DIFFERENTIAL PRIVACY OF CROSS-ATTENTION WITH PROVABLE GUARANTEE

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-attention has become a fundamental module nowadays in many important artificial intelligence applications, e.g., retrieval-augmented generation (RAG), system prompt, guided stable diffusion, and many more. Ensuring cross-attention privacy is crucial and urgently needed because its key and value matrices may contain sensitive information about model providers and their users. In this work, we design a novel differential privacy (DP) data structure to address the privacy security of cross-attention with a theoretical guarantee. ~~In detail, let n be the input token length of system prompt/RAG data, d be the feature dimension, $0 < \alpha \leq 1$ be the relative error parameter, R be the maximum value of the query and key matrices, R_w be the maximum value of the value matrix, and r, s, ϵ_s be parameters of polynomial kernel methods. Then, our data structure requires $\widetilde{O}(ndr^2)$ memory consumption with $\widetilde{O}(nr^2)$ initialization time complexity and $\widetilde{O}(\alpha^{-1}r^2)$ query time complexity for a single token query. In addition, our data structure can guarantee that the process of answering user query satisfies (ϵ, δ)-DP with $\widetilde{O}(n^{-1}\epsilon^{-1}\alpha^{-1/2}R^{2s}R_w r^2)$ additive error and $n^{-1}(\alpha + \epsilon_s)$ relative error between our output and the true answer.~~ Furthermore, our result is robust to adaptive queries in which users can intentionally attack the cross-attention system. To our knowledge, this is the first work to provide DP for cross-attention and is promising to inspire more privacy algorithm design in large generative models (LGMs).

1 INTRODUCTION

The development of Artificial Intelligence (AI) has four stages: (1) prediction AI, e.g., ResNet (He et al., 2016) in image classification; (2) generation AI, e.g., ChatGPT (Achiam et al., 2023) in language generation; (3) autonomous agent AI, Voyager (Wang et al., 2023a) autonomously plays Minecraft game (Fan et al., 2022); (4) Artificial Generalization Intelligence (AGI). Humans have made rapid progress in generative AI, and we are excitingly heading to the third stage, the era of AI agent (Liu et al., 2023). One prevalent application of AI agents is customized large generative models (LGMs) agents (OpenAI, 2024a), e.g., AgentGPT (GitHub, 2024a), SuperAGI (GitHub, 2024d), MetaGPT (Hong et al., 2024b;a), GPT Researcher (GitHub, 2024c) and many so on. In particular, recently, Apple Inc. introduced Apple Intelligence (Apple, 2024), signaling the integration of LGMs into physical devices. This innovation allows devices to use personal information for real-life assistance, such as entering passport numbers when booking flights or informing users of their latest meetings. With increased AI capabilities, privacy concerns become significant, as the more personal information devices handle, the greater the potential privacy risks.

One fundamental technique used in LGMs is cross-attention (Vaswani et al., 2017), which is an essential module in retrieval-augmented generation (RAG) (Lewis et al., 2020), system prompt, guided stable diffusion, and many so on. In RAG, to be more professional, the LGMs answer user input queries by using a domain-specific database under cross-attention, which may contain specific privacy data and knowledge so that the LGMs gain additional power. For system prompts, based on cross-attention, some customized long prompts, e.g., user information or concrete rules, are concatenated before user input to follow human instructions better, which are commonly used in ChatGPT (GitHub, 2024b), Claude3 (Anthropic, 2024) and other commercial LGMs.

Consequently, protecting the privacy of domain-specific data in RAG or system prompts is crucial as they contain sensitive information about users and companies. These data and prompts are the core assets of many start-ups. However, these data and prompts can be easily recovered (Li et al., 2023b), jailbroken (Jin et al., 2024), and released (Li et al., 2023a) by user adversarial attack (Yu et al., 2024), e.g., there are 1700 tokens in ChatGPT system prompts (Patel, 2024). These findings highlight the critical importance of robust privacy protections in LGMs, making privacy not just essential but an urgent issue that demands immediate attention.

To fundamentally preserve cross-attention privacy, we borrow the powerful tools from differential privacy (DP) (Dwork et al., 2006), which provides measurable privacy and combines with statistical machine learning seamlessly (Ponomareva et al., 2023). Thus, in this work, we would like to ask and answer the following question,

How can we use differential privacy to protect the security of cross-attention in LGMs?

Our work demonstrates that the Softmax cross-attention computation is equivalent to computing the weighted distance problem.

Definition 1.1 (Softmax cross-attention)**.** *Let n and m be the token length of the data and input query, respectively. Let d be the feature dimension. Given fixed key matrix $K \in [0, R]^{n \times d}$ and fixed value matrix $V \in [-R_w, R_w]^{n \times d}$, for any input query matrix $Q \in [0, R]^{m \times d}$, the goal of the Softmax Cross-Attention Computation is to get the matrix $\mathrm{Attn}(Q, K, V) \in \mathbb{R}^{m \times d}$, which is*

$$\mathrm{Attn}(Q, K, V) := D^{-1} A V,$$

where $A \in \mathbb{R}^{m \times n}$ satisfies $A_{i,j} := \exp(\langle Q_i, K_j \rangle / d)$ for any $i \in [m], j \in [n]$ (Q_i and K_j denote the i-th and j-th rows of Q and K, respectively) and $D := \mathrm{diag}(A \mathbf{1}_n) \in \mathbb{R}^{m \times m}$ is a diagonal matrix.

Note that $\mathsf{Softmax}(QK^\top / d) = D^{-1} A \in \mathbb{R}^{m \times n}$ in Definition 1.1, which is the standard function used in transformers, and usually, we call it as attention matrix. Our main theorem, presented below, provides a robust solution of cross-attention, ensuring privacy and accuracy guarantees.

Theorem 1.2 (Main result; Informal version of Theorem 3.1)**.** *Let Q, K, V, Attn be defined in Definition 1.1. Let $\alpha \in (0, 1)$ be the relative error parameter and p_f be the probability of failure parameter. Let r, s, ϵ_s be the parameters of the polynomial kernel methods (Lemma C.7). Then, our Algorithm 1 requires $\widetilde{O}(ndr^2)$ memory with $\widetilde{O}(ndr^2)$ initialization time and $\widetilde{O}(\alpha^{-1} dr^2)$ query time, such that with probability $1 - p_f$, the output process of cross-attention satisfies (ϵ, δ)-DP and is robust to adaptive query with error $\widetilde{O}(n^{-1}\epsilon^{-1} R \exp(R^2 + 2R\epsilon^{-1})((1 + \alpha + \epsilon_s) \cdot (AV)_{i,k} + \epsilon^{-1}\alpha^{-1/2} l \Gamma_{R,s}^2 R_w r \sqrt{\log(l/\delta')}))$.*

Our main technique in Theorem 1.2 ensures that cross-attention is differentially private by using the polynomial kernel approximation method and transforming it into a weighted distance problem. We then solve the problem by summing over weighted distances (depending on the value embedding) between the query embedding and the key embedding. We build a data structure for weighted Softmax queries in Section 4.3, and we extend this data structure to handle adaptive queries using the ϵ_0-net/metric entropy argument in Section 4.4. Furthermore, our error decreases as the input token length grows, diminishing both the relative and additive errors to zero.

Our contributions are as follows:

- We demonstrate that cross-attention computations are equivalent to the weighted distance problem (Section 3).
- We design a novel algorithm (Algorithm 3) that privately answers weighted Softmax queries with high probability and a concrete accuracy bound.
- Our algorithm (Algorithm 1) handles multiple cross-attention queries and is robust against adaptive query attacks (Theorem 3.1), meaning that potential attackers cannot intentionally extract information of system prompts/RAG data.

To our knowledge, this is the first work to utilize DP to protect prompts in LGMs with theoretically provable guarantees. While some have explored protecting user/system prompts with DP (Edemacu & Wu, 2024; Mai et al., 2023), they are primarily empirical and lack theoretical guarantees. Additionally, many others are working on protecting private datasets by applying DP to the fine-tuning stage of LGMs (Behnia et al., 2022; Singh et al., 2024; Liu et al., 2024b; Yu et al., 2021; Li et al., 2021; Shi et al., 2022a), which diverges from our work. The strength of DP lies in its strong, unambiguous, and concrete definition of privacy, enabling algorithm designs with provable privacy and accuracy analysis. Therefore, we believe that the theoretical aspects of DP applications in LGMs remain a highly impactful direction, and we aim to pave the way for further exploration in this area.

1.1 Related Work

Differential Privacy in Data Structure and Attention. Differential privacy (DP) is a flourishing and powerful technique that has enormous applications in the topic of private machine learning. In the era of Large Generative Models (LGMs), there are three primary approaches to ensuring privacy: (1) during the pre-training stage: to protect training data (Abadi et al., 2016; Ponomareva et al., 2023), (2) during the adaptation stage: to protect target data (Behnia et al., 2022; Singh et al., 2024; Liu et al., 2024b; Yu et al., 2021; Li et al., 2021; Shi et al., 2022a; Huang et al., 2024), (3) during the inference stage: to protect user/system prompts (Edemacu & Wu, 2024) and RAG data (Lewis et al., 2020). To protect training data, DP-SGD (Abadi et al., 2016) uses DP optimizer to ensure data privacy, severing as the traditional baseline method. Recently, numerous works have aimed to improve this method by integrating DP in both the pre-training and fine-tuning stages of LGMs (Yu et al., 2021; Li et al., 2021; Golatkar et al., 2022; Behnia et al., 2022; Shi et al., 2022a; Mattern et al., 2022; Singh et al., 2024; Zheng et al., 2024; Liu et al., 2024b). However, DP-SGD confines differential privacy to the optimizer. In contrast, we propose a novel approach that integrates DP directly into the attention mechanism, supported by strong theoretical analysis and guarantees. Given the resource-intensive nature of training LGMs, our technique offers a practical alternative for models trained with standard SGD, which lack inherent privacy guarantees. In such cases, applying DP-SGD would require retraining the models, which is computationally expensive, whereas our method avoids this additional cost.

To protect user/system prompts, Edemacu & Wu (2024) provides a survey on both DP and non-DP methods. In the use of LGMs, prompting methods almost become a standard way for inference (Schulhoff et al., 2024). Given the billions of prompt interactions daily, ensuring privacy is essential (Mai et al., 2023). We refer readers to Appendix A for more related works.

Roadmap. In Section 2, we present the preliminary of differential privacy (DP) and cross-attention. In Section 3, we present the main result of our cross-attention theorem (Theorem 3.1). In Section 4, we outline the main results of our algorithms. In Section 5, we discuss DP-related topics and potential extensions. In Section 6, we conclude our paper.

2 Preliminary

In this section, we give the preliminary of differential privacy (DP) and cross-attention. In Section 2.1, we describe the notations. In Section 2.2, we give definitions related to DP.

2.1 Notations

We use $\Pr[]$ to denote the probability. We use $\mathbb{E}[]$ to denote the expectation. We use $\mathrm{Var}[]$ to denote the variance. For two vectors $x \in \mathbb{R}^d$ and $y \in \mathbb{R}^d$, we use $\langle x, y \rangle$ to denote the inner product between x, y, i.e., $\langle x, y \rangle = \sum_{i=1}^d x_i y_i$. We use $X \subset \mathbb{R}^d$ and $|X| = n$ to mean the same thing as $X \in \mathbb{R}^{n \times d}$. Also, we denote $x_i^\top$ as the i-th row of X. We use $x_{i,j}$ to denote the j-th coordinate of $x_i \in \mathbb{R}^n$. We use $\mathbf{1}_n$ to denote a length-n vector where all the entries are ones. We use $\|x\|_p$ to denote the ℓ_p norm of a vector $x \in \mathbb{R}^n$, i.e., $\|x\|_1 := \sum_{i=1}^n |x_i|$, $\|x\|_2 := (\sum_{i=1}^n x_i^2)^{1/2}$, and $\|x\|_\infty := \max_{i \in [n]} |x_i|$. We denote polynomial time complexity with respect to n as $\mathrm{poly}(n)$. For a function f, we use $\widetilde{O}(f)$ to represent f multiplied by a polylogarithmic factor, i.e., $f \cdot \mathrm{poly}(\log f)$. This notation, known as soft-O or tilde notation, simplifies expressions by omitting logarithmic factors, focusing on the dominant term's growth rate.

2.2 Differential Privacy Definitions

In this section, we give several definitions related to differential privacy (DP). We refer the reader to Dwork & Roth (2014) for more background and details on DP.

Definition 2.1 (Neighboring dataset)**.** *Two datasets $X, X' \in [0, R]^{n \times d}$ are neighboring if they differ in exactly one row, i.e., there exists $i \in [n]$ such that $X_{i,*} \neq X'_{i,*}$ and $X_{j,*} = X'_{j,*}$ for all $j \neq i$.*

Definition 2.2 (Sensitivity)**.** *The sensitivity of a function $f : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d'}$ is: $\Delta := \max_{X, X' \in \mathbb{R}^{n \times d}} \|f(X) - f(X')\|_1$, where X, X' are neighboring datasets and $\|\cdot\|_1$ is the entry-wise ℓ_1-norm.*

Definition 2.3 ((ϵ, δ)-DP)**.** *For $\epsilon > 0, \delta \geq 0$, a randomized algorithm $\mathcal{A}$ is (ϵ, δ)-DP, if for all $\mathcal{S} \subseteq \mathrm{Range}(\mathcal{A})$ and for all neighboring datasets X, X' ~~such that $\|X - X'\|_1 \leq 1$~~:*

$$\Pr[\mathcal{A}(X) \in \mathcal{S}] \leq \exp(\epsilon) \Pr[\mathcal{A}(X') \in \mathcal{S}] + \delta.$$

When $\delta = 0$, the algorithm is said to have pure differential privacy.

We mainly use the truncated Laplace mechanism, which has the following definitions.

Definition 2.4 (Truncated Laplace distribution)**.** *We use $\mathrm{TLap}(\Delta, \epsilon, \delta)$ to denote the Truncated Laplace distribution with pdf proportional to $\exp(-\epsilon |z| / \Delta)$ on the region $[-B, B]$, where $B = \frac{\Delta}{\epsilon} \cdot \log(1 + \frac{\exp(\epsilon) - 1}{2\delta})$.*

Fact 2.5 (Theorem 3 in Geng et al. (2020))**.** *Let z denote a $\mathrm{TLap}(\Delta, \epsilon, \delta)$ random variable. Then we have $\mathbb{E}[z] = 0$, and*

$$\mathrm{Var}[z] = \frac{2\Delta^2}{\epsilon^2}(1 - \delta \cdot \frac{\log^2(1 + \frac{e^\epsilon - 1}{2\delta}) + 2\log(1 + \frac{e^\epsilon - 1}{2\delta})}{e^\epsilon - 1}).$$

Furthermore, if $\delta = 0$, we have $\mathrm{Var}[z] = 2\Delta^2 / \epsilon^2$, meaning truncated Laplacian mechanism will be reduced to the standard Laplacian mechanism.

Lemma 2.6 (Laplace mechanism, (Dwork & Roth, 2014; Geng et al., 2020), see Lemma 2.2 in Andoni et al. (2023))**.** *Given a numeric function f that takes a dataset X as the input, and has sensitivity Δ, the mechanism that outputs $f(X) + z$ where $z \sim \mathrm{Lap}(\Delta/\epsilon)$ is $(\epsilon, 0)$-DP. In addition, if $\epsilon, \delta \in (0, 0.5)$, $f(X) + z$, where $z \sim \mathrm{TLap}(\Delta, \epsilon, \delta)$ is (ϵ, δ)-DP. Moreover, the truncated Laplace mechanism is always accuracy up to error B.*

Algorithm 1 DP cross-attention algorithm

datastrucutre DPCrossAttention ▷ Theorem 3.1
members
  $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_d$: DPTreeSoftmaxAdaptive ▷ Algorithm 8
end members
procedure Init($K \in [0, R]^{n \times d}$, $V \in [-R_w, R_w]^{n \times d}$, $\epsilon \in (0, 1)$, $\delta \in (0, 1)$, $\delta' \in (0, 1)$, $c \in (0, 0.1), \epsilon_s \in (0, 0.1), p_f \in (0, 0.01)$) ▷ $n = |K|$
  for $k = 1 \to d$ **do**
    $\mathcal{D}_k.\textsc{Init}(K, n, V_{:,k}, \epsilon/2, \delta/2, \delta'/2, c, \epsilon_s, p_f)$ ▷ Compute AV
  end for
  $\mathcal{D}_0.\textsc{Init}(K, n, \mathbf{1}_n, \epsilon/2, \delta/2, \delta'/2, c, \epsilon_s, p_f)$ ▷ Compute D
end procedure
procedure Query($Q_i \in [0, R]^d$, $\alpha \in (0, 1)$)
  $O \leftarrow 0^d$
  $D \leftarrow \mathcal{D}_0.\textsc{DistanceQuery}(Q_i, \alpha)$
  for $k = 1 \to d$ **do**
    $O_k \leftarrow D^{-1} \cdot \mathcal{D}_k.\textsc{DistanceQuery}(Q_i, \alpha)$
  end for
  return O
end procedure
end datastrucutre

3 MAIN RESULTS: CROSS-ATTENTION

In this section, we show our main result for cross-attention. Theorem 3.1 states that we can ensure the entire cross-attention module satisfies DP and is robust to adaptive queries. Our high-level idea is based on the similarity between weighted distance problem and cross-attention. For a typical weighted distance problem, we define the following: Let $w \in \mathbb{R}^n$ be the weights, $X \in \mathbb{R}^{n \times d}$ be the data matrix, where $x_i^\top$ is the i-th row of X for $i \in [n]$, and let $y \in \mathbb{R}^d$ be the query. Suppose we need to answer ℓ_1-distance query. We have

$$\sum_{i \in [n]} \underbrace{w_i}_{\text{weight}} \| \underbrace{y}_{\text{query}} - \underbrace{x_i}_{\text{data}} \|_1.$$

Now we introduce cross-attention. Let Q, K, V, Attn be defined in Definition 1.1. In a standard cross-attention process, K and V are accessible before inference, while the user input Q becomes available only when the user provides it. Here, K and V represent values stored in memory or disks and are considered private assets protected within the model, whereas Q is treated as public.

For the cross-attention mechanism Attn (Definition 1.1), we aim to ensure that the matrix AV satisfies DP guarantee. Let $A_{i,j} = \exp(\langle Q_i, K_j \rangle / d)$ for $i \in [m], j \in [n]$. Let $V_{j,k} \in \mathbb{R}$ be the (j,k)-th entry of V, for $j \in [n], k \in [d]$. ~~Let $D = \mathrm{diag}(A\mathbf{1}_n)$, acting as a normalizing factor that aggregates all the information. We store both K and its corresponding noises. For computing AV, we use the perturbed K, whereas for computing D, we rely on the original, unperturbed K. By post-processing property (Fact B.7), to ensure that the forward output $\mathrm{Attn}(Q, K, V) = D^{-1}AV$ (Definition 1.1) satisfies DP, we only need to ensure the DP of its component AV.~~

The (i,k)-th entry of AV for each $i \in [m], k \in [d]$ is computed by

$$(AV)_{i,k} = \sum_{j=1}^{n} \underbrace{V_{j,k}}_{\text{weight}} \exp(\langle \underbrace{Q_i}_{\text{query}}, \underbrace{K_j}_{\text{data}} \rangle / d), \tag{1}$$

which can be viewed as a weighted Softmax problem, where V provides the weights, Q is the query, and K is the dataset. Thus, we choose to add noise to K and V based on the similarity between the weighted distance problem and cross-attention. Furthermore, we find that we can only handle one column of V, i.e., $V_{*,k} \in \mathbb{R}^n$, in a single data structure. Therefore, we need to initialize a total of d different data structures, each with weights $V_{*,k}$ for $k \in [d]$.

Here, we present our main result below.

Theorem 3.1 (Softmax cross-attention, informal version of Theorem I.12)**.** *Let Q, K, V, Attn be defined in Definition 1.1. Let $\alpha \in (0,1)$ be the relative error parameter and p_f be the probability of failure parameter. Let r, s, ϵ_s be parameters of polynomial kernel methods (Lemma C.7). Let $\Gamma_{R,s} := \max_{j \in [s]} \frac{R^j}{\sqrt{j!}}$ (Definition I.3). Let $l = O(r \log(dR/(\epsilon_s p_f)))$. There is a data structure* DPTREECROSSATTENTION *(Algorithm 1) that uses $O(lnrd)$ spaces to ensure cross-attention DP and supports the following operations:*

- INIT$(K, V, \epsilon \in (0,1), \delta \in (0,1), \delta' \in (0,1), c \in (0, 0.1), \epsilon_s \in (0, 0.1), p_f \in (0, 0.01))$ *(Algorithm 1). It takes $O(lnrd)$ time to initialize.*
- *At query time, for user input Q, we process one token at a time by passing the i-th row of Q, denoted $Q_i \in [0, R]^d$, to* QUERY$(Q_i, \alpha \in (0,1))$ *(Algorithm 1) for each $i \in [m]$. It takes $O(\alpha^{-1} ldr \log^2 n)$ time to output an entry z in $\mathrm{Attn}(Q, K, V)$ such that*
  - *the process of output z satisfies $(\epsilon, \delta + \delta')$-DP,*
  - *the process of output z has error*

    $$\widetilde{O}(n^{-1}\epsilon^{-1} R \exp(R^2 + 2R\epsilon^{-1})((1 + \alpha + \epsilon_s) \cdot (AV)_{i,k} + \epsilon^{-1}\alpha^{-1/2} l \Gamma_{R,s}^2 R_w r \sqrt{\log(l/\delta')}))$$

    where $\widetilde{O}$ hide logarithm dependecy on n,
  - *it holds with probability $1 - p_f$ (where p_f is used in l),*
  - *it is robust to adaptive query.*

In Theorem 3.1, we use our DPTREECROSSATTENTION (Algorithm 1) and guarantee that, for each query token of cross-attention, the output process satisfies $(\epsilon, \delta + \delta')$-DP with error $\widetilde{O}(n^{-1}\epsilon^{-1}R\exp(R^2 + 2R\epsilon^{-1})((1+\alpha+\epsilon_s)\cdot(AV)_{i,k} + \epsilon^{-1}\alpha^{-1/2}l\Gamma_{R,s}^2 R_w r\sqrt{\log(l/\delta')}))$ ~~$n^{-1}(\alpha+\epsilon_s)$ relative error and $O(n^{-1}\epsilon^{-1}\alpha^{-1/2}l\Gamma_{R,s}^2 R_w r\sqrt{\log(l/\delta')}\cdot\log^{3/2} n)$ additive error~~, and $O(\alpha^{-1}ldr\log^2 n)$ running time under adaptive query. More specifically, the algorithm creates $d+1$ DPTREESOFTMAXADAPTIVE (Algorithm 8) data structures, each requiring $O(lnr)$ memory consumption and $O(lnr)$ initialization time. Notably, our error is inversely proportional to n, meaning that as the input token length increases, both the relative and approximate errors approach zero. ~~This is achieved by the normalizing matrix D (Definition 1.1).~~ We refer the reader to Section I for proof details.

Thus, our algorithm theoretically protects system prompts/RAG data in cross-attention as discussed in Section 1. In Section 4, we provide a detailed technical overview, and in Section 5, we will present self-attention and DP-related discussion.

Algorithm 2 DPTree initialization and query

datastructure DPTREE ▷ Theorem C.1
members
  $b : \mathbb{R}^{2n-1}$
  $c : \mathbb{R}^{2n-1}$
end members
procedure INIT($a \in \mathbb{R}^n, n \in \mathbb{N}_+, \Delta \in \mathbb{R}, \epsilon \in (0,1), \delta \in (0,1)$) ▷ Lemma D.3, Lemma C.3
  $b[n, 2n-1] \leftarrow a$
  for $i = n \to 2n-1$ **do**
    $c[i] \leftarrow b[i] + \mathrm{TLap}(\Delta, \epsilon/\log n, \delta/\log n)$
  end for
  for $i = (\log n) \to 1$ **do**
    for $j = 1 \to 2^{i-1}$ **do**
      $k \leftarrow 2^{i-1} + j - 1$
      $b[k] \leftarrow b[2k] + b[2k+1]$
      $c[k] \leftarrow b[k] + \mathrm{TLap}(\Delta, \epsilon/\log n, \delta/\log n)$
    end for
  end for
end procedure
procedure QUERY($x \in [n], y \in [n]$) ▷ Lemma D.4, D.5, D.6
  Trace from bottom nodes of x and y to find their lowest common ancestor, then we report the summation (based on c) by using at most $2\log n$ nodes on the path. Let Value be the above summation.
  return Value
end procedure
procedure TRUEQUERY($x \in [n], y \in [n]$)
  Trace from bottom nodes of x and y to find their lowest common ancestor, then we report the summation (based on b) by using at most $2\log n$ nodes on the path, where the height of the tree is $\log n$, and we need left and right boundary points. Let Value be the above summation.
  return Value
end procedure
end datastructure

4 KEY DATA STRUCTURE: DPTREE

This section provides our key data structures: DPTREE (Algorithm 2), DPTREEDISTANCE (Algorithm 5 and 6), DPTREEHIGHDIM (Algorithm 7), DPTREESOFTMAX (Algorithm 3), and DPTREESOFTMAXADAPTIVE (Algorithm 8).

In Section 4.1, we provide our high-level proof insights. In Section 4.2, we give our basic building block algorithms DPTREE, DPTREEDISTANCE and DPTREEHIGHDIM. In Section 4.3, we present our DPTREESOFTMAX algorithm that solves the weighted Softmax problem. In Section 4.4, we present our DPTreeSoftmaxAdaptive algorithm that enables DPTreeSoftmax to handle adaptive query problem.

4.1 Technique Overview

Notice that Eq. (1) is not a typical distance measure like ℓ_1 or ℓ_2, but by using polynomial kernel method techniques, we transform it into a distance measure. Alman & Song (2023) states that the exponential inner product can be approximated by polynomial kernel function $P(\cdot) : \mathbb{R}^d \to \mathbb{R}^r$, i.e., $P(x)^\top P(y) \approx \exp(x^\top y/d)$ for two vector $x, y \in \mathbb{R}^d$, with a relative error. Then, by the Law of Cosines, we transform the inner product of polynomial kernel functions into a distance measure, i.e.,

$$2P(x)^\top P(y) = -\|P(x) - P(y)\|_2^2 + \|P(x)\|_2^2 + \|P(y)\|_2^2. \tag{2}$$

After transforming Eq. (1) into a distance measure, we design the DPTree series data structures to provide cross-attention DP guarantee.

In summary, we first design the data structure DPTree (Algorithm 2) that builds a binary segment tree with truncated Laplace noise added in the leaf nodes to ensure DP guarantee. Then, based on this data structure, we design DPTreeDistance (Algorithm 5 and 6) to answer one dimensional weighted distance queries $\sum_{i=1}^n w_i \cdot |y - x_i|$, which utilizes DPTree to store and return noised weights w_i multiplied with the approximated distances between the query y and data x_i. We further decompose high dimensional ℓ_p^p-distance problem into one dimensional ℓ_1-distance problems using

$$\sum_{i=1}^n w_i \cdot \|y - x_i\|_p^p = \sum_{k=1}^d \sum_{i=1}^n w_i \cdot |y_k - x_{i,k}|^p. \tag{3}$$

Based on this decomposition, we design DPTreeHighDim (Algorithm 7) which is capable of answering high dimension queries. Then, using Eq. (2) and DPTreeHighDim, we design DPTreeSoftmax (Algorithm 3) to answer Softmax queries. By building multiple copies of this data structure, we boost the success probability such that it can answer any query (including adaptive query) with an additive error, establishing the final data structure DPTreeCrossAttention (Algorithm 1). See Section C for a more detailed outline of algorithms and proof techniques.

4.2 DPTree, DPTreeDistance, and DPTreeHighDim

The unweighted distance query has been explored in prior works (Huang & Roth, 2014; Backurs et al., 2024; Liu et al., 2024a). Specifically, Huang & Roth (2014) leverages online learning techniques to approximate the sum of distances, while Backurs et al. (2024) introduces a DP data structure based on a node-contaminated balanced binary tree. Furthermore, Liu et al. (2024a) presents a new data representation in tree nodes, where each node stores the sum of distances from one point to multiple points. In contrast, we focus on the weighted distance query, generalizing their results.

We design a basic data structure DPTree (Algorithm 2) that answers summation queries by a summation segment tree with truncated Laplace noise (Definition 2.4). The algorithm first builds a binary summation tree in an array and then adds truncated Laplace noises to each node. In query time, we first trace from bottom nodes to find their lowest common ancestor, then report the summation by using at most $2 \log n$ nodes on the path (Algorithm 2). Based on the parallel composition rule of DP (Fact B.9), we find that if we have multiple disjoint interval queries, the error of the weighted sum of the intervals can be bounded independently of the number of queries (Lemma D.8). See more details in Section D.

We then design DPTreeDistance, a one-dimensional weighted ℓ_1 distance data structure detailed in Algorithm 5 and 6. Initialization involves rounding each data point to the nearest multiple of a small interval and aggregating their weights into an array (illustrated in Figure 1), which is then input into our DPTree. At query time, we retrieve aggregated weights within small intervals and multiply them by their distances to the query point. We introduce a relative error parameter α to reduce the number of iterations to $O(\log(n)/\alpha)$, improving efficiency. Guided by Eq.(3), we design DPTreeHighDim (Algorithm 7), which extends DPTreeDistance to higher dimension by constructing independent data structures for each coordinate. See details in Section F and G.

Algorithm 3 Softmax query

datastrucutre DPTreeSoftmax ▷ Theorem 4.2
members
  $\mathcal{D}_0, \mathcal{D}_1, \dots, \mathcal{D}_r$: DPTreeDistance ▷ Algorithm 5, Theorem I.7
  $P : [0, \Gamma_{R,s}]^{n \times r}$ ▷ Definition I.3 for $\Gamma_{R,s}$, Eq. (9) for s, Eq. (10) for r
  $w : [-R_w, R_w]^n$
  $P_{wx},\ s_w,\ \epsilon_s : \mathbb{R}$
end members
procedure Init($X \subset [0,R]^d$, $n \in \mathbb{N}_+$, $w \in [-R_w, R_w]^n$, $\epsilon \in (0,1)$, $\delta \in (0,1)$, $\delta' \in (0,1)$, $c \in (0, 0.1)$, $\epsilon_s \in (0, 0.1)$) ▷ Lemma C.7
  $\epsilon_s,\ w,\ P,\ P_{wx},\ s_w \leftarrow \epsilon_s,\ w,\ 0^{n \times r},\ 0,\ 0$
  for $j = 1 \to n$ **do**
    Compute $P(x_j)$ ▷ Polynomial kernel function $P(\cdot)$, Lemma I.5
    ~~Compute~~ $\cancel{w_j \|P(x_j)\|_2^2}$
    $\cancel{P_{wx} \leftarrow P_{wx} + w_j \|P(x_j)\|_2^2}$
    $\cancel{s_w \leftarrow s_w + w_j}$
    $P_{j,:} \leftarrow P(x_j)$
  end for
  for $i = 1 \to r$ **do**
    $\cancel{\mathcal{D}_i.\textsc{Init}(P_{:,i},\ n,\ w,\ c\epsilon/\sqrt{r \log(1/\delta')},\ \delta/r)}$ ~~▷ Algorithm 5~~
    $\mathcal{D}_i.\textsc{Init}(P_{:,i},\ n,\ w,\ \frac{c\epsilon}{3\sqrt{r \log(2/\delta')}},\ \frac{\delta}{3r})$ ▷ Algorithm 5
    $P_{wx} \leftarrow P_{wx} + \mathcal{D}_i.\textsc{DistanceQuery}(0, \alpha)$
  end for
  $\mathcal{D}_0.\textsc{Init}(\mathbf{1}_n,\ n,\ w,\ \epsilon/3,\ \delta/3)$
  $s_w \leftarrow s_w + \mathcal{D}_0.\textsc{DistanceQuery}(0, \alpha)$
end procedure
procedure DistanceQuery($y \in [0,R]^d$, $\alpha \in (0,1)$) ▷ Lemma C.7
  Value $\leftarrow 0$
  Compute $P(y)$
  Compute $\|P(y)\|_2^2$
  for $i = 1 \to r$ **do**
    Value $\leftarrow$ Value + $\mathcal{D}_i.\textsc{DistanceQuery}(P(y)_i, \alpha)$ ▷ Algorithm 6
  end for
  Value $\leftarrow 0.5 \cdot (P_{wx} + s_w \|P(y)\|_2^2\ -$ Value)
  return Value
end procedure
end datastrucutre

4.3 Softmax Activation

In this section, we present DPTreeSoftmax (Algorithm 3) that answers the weighted Softmax query (Definition 4.1) and is further used to design DP cross-attention. First, we introduce the definition of weighted Softmax query, an abstraction for the problem described in Eq. (1).

Definition 4.1 (Weighted Softmax query (without normalization))**.** *For the dataset $X \in [0,R]^{n \times d}$ where $x_i^\top$ is the i-th row of X and query $y \in [0,R]^d$, we define the weighted exponential inner product/Softmax query to be:*

$$\sum_{i \in [n]} w_i \exp(\langle x_i, y \rangle / d) = w^\top \exp(Xy/d).$$

Building on Definition 4.1, we develop a novel algorithm to answer differentially private weighted Softmax queries using the polynomial kernel method from Alman & Song (2023). Specifically, in Eq.(2), there is a term that computes the weighted ℓ_2^2 distance, which we calculate using DPTreeHighDim. We then compute the exact term for the weighted ℓ_2^2 norms of the approximation kernel. By summing these terms with a controlled error, we extend DPTreeHighDim to answer the Softmax query efficiently. More details can be found in Section I.

Theorem 4.2 (Softmax query, informal version of Theorem I.8)**.** *Let $R \geq 1$. Let $r \leq \binom{2s+2d}{2s}$ and $s = O(\max\{\frac{\log(1/\epsilon_s)}{\log(\log(1/\epsilon_s)/R)}, R^2\})$. Let $\Gamma_{R,s} := \max_{j\in[s]} \frac{R^j}{\sqrt{j!}}$ (Definition I.3). Let the accuracy parameter be $\epsilon_s \in (0, 0.1)$. Our data structure* DPTreeSoftmax *(Algorithm 3) uses $O(nr)$ spaces to solve Softmax query problem for dataset $X \subset [0, R]^d$ and support following operations:*

- Init$(X \subset [0, R]^d, n \in \mathbb{N}_+, w \in [-R_w, R_w]^n, \epsilon \in (0, 1), \delta \in (0, 1), \delta' \in (0, 1), c \in (0, 0.1), \epsilon_s \in (0, 0.1))$. *(Algorithm 3) It takes $O(nr)$ time to initialize the data structure.*
- DistanceQuery$(y \in [0, R]^d, \alpha \in (0, 1))$. *(Algorithm 3) It takes $O(\alpha^{-1} r \log^2 n)$ time to output a number z such that*
  - *the process of output z satisfies $(\epsilon, \delta + \delta')$-DP private, which computes $w^\top \exp(Xy/d)$,*
  - *the error bound satisfies $|z - w^\top \exp(Xy/d)| \leq (\alpha + \epsilon_s) \cdot w^\top \exp(Xy/d)$ $+ O(\epsilon^{-1}\alpha^{-1/2}\Gamma_{R,s}^2 R_w r \sqrt{\log(1/\delta')} \cdot \log^{3/2} n)$,*
  - *it holds with probability at least 0.99.*

Remark 4.3. *In Theorem 4.2, the parameter ϵ_s is the accuracy parameter