replace with final publication details.).

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Thomas Hofmann, Bernhard Schölkopf, and Alexander Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36, 01 2007.

Charlie Hou, Hongyuan Zhan, Akshat Shrivastava, Sid Wang, Aleksandr Livshits, Giulia Fanti, and Daniel Lazar. Privately customizing prefinetuning to better match user data in federated learning. *CoRR*, abs/2302.09042, 2023.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Jerry Yao-Chieh Hu, Thomas Lin, Zhao Song, and Han Liu. On computational limits of modern hopfield models: A fine-grained complexity analysis. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

Mengdi Huai, Di Wang, Chenglin Miao, Jinhui Xu, and Aidong Zhang. Privacy-aware synthesizing for crowdsourced data. In *International Joint Conference on Artificial Intelligence*, 2019.

Baihe Huang, Zhao Song, Omri Weinstein, Junze Yin, Hengjie Zhang, and Ruizhe Zhang. A dynamic low-rank fast gaussian transform. *arXiv preprint arXiv:2202.12329*, 2022.

Zhiyi Huang and Aaron Roth. Exploiting metric structure for efficient private query release. In Chandra Chekuri (ed.), *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pp. 523–534. SIAM, 2014.

Fengqing Jiang, Zhangchen Xu, Luyao Niu, Boxin Wang, Jinyuan Jia, Bo Li, and Radha Poovendran. Poster: Identifying and mitigating vulnerabilities in llm-integrated applications. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, pp. 1949–1951, 2024.

David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, 3rd edition, 2016.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. A tighter complexity analysis of sparsegpt. *arXiv preprint arXiv:2408.12151*, 2024a.

Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Zhangheng Li, Junyuan Hong, Bo Li, and Zhangyang Wang. Shake to leak: Fine-tuning diffusion models can amplify the generative privacy risk. In *2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 18–32. IEEE, 2024b.

Jiehao Liang, Zhao Song, Zhaozhuo Xu, and Danyang Zhuo. Dynamic maintenance of kernel density estimation data structure: From practice to theory. *arXiv preprint arXiv:2208.03915*, 2022.

Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *IMC '20: ACM Internet Measurement Conference, Virtual Event, USA, October 27-29, 2020*, pp. 464–483. ACM, 2020.

Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, Harsha Nori, and Sergey Yekhanin. Differentially private synthetic data via foundation model apis 1: Images. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*, 2023a.

Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860*, 2023b.

Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, Lifang He, and Lichao Sun. Sora: A review on background, technology, limitations, and opportunities of large vision models, 2024.

Haozheng Luo, Jiahao Yu, Wenxin Zhang, Jialong Li, Jerry Yao-Chieh Hu, Xingyu Xin, and Han Liu. Decoupled alignment for robust plug-and-play adaptation. *arXiv preprint arXiv:2406.01514*, 2024.

Jirí Matousek. *Lectures on discrete geometry*, volume volume 212 of Graduate texts in mathematics. Springer, 2002.

Ardalan Mirshani, Matthew Reimherr, and Aleksandra Slavković. Formal privacy for functional data with gaussian perturbations. In *International Conference on Machine Learning*, pp. 4595–4604. PMLR, 2019.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

OpenAI. Gpt-4 technical report. *ArXiv*, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Xiangyu Qi, Yangsibo Huang, Yi Zeng, Edoardo Debenedetti, Jonas Geiping, Luxi He, Kaixuan Huang, Udari Madhushani, Vikash Sehwag, Weijia Shi, et al. Ai risk management should incorporate both safety and security. *arXiv preprint arXiv:2405.19524*, 2024.

Lianke Qin, Aravind Reddy, Zhao Song, Zhaozhuo Xu, and Danyang Zhuo. Adaptive and dynamic multi-resolution hashing for pairwise summations. In *2022 IEEE International Conference on Big Data (Big Data)*, pp. 115–120. IEEE, 2022.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

Rusheb Shah, Soroush Pour, Arush Tagade, Stephen Casper, Javier Rando, et al. Scalable and transferable black-box jailbreaks for language models via persona modulation. *arXiv preprint arXiv:2311.03348*, 2023.

John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "Do Anything Now": Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models. In *The ACM Conference on Computer and Communications Security (CCS)*, 2024.

Michael T Smith, Mauricio A Álvarez, Max Zwiessele, and Neil D Lawrence. Differentially private regression with gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 1195–1203. PMLR, 2018.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems (NeurIPS)*, 32, 2019.

Lichao Sun, Yue Huang, Haoran Wang, Siyuan Wu, Qihui Zhang, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, et al. Trustllm: Trustworthiness in large language models. *arXiv preprint arXiv:2401.05561*, 2024.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Florian Tramèr, Reza Shokri, Ayrton San Joaquin, Hoang Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. Truth serum: Poisoning machine learning models to reveal their secrets. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (eds.), *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pp. 2779–2792. ACM, 2022.

Tal Wagner, Yonatan Naamad, and Nina Mishra. Fast private kernel density estimation via locality sensitive quantization. In *International Conference on Machine Learning (ICML)*, 2023.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. *arXiv preprint arXiv:2402.18567*, 2024a.

Yan Wang, Lihao Wang, Yuning Shen, Yiqun Wang, Huizhuo Yuan, Yue Wu, and Quanquan Gu. Protein conformation generation via force-guided se (3) diffusion models. *arXiv preprint arXiv:2403.14088*, 2024b.

Ziteng Wang, Chi Jin, Kai Fan, Jiaqi Zhang, Junliang Huang, Yiqiao Zhong, and Liwei Wang. Differentially private data releasing for smooth queries. *J. Mach. Learn. Res.*, 17:51:1–51:42, 2016.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*, 2021.

Chulin Xie, Zinan Lin, Arturs Backurs, Sivakanth Gopi, Da Yu, Huseyin A. Inan, Harsha Nori, Haotian Jiang, Huishuai Zhang, Yin Tat Lee, Bo Li, and Sergey Yekhanin. Differentially private synthetic data via foundation model apis 2: Text. *CoRR*, abs/2403.01749, 2024a.

Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, et al. Sorry-bench: Systematically evaluating large language model safety refusal behaviors. *arXiv preprint arXiv:2406.14598*, 2024b.

Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. Practical gan-based synthetic IP header trace generation using netshare. In Fernando Kuipers and Ariel Orda (eds.), *SIGCOMM '22: ACM SIGCOMM 2022 Conference, Amsterdam, The Netherlands, August 22 - 26, 2022*, pp. 458–472. ACM, 2022.

Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. *ArXiv*, abs/2102.12677, 2021.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Da Yu, Sivakanth Gopi, Janardhan Kulkarni, Zinan Lin, Saurabh Naik, Tomasz Lukasz Religa, Jian Yin, and Huishuai Zhang. Selective pre-training for private fine-tuning. *Trans. Mach. Learn. Res.*, 2024, 2024a.

Jiahao Yu, Xingwei Lin, and Xinyu Xing. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*, 2023.

Jiahao Yu, Haozheng Luo, Jerry Yao-Chieh Hu, Wenbo Guo, Han Liu, and Xinyu Xing. Enhancing jailbreak attack against large language models through silent tokens. *arXiv preprint arXiv:2405.20653*, 2024b.

Xiang Yue, Huseyin A. Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Hoda Shajari, Huan Sun, David Levitan, and Robert Sim. Synthetic text generation with differential privacy: A simple and practical recipe. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

Yi Zeng, Yu Yang, Andy Zhou, Jeffrey Ziwei Tan, Yuheng Tu, Yifan Mai, Kevin Klyman, Minzhou Pan, Ruoxi Jia, Dawn Song, et al. Air-bench 2024: A safety benchmark based on risk categories from regulations and policies. *arXiv preprint arXiv:2407.17436*, 2024.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*, 2024.

Xiangxin Zhou, Xiwei Cheng, Yuwei Yang, Yu Bao, Liang Wang, and Quanquan Gu. Decompopt: Controllable and decomposed diffusion models for structure-based molecular optimization. *arXiv preprint arXiv:2403.13829*, 2024a.

Xiangxin Zhou, Dongyu Xue, Ruizhe Chen, Zaixiang Zheng, Liang Wang, and Quanquan Gu. Antigen-specific antibody design via direct energy-based preference optimization. *arXiv preprint arXiv:2403.16576*, 2024b.

Impact Statement

Our results offer a combination of fast queries, tight accuracy, and robust privacy guarantees. These results are essential for large-scale data analysis scenarios where computational efficiency and data confidentiality are critical. By the formal nature of this work, we do not expect any immediate negative social impact.

LLM Usage Disclosure

We used large language models (LLMs) to aid and polish writing, such as improving clarity, grammar, and conciseness. We also used LLMs for retrieval and discovery, for example exhausting literature to identify potential missing related work. All technical content, proofs, experiments, and results are original contributions by the authors.

A Proof-of-Concept Experiments

We validate the efficacy of our theory in both synthetic and real-world experiments.

A.1 Synthetic Data

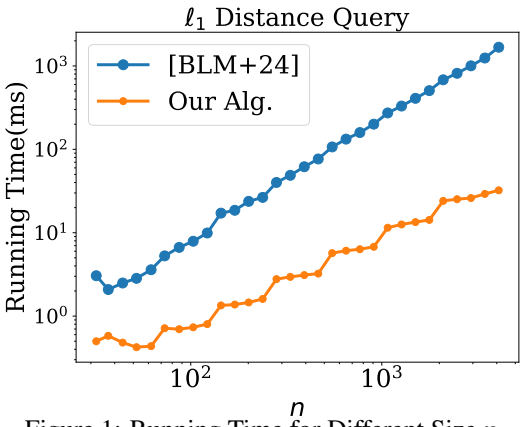


Figure 1: Running Time for Different Size n

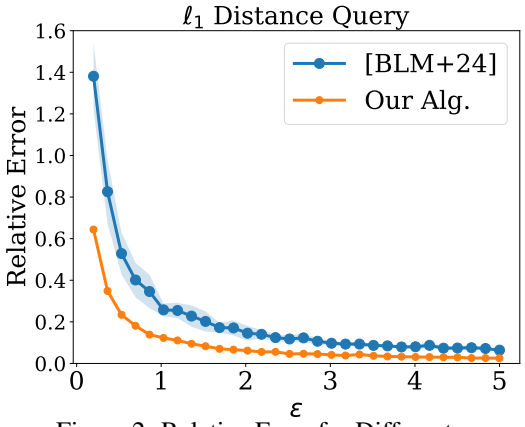


Figure 2: Relative Error for Different ϵ

Here we present minimally sufficient numerical results to back up our theoretical analysis of ℓ_1 algorithm.

Experiments Setup. In the ℓ_1 experiments, the objective is to compute function $f(y) = \sum_{x \in X} |x - y|$. For each experiment, we compute the average of 15 trials, with the standard deviation (± 1) shaded where appropriate.

Computational Resource. We conduct all experiments on a laptop with Intel i7-9750H CPU and 16 GB RAM. All codes are implemented in Python 3.12.3.

Baseline. We choose (Backurs et al., 2024) as our baseline and compare our algorithm with theirs on both running time and relative error. In their algorithm, we set the hyper-parameter $\alpha = 0.1$ which corresponds to the experiments settings in (Backurs et al., 2024).

Approximation Test. We first show that as ϵ increases, both (Backurs et al., 2024) and our algorithm approximate the true function f. To illustrate this, our first synthetic dataset X consists of 1000 data points evenly distributed in the range $[0, 1]$, with query points also evenly distributed within the same range. In this case, the ground truth function f is approximately equal to the integral $f(y) = \int_0^1 |x - y| \, dx = y^2 - y + 1/2$ for $y \in [0, 1]$. This setup allows for an easy comparison of our output to the true function. In Figure 3, we show that both the outputs of (Backurs et al., 2024) and our algorithm approximate the true function f as ϵ increases, with our method achieving a faster convergence rate.

Running Time Test. In Figure 1, we compare the running time and relative error between (Backurs et al., 2024) and our algorithm. For running time comparison, our datasets X consists of random points in $[0, 1]$ with different sizes ranging from 32 to 4096. The number of queries equals to the

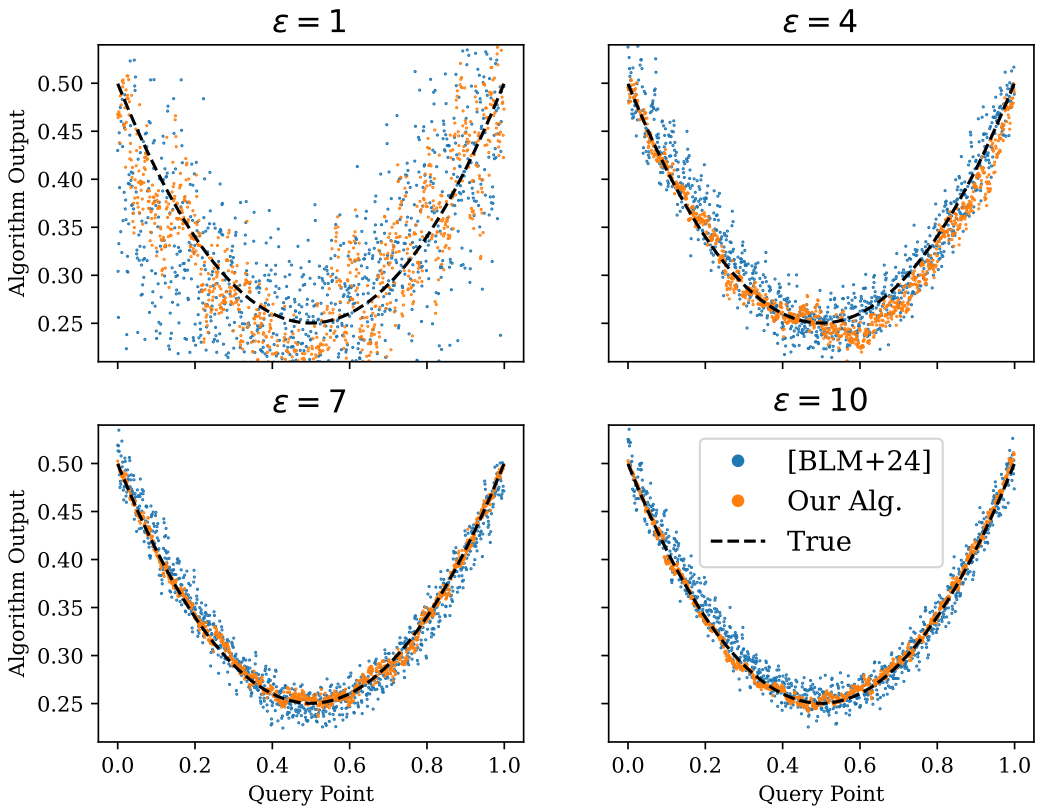


Figure 3: We benchmark our results against those of the previous best (Backurs et al., 2024). The dashed line is the ground truth. The orange and blue dots correspond to our method and (Backurs et al., 2024), respectively. While, both algorithms approximate the true ℓ_1 distance as ϵ increases, it is clear that our method delivers more accurate and more robust predictions.

size of $|X|$. We demonstrate that our algorithm consistently has a lower running time than (Backurs et al., 2024).

Relative Error Test. In Figure 2, we set $|X| = 1000$ and compute relative errors. Every data point and query point is uniformly chosen from $[0, 1]$ independently. The number of queries also equals to the size of $|X|$. For every ϵ ranging from 0.2 to 5, our algorithm consistently achieves a lower relative error compared to (Backurs et al., 2024).

These numerical results align with our theory: our algorithm improves (Backurs et al., 2024) in both running time and relative error.

A.2 Real-World Data

Here we extend our minimally sufficient numerical results to real-world data (CIFAR-10).

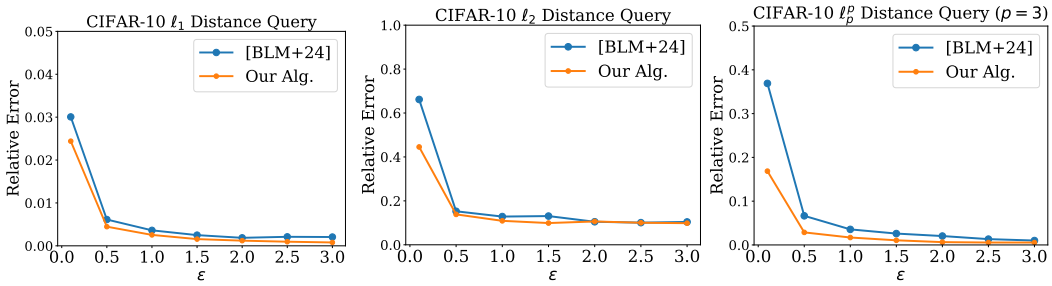
Data. CIFAR-10 is a dataset consisting of 60000 32x32 color images. We project each of them into a real vector of dimension 100 and compute distance queries for various ϵ.

Experiments Setup. We conduct new experiments on CIFAR-10 to compare the performances between our algorithm and (Backurs et al., 2024).

In our experiments, we choose different $\epsilon \in \{0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$. Then we construct a database containing 10000 vectors of dimension 100 and each time we process 100 queries randomly drawn from the CIFAR-10 test batch.

Relative Error Test. We consider three types of distance queries: ℓ_1 (Figure 4(a)), ℓ_2 (Figure 4(b)) and ℓ_p^p with $p = 3$ (Figure 4(c)). For each of them, we compute the average relative error of the

queries. Figure 4 shows that our algorithm outperforms (Backurs et al., 2024) on all of the three distance metrics.



(a) ℓ_1 queries (b) ℓ_2 queries (c) ℓ_p^p queries

Figure 4: We conduct different types of queries on CIFAR-10. The blue line is the (Backurs et al., 2024) and the yellow line is the result of our algorithm.

B Related Work

Privacy, Security and Safety in Large Foundation Models. The motivating problems for this work come from the privacy concerns of large foundation models (Lin et al., 2024; Xie et al., 2024a). In modern machine learning, Foundation models (Bommasani et al., 2021), including pretrained transformer and diffusion models, gain popularity in many AI applications due to their ability to generalize across diverse tasks with minimal fine-tuning. Pretrained transformers, such as BERT (Devlin et al., 2018), GPT (Brown et al., 2020), and Llama (Touvron et al., 2023b;a), leverage vast amounts of data to learn general-purpose, context-aware representations. Diffusion models (Peebles & Xie, 2023; Ho et al., 2020; Song & Ermon, 2019), on the other hand, excel in generative tasks, particularly in producing high-quality images and data distributions through iterative refinement (Nichol et al., 2021; Ramesh et al., 2022; Liu et al., 2024; Zhou et al., 2024a;b; Wang et al., 2024a;b). Importantly, what makes them fundamental and versatile is their flexibility for diverse downstream tasks via fine-tuning methods (Zheng et al., 2024; Ding et al., 2022; Lester et al., 2021; Liu et al., 2021; Hu et al., 2022), hence "foundation." Together, these models signify a shift towards more powerful AI systems that serve as the basis for a wide range of applications across different domains.

However, there exists a gap from productizing many these advancements due to privacy, safety, and security concerns (Qi et al., 2024; Sun et al., 2024; Li et al., 2024b; Weidinger et al., 2021). These models, due to their vast scale and extensive training on large datasets, risk exposing sensitive information and amplifying biases present in the data. Privacy issues arise when models inadvertently memorize and reproduce personal data from training sets (Carlini et al., 2023b; 2021). Safety concerns include the potential for generating harmful or misleading content, especially in applications where accuracy is paramount (Weidinger et al., 2021; OpenAI, 2023; Team et al., 2023; Anthropic, 2023; Yu et al., 2024b; Luo et al., 2024; Shen et al., 2024; Deng et al., 2023; Liu et al., 2023b;a; Shah et al., 2023; Yu et al., 2023). Security vulnerabilities also exist, as these models may be susceptible to adversarial attacks that manipulate outputs or extract proprietary information (Zeng et al., 2024; Jiang et al., 2024; Xie et al., 2024b). Addressing these challenges is essential to ensure the responsible and secure use of foundation models across various applications. In this work, we focus on the privacy of foundation models at a fundamental level by study the formalized KDE query Problem 1.1.

Differential Privacy (DP). Differential Privacy (DP) is a standard tool for understanding and mitigating privacy concerns in machine learning. Proposed by Dwork et al. (2006), DP offers a robust approach to protecting sensitive information. Given the fact that non-private ML models expose sensitive user data (Fredrikson et al., 2015; Carlini et al., 2019; Chen et al., 2020; Carlini et al., 2021; 2022; 2023a; Haim et al., 2022; Tramèr et al., 2022; Carlini et al., 2023b), researchers propose various methodologies. One approach involves generating synthetic datasets that closely resemble the original private data and training models on these synthetic datasets (Lin et al., 2020; Li et al., 2022; Yu et al., 2022; Yin et al., 2022; Yue et al., 2023; Lin et al., 2024). Another strategy is to use similar public examples for pre-training models (Hou et al., 2023; Yu et al., 2024a). Additionally, the

DP-SGD method has benefited from incorporating public data similar to private datasets to enhance downstream model performance (Yu et al., 2021; De et al., 2022; Yu et al., 2022; Yin et al., 2022; Li et al., 2022; Hou et al., 2023; Yu et al., 2024a; Lin et al., 2024). As highlighted by Backurs et al. (2024), the common denominator of all these works is the need to *compute similarities to a private dataset*. In this work, we study this problem at a fundamental level by formalizing it as the DP-KDE query problem (Problem 1.1).

Kernel Density Estimation (KDE). Kernel Density Estimation (KDE) is a key technique in statistics and machine learning. This technique converts a collection of data points into a smoothed probability distribution (Schölkopf & Smola, 2002; Shawe-Taylor & Cristianini, 2004; Hofmann et al., 2007). It is prevalent in many private applications such as crowdsourcing and location sharing (Huai et al., 2019; Cunningham et al., 2021). Although research on non-private KDE already produce efficient methods (Backurs et al., 2018; 2019; Alman et al., 2020; Charikar et al., 2020; Liang et al., 2022; Qin et al., 2022; Bakshi et al., 2022; Huang et al., 2022; Deng et al., 2022), adapting these techniques to DP remains complex (Gupta et al., 2012; Blum et al., 2013; Aldà & Rubinstein, 2017; Wagner et al., 2023; Backurs et al., 2024). The best algorithm to date is by Backurs et al. (2024). In this work, we introduce a new data structure for DP-KDE that improves upon (Backurs et al., 2024) in both privacy-utility trade-off and efficiency.

Comparison with Function-Sanitizing Approaches for Differentially Private KDE. Hall et al.(2013) (Hall et al., 2013) pioneer DP kernel density estimation by adding a Gaussian-process perturbation to the entire density function. This yields an (ϵ, δ)-DP sanitized function that can be queried anywhere, but it requires heavy noise and came with limited accuracy guarantees. Mirshani et al.(2019) (Mirshani et al., 2019) extend this "function release" approach using Gaussian process noise in Banach spaces, likewise achieving (ϵ, δ)-DP over the full function at a high privacy cost. Aldà and Rubinstein (2017) (Aldà & Rubinstein, 2017) instead propose the *Bernstein mechanism*, approximating the KDE with a finite polynomial basis and adding independent noise to each coefficient. This produces a published density function under (pure) DP with improved utility but requires truncating the function to a fixed basis. Smith et al.(2018) (Smith et al., 2018) apply DP to Gaussian process models of the KDE. They show that releasing an unrestricted GP-based density demands large noise, and they improve accuracy by assuming a fixed set of query points (reducing query flexibility). All these methods treat the KDE as a static function to privatize and publish up front, incurring a large one-time privacy cost or forcing constraints on queries. In contrast, our work uses a specialized ℓ_1 kernel decomposition and a balanced tree data structure to answer KDE queries efficiently under DP. Rather than outputting an explicit density estimate, we privately pre-compute a tree that returns kernel density queries *on the fly* in sublinear time. This design concentrates the privacy budget on the queried regions and exploits the ℓ_1 kernel's structure Consequently, our proposal yields higher accuracy and faster query responses than prior DP-KDE methods under the same privacy requirements.

DP Tree and Matrix mechanisms. Dwork et al. (2010a) and Chan et al. (2011) pioneer tree-based mechanisms for answering range queries under differential privacy. Their key insight is to store counts in a balanced binary tree (or equivalently use the matrix mechanism) so that each query touches only $O(\log n)$ noisy aggregates, yielding lower error than naive solutions. Recent work by Henzinger et al. (2023) further optimize these ideas to improve continual counting bounds. Our approach builds on this line of research. We also employ a balanced binary tree for fast, private query evaluation. However, we extend the classic tree mechanism by storing both counts and partial distance sums at each node, rather than just counts. This allows us to handle ℓ_1 kernel density estimation queries efficiently: we retrieve the relevant (noisy) partial sums from each level of the tree and combine them to compute $|x - y|$ over the dataset. Thus, while our algorithm inherits the core principle of aggregating noisy statistics along a tree, it generalizes the framework to distance-weighted queries, reducing error and removing the need for an additional approximation factor.

C Limitations and Future Work

Limitations. Our technique and analysis focus *only* on *static* datasets and the ℓ_1, ℓ_2, and ℓ_p^p kernels; extending the approach to other similarity measures or dynamic/streaming data remains an open challenge. While our space and preprocessing time match existing solutions, they are still $O(nd)$ and may become restrictive in very high dimensions or for extremely large n. Moreover, although we reduce the query time by an $\alpha^{-1} \log n$ factor and improve approximation/error bounds,

the constants involved in the data structure could be non-negligible in practice, especially under tight privacy constraints.

Future Work. Our novel tree construction strategy may inspire further research on advanced data structures for privacy-preserving query processing. Future work could explore refining these constants, handling broader classes of kernels, and adapting the method to incremental or online data settings. We leave these directions for future exploration.

D Basic Facts

Fact D.1. *Let X denote a random variable with $\mathbb{E}[X] = 0$. Then it holds*

$$\mathbb{E}[|X|] \leq (\mathrm{Var}[X])^{1/2}.$$

Proof. It is easy to see that $(\mathbb{E}[|X|])^2 \leq \mathbb{E}[|X|^2] = \mathbb{E}[X^2]$. Next, we show

$$(\mathbb{E}[|X|])^2 \leq \mathbb{E}[X^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \leq \mathrm{Var}[X].$$

Thus, we complete the proof. □

Fact D.2 (Union Bound)**.** *Let $A_1, A_2, ..., A_n$ be a set of probability events. It holds*

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] \leq \sum_{i=1}^{n} \Pr[A_i].$$

Proof. We prove union bound by induction. For $n = 1$, $\Pr[A_1] = \Pr[A_1]$. Suppose this inequality holds true for k, then for $k + 1$,

$$\begin{aligned}
\Pr\left[\bigcup_{i=1}^{k+1} A_i\right] &= \Pr\left[\bigcup_{i=1}^{k} A_i\right] + \Pr[A_{k+1}] - \Pr\left[(\bigcup_{i=1}^{k} A_i) \bigcap A_{k+1}\right] \\
&\leq \Pr\left[\bigcup_{i=1}^{k} A_i\right] + \Pr[A_{k+1}] \\
&\leq (\sum_{i=1}^{k} \Pr[A_i]) + \Pr[A_{k+1}] \\
&= \sum_{i=1}^{k+1} \Pr[A_i].
\end{aligned}$$

Thus, we complete the proof. □

E ℓ_2 Distance Query

Using the same technique in (Backurs et al., 2024), we extend our ℓ_1 result to ℓ_2 norm. With our faster tree data structure, we reduce the error in (Backurs et al., 2024) from $(1 + \alpha, O(\epsilon^{-1}\alpha^{-1.5}R\log^2 n))$ to $(1 + \alpha, O(\epsilon^{-1}\alpha^{-1}R\log^2 n))$. This method uses a mapping from ℓ_2 to ℓ_1 space that retains distances between data points with high probability.

References	Approx. Ratio	Error	Query Time	Init time	Space
(Backurs et al., 2024)	$1+\alpha$	$\alpha^{-1.5}\epsilon^{-1}R\log^2 n$	$\alpha^{-2}(d+\alpha^{-1}\log^2 n)\log n \cdot \log(1/\alpha)$	$\alpha^{-2}nd\log n \cdot \log(1/\alpha)$	$n(d+\alpha^{-2}\log n\log(1/\alpha))$
Theorem E.2	$1+\alpha$	$\alpha^{-1}\epsilon^{-1}R\log^2 n$	$\alpha^{-2}(d+\log n)\log n$	$\alpha^{-2}nd\log n$	$n(d+\alpha^{-2}\log n)$

Table 2: Comparison of $\|x - y\|_2$ Results with Best Known Algorithm by (Backurs et al., 2024). We ignore the big-O notation in the table, for simplicity of presentation. We use n to denote the number of points in dataset. We use d to denote the dimension for each point in the dataset. We assume all the points are bounded by R. We use the ϵ to denote ϵ-DP.

E.1 Error and DP Guarantees

Lemma E.1 ((Matousek, 2002)). *Let $\gamma \in (0,1)$ and define $T : \mathbb{R}^d \to \mathbb{R}^k$ by*

$$T(x)_i = \frac{1}{\beta k} \sum_{j=1}^{d} Z_{ij} x_j, i \in [k],$$

where $\beta = \sqrt{2/\pi}$ and Z_{ij} are standard Gaussians. Then for every vector $x \in \mathbb{R}^d$, we have

$$\Pr[(1-\gamma)\|x\|_2 \le \|T(x)\|_1 \le (1+\gamma)\|x\|_2] \ge 1 - e^{-c\gamma^2 k},$$

where $c > 0$ is a constant

Theorem E.2. *Let $X \subset \mathbb{R}^d$ be a private dataset of size n with a bounded diameter of R in ℓ_2. For any $\alpha \in [0,1]$, there exists an ϵ-DP data structure such that for any fixed query $y \in \mathbb{R}^d$, with probability 0.99, it outputs $Z \in \mathbb{R}$ satisfying*

$$|Z - \sum_{x \in X} \|x - y\|_2| \le \alpha \sum_{x \in X} \|x - y\|_2 + O(\epsilon^{-1}\alpha^{-1} R \log^2 n).$$

Proof. Let $\gamma = \alpha$, and $k = O((\log n)/\alpha^2)$. Our algorithm first maps X to $T(X)$ using Lemma E.1, then solves ℓ_1 query on $T(X)$ with Algorithm 3. By triangle inequality, we have

$$\begin{aligned}
&|Z - \sum_{x \in X} \|x - y\|_2| \\
\le\ & \underbrace{|\sum_{x \in X} \|T(x) - T(y)\|_1 - \sum_{x \in X} \|x - y\|_2|}_{\mathrm{err}_1} + \underbrace{|Z - \sum_{x \in X} \|T(x) - T(y)\|_1|}_{\mathrm{err}_2}.
\end{aligned}$$

Therefore, we divide the error into two parts, err_1 and err_2. err_1 is from mapping T (Lemma E.1), err_2 is from Algorithm 3.

Bound of err_1. We prove with probability 0.99,

$$|\sum_{x \in X} \|T(x) - T(y)\|_1 - \sum_{x \in X} \|x - y\|_2| \le \alpha \sum_{x \in X} \|x - y\|_2.$$

From Lemma E.1, let $\gamma = \alpha, k = c^{-1}(\log n + c')/\alpha^2$, where c' is a sufficiently large constant. We have

$$\Pr[|\|x\|_2 - \|T(x)\|_1| > \alpha\|x\|_2] \le e^{-(c' + \log n)} \le 0.01 \cdot \frac{1}{n}.$$

Therefore,

$$\begin{aligned}
&\Pr\left[|\sum_{x \in X} \|T(x) - T(y)\|_1 - \sum_{x \in X} \|x - y\|_2| \le \alpha \sum_{x \in X} \|x - y\|_2\right] \\
\ge\ & \Pr[\ \forall x \in X \mid |\|x\|_2 - \|T(x)\|_1| \le \alpha\|x\|_2] \\
\ge\ & 1 - \sum_{x \in X} \Pr[\ |x\|_2 - \|T(x)\|_1| > \alpha\|x\|_2] \\
\ge\ & 1 - n \cdot 0.01 \cdot \frac{1}{n} \\
=\ & 0.99.
\end{aligned}$$

The second step follows from union bound (Fact D.2).

Bound of err_2. After mapping T, with high probability, each coordinate ranges from 0 to $R' := O(R/k)$. Therefore, the additive error caused by Algorithm 3 equals

$$\begin{aligned}
|Z - \sum_{x \in X} \|T(x) - T(y)\|_1| &= O(\epsilon^{-1} R' k^{1.5} \log^{1.5} n) \\
&= O(\epsilon^{-1} \frac{R\alpha^2}{\log n} (\frac{\log n}{\alpha^2})^{1.5} \log^{1.5} n) \\
&= O(\alpha^{-1}\epsilon^{-1} R \log^2 n),
\end{aligned}$$

where the second step follows from choice of R'.

This completes the proof. □

E.2 Init and Query Time

Finally, we prove the init time and query time of our algorithm for ℓ_2 kernel distance.

Init Time. For initializing, our algorithm contains two steps. First we map each data point from d-dimensional to k-dimensional. This step is a matrix multiplication and hence takes $O(ndk)$ time. Then we build a balanced binary tree. This step takes $O(nk)$ time. Therefore, given $k = (\log n)/\alpha^2$, the total init time is $O(\alpha^{-2} nd \log n)$.

Query Time. For each query, our algorithm also contains two steps. First we map query y from d-dimensional to k-dimensional. This step is a matrix multiplication and hence takes $O(dk)$ time. Then we query y on the balanced binary tree. This step takes $O(k \log n)$ time. Therefore, given $k = (\log n)/\alpha^2$, the total query time is $O(\alpha^{-2}(d + \log n) \log n)$.

F ℓ_p^p Distance Query

Suppose $p \geq 1$ is an integer. To calculate ℓ_p^p distance queries, we use binomial expansion techniques. Similar to ℓ_1, we first consider the one-dimensional case.

References	Approx. Ratio	Error	Query Time	Init time	Space
(Backurs et al., 2024)	$1+\alpha$	$\epsilon^{-1}\alpha^{-0.5}R^p d^{1.5}\log^{1.5} n$	$\alpha^{-1} d \log^2 n$	nd	nd
ℓ_p^p	1	$\epsilon^{-1} p^{-0.5} R^p d^{1.5} \log^{1.5} n$	$d \log n$	$(n + pn^{1/p})d$	$pn^{1/p}d$

Table 3: Comparison of $\|x-y\|_p^p$ Results with Best Known Algorithm by (Backurs et al., 2024). We ignore the big-O notation in the table, for simplicity of presentation. We use n to denote the number of points in dataset. We use d to denote the dimension for each point in the dataset. We assume all the points are bounded by R. We use the ϵ to denote ϵ-DP.

Remark F.1. *We remark that for the small p regime $p \in [1, 2]$, our result (see Table 3) match the init time and space of (Backurs et al., 2024), improve the approximation from $(1+\alpha)$ to 1, reduce the error by a factor of $\alpha^{-0.5}$, and reduce the query time by a factor of $\alpha^{-1} \log n$.*

We first extend our key observation Lemma 3.1 with binomial expansion.

Lemma F.2. *For a collection of numbers $\{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}$ and a number $y \in \mathbb{R}$. We define two sets*

$$S_+ := \{k \in [n] \,:\, x_k > y\}$$
$$S_- := \{k \in [n] \,:\, x_k < y\},$$

It holds

$$\sum_{k=1}^n |x_k - y|^p = \sum_{j=0}^p \binom{p}{j} y^{p-j} ((-1)^{p-j} \sum_{k \in S_+} x_k^j + (-1)^j \sum_{k \in S_-} x_k^j),$$

where $\binom{p}{j}$ denotes the binomial coefficient that $\binom{p}{j} = \frac{p!}{j!(p-j)!}$.

Proof. We show that

$$\begin{aligned}
\sum_{k=1}^n |x_k - y|^p &= \sum_{x_k \in S_+} (x_k - y)^p + \sum_{x_k \in S_-} (y - x_k)^p \\
&= (\sum_{x_k \in S_+} \sum_{j=0}^p (-1)^{p-j} \binom{p}{j} x_k^j y^{p-j}) + (\sum_{x_k \in S_-} \sum_{j=0}^p (-1)^j \binom{p}{j} x_k^j y^{p-j}) \\
&= \sum_{j=0}^p (\binom{p}{j} (-1)^{p-j} y^{p-j} \sum_{k \in S_+} x_k^j) + \sum_{j=0}^p (\binom{p}{j} (-1)^j y^{p-j} \sum_{k \in S_-} x_k^j)
\end{aligned}$$

$$= \sum_{j=0}^{p} \binom{p}{j} y^{p-j} ((-1)^{p-j} \sum_{k \in S_+} x_k^j + (-1)^j \sum_{k \in S_-} x_k^j).$$

Thus, we complete the proof. □

Lemma F.2 presents a decomposition of the final answer. To calculate the final answer, on each node, we store $\sum_{x_k \in I} x_k^q$ for all $q \in \mathbb{N}, 0 \leq q \leq p$. I denotes the interval that the node maintains. For each query, we define $s_{\text{left},q}$ and $s_{\text{right},q}$ as $\sum_{k \in S_-} x_k^q$ and $\sum_{k \in S_+} x_k^q$ respectively. Then we calculate them and construct the final answer.

Algorithm 4 ℓ_p^p Distance Query

data structure ℓ_p^p FASTERTREE ▷ Theorem F.3
members
  $T := \{s_{l,j,q}\}$ ▷ $s_{l,j,q}$ represents $\sum_{x_k \in [(j-1)\cdot \frac{R}{2^{l-1}}, j \cdot \frac{R}{2^{l-1}})} x_k^q$ as defined above.
end members

procedure INIT(X, n, ϵ, L)
  for each x_k in X **do**
    find the leaf node j of layer L that x_k is in the interval $I_{L,j} = [(j-1) \cdot \frac{R}{2^{L-1}}, j \cdot \frac{R}{2^{L-1}})$
    for q from 0 to p **do**
      $s_{L,j,q} \leftarrow s_{L,j,q} + x_k^q$
    end for
  end for
  for each layer l from $L-1$ to 1 **do**
    for each node j in layer l **do**
      for q from 0 to p **do**
        $s_{l,j,q} \leftarrow s_{l+1,2j-1,q} + s_{l+1,2j,q}$
      end for
    end for
  end for
  for q from 0 to p **do**
    Add independent noises drawn from $\mathsf{Laplace}(pLR^q/\epsilon)$ to each $s_{l,j,q}$
  end for
  $T \leftarrow \{s\}$
end procedure

procedure QUERY($y \in \mathbb{R}$)
  for q from 0 to p **do**
    $s_{\text{left},q}, s_{\text{right},q} \leftarrow 0$
  end for ▷ As previously defined
  for each layer l from 2 to L **do**
    find the node j of layer l that y is in the interval $I_{l,j} = [(j-1) \cdot \frac{R}{2^{l-1}}, j \cdot \frac{R}{2^{l-1}})$
    for q from 0 to p **do**
      if $j \mod 2 = 0$ **then** ▷ if node j is the right child of its parent
        $s_{\text{left},q} \leftarrow s_{\text{left},q} + s_{l,j-1,q}$
      else ▷ if node j is the left child of its parent
        $s_{\text{right},q} \leftarrow s_{\text{right},q} + s_{l,j+1,q}$
      end if
    end for
  end for
  return $\sum_{q=0}^{p} \binom{p}{q} y^{p-q} ((-1)^{p-q} s_{\text{right},q} + (-1)^j s_{\text{left},q})$ ▷ Lemma F.2
end procedure
end data structure

Theorem F.3 (1-Dimensional ℓ_p^p DP Distance Query)**.** *Given a dataset $X \subset \mathbb{R}$ with $|X| = n$, there is an algorithm that uses $O(pn^{1/p})$ space to build a data structure (Algorithm 4) which supports the following operations*

- **INIT**(X, ϵ). *It takes dataset X and privacy parameter ϵ as input and spends $O(n + pn^{1/p})$ time to build a data-structure.*
- **QUERY**$(y \in \mathbb{R})$. *It takes y as input and spends $O(\log n)$ time to output a scalar A such that $\mathbb{E}[|A - A'|] \leq O(p^{-0.5}\epsilon^{-1}R^p \log^{1.5} n)$.*

Furthermore, the data structure is ϵ-DP.

Proof. We set the total layers $L = (\log n)/p$.

Init Time. The total number of nodes on the tree is $2^{L+1} - 1 = O(n^{1/p})$. Each node stores p values, so there are $O(pn^{1/p})$ values stored on the tree. Plus the time of iterating all data points, initializing these values takes $O(n + pn^{1/p})$ time.

Query Time. Each query iterate through all layers. On each layer it takes $O(p)$ time to calculate $s_{\text{left},q}$ and $s_{\text{right},q}$. There are $(\log n)/p$ layers, so the total query time is $O(\log n)$.

Privacy Guarantees. Similar to Lemma 3.5, we consider each q separately. For q, let $F_q(X) : [0, R)^n \to \mathbb{R}^{2^{L+1}-1}$ be the mapping from dataset X to $s_{l,j,q}$ of all l, j. Next we prove each $F_q(X)$ is (ϵ/p)-DP.

For $F_q(X)$, when changing each data point, at most one node changes at each layer. Each node changes by at most $O(R^q)$. Therefore, the sensitivity is $O(LR^q)$. Adding coordinate-wise Laplace noise with magnitude $\eta = O(pLR^q/\epsilon)$ suffices to ensure (ϵ/p)-DP using the standard Laplace mechanism.

By Lemma 2.2, the differential privacy parameter ϵ of the tree $T := (F_q(X) : 0 \leq q \leq p)$ equals $p \cdot \epsilon/p = \epsilon$. This completes the proof.

Error Guarantees. Similar to Lemma 3.6, the additive error consists of two parts.

The first part is from the data in the leaf node which contains query y. The error is

$$\sum_{x_k \in [(j-1)\cdot R/2^L, j\cdot R/2^L)} |x_k - y|^p \leq n \cdot (\frac{R}{2^L})^p.$$

When $L = (\log n)/p$, this error is $O(R^p)$.

The second part is the Laplace noise. We first show that

$$\begin{aligned}
\mathbb{E}[|\sum_{i=1}^{L} \mathsf{Laplace}(\lambda)|] &\leq \sqrt{\mathrm{Var}[\sum_{i=1}^{L} \mathsf{Laplace}(\lambda)]} \\
&\leq \sqrt{L \cdot \mathrm{Var}[\mathsf{Laplace}(\lambda)]} \\
&= \sqrt{2L\lambda^2} \\
&= \sqrt{2L}\lambda,
\end{aligned}$$

where the first step follows from Fact D.1, the third step follows from $\mathrm{Var}[\mathsf{Laplace}(x)] = 2x^2$.

Replacing $\lambda = pR^pL/\epsilon$, we have

$$\mathbb{E}[|\sum_{i=1}^{L} \mathsf{Laplace}(pR^pL/\epsilon)|] \leq \sqrt{2}pR^pL^{1.5}/\epsilon. \tag{F.1}$$

Then we bound the error with this inequality:

$$\begin{aligned}
\mathbb{E}[|A - A'|] &\leq \mathbb{E}[|\sum_{q=0}^{p} \binom{p}{q} y^{p-q} \sum_{i=1}^{L}((-1)^{p-j}\mathsf{Laplace}(pR^qL/\epsilon) + (-1)^j \mathsf{Laplace}(pR^qL/\epsilon))|] \\
&\leq \sum_{q=0}^{p} \binom{p}{q} y^{p-q}\, \mathbb{E}[|\sum_{i=1}^{L}(\mathsf{Laplace}(pR^qL/\epsilon) + \mathsf{Laplace}(pR^qL/\epsilon))|]
\end{aligned}$$

$$= \sum_{q=0}^{p} \binom{p}{q} y^{p-q} \cdot 2pR^q L^{1.5}$$
$$= 2pL^{1.5} \sum_{q=0}^{p} \binom{p}{q} y^{p-q} R^q$$
$$= 2pL^{1.5}(y+R)^p$$
$$= O(p^{-0.5} R^p \log^{1.5} n),$$

where the third step follows from Eq. (F.1). The last step is from $L = (\log n)/p, y \in [0, R)$. □

For high dimensional ℓ_p^p queries, we use the same procedures in Section 3.6. We build d independent data structures for each dimension following Algorithm 3. We also compare the privacy and error guarantees with those in (Backurs et al., 2024) in Table 3.

Theorem F.4 (d-Dimensional ℓ_p^p DP Distance Query)**.** *Given a dataset $X \subset \mathbb{R}^d$ with $|X| = n$, there is an algorithm that uses $O(pn^{1/p}d)$ space to build a data-structure which supports the following operations*

- **INIT**(X, ϵ)**.** *It takes dataset X and privacy parameter ϵ as input and spends $O(nd + pn^{1/p}d)$ time to build a data-structure.*
- **QUERY**$(y \in \mathbb{R}^d)$**.** *It takes y as input and spends $O(d \log n)$ time to output a scalar A such that $\mathbb{E}[|A - A'|] \leq O(p^{-0.5}\epsilon^{-1} R^p d^{1.5} \log^{1.5} n)$.*

Furthermore, the data structure is ϵ-DP.

Proof. We omit the proof as it follows from Section 3.6 in a straightforward manner. □

G Diagram Examples Showing the Difference Between (Backurs et al., 2024)'s and Ours Algorithms

Here we provide visualization examples for Section 2.2 and Section 2.3 to contrast our proposal with that of (Backurs et al., 2024). To better understand our idea, let us consider the non-DP version, which has no noise. Moreover, we simplify our tree structure. By "simplify", we mean that, we drop some information from our trees even before adding any DP noise.

Under this setting, here we provide a simple way to show the difference between their algorithm and ours. Suppose the data is [1, 2, 3, 4].

- **(Backurs et al., 2024)'s tree** is a classical binary tree, as shown in Figure 5. Each leaf stores the original value, and each intermediate node stores the sum of its two children.
- **Our tree** is a prefix-sum tree, as shown in Figure 6. Each intermediate node stores the prefix-sum immediately to the left of all its descendant leaves. For example, node c (value 3) stores the value of node f. Each leaf node stores the prefix-sum from the leftmost leaf node up to itself. For another example, leaf node g stores the sum of all values to its left.

For details on prefix-sum trees, we refer readers to the following standard references. The first is by CMU professor Guy Blelloch's lecture notes, "Prefix Sums and Their Applications" (Blelloch, 1990). The second is the textbook "Programming Massively Parallel Processors: A Hands-On Approach" (Kirk & Hwu, 2016). The last is the classic algorithms textbook, "Introduction to Algorithms" (3rd Ed.) (Cormen et al., 2009).

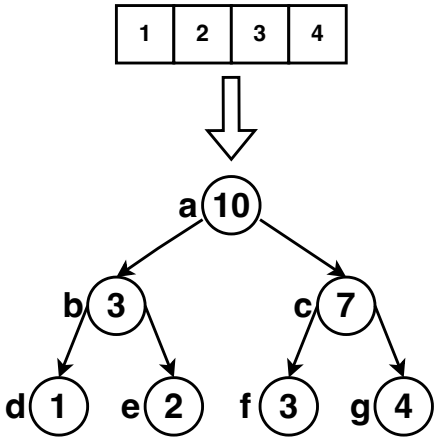


Figure 5: Example of binary tree

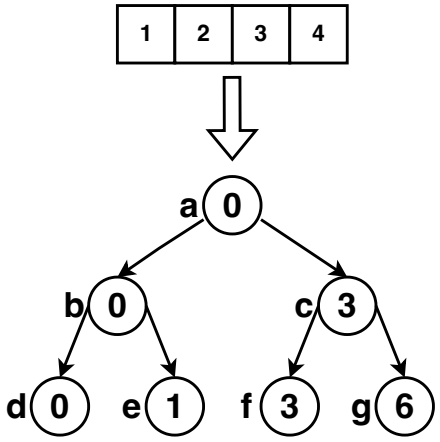


Figure 6: Example of prefix-sum tree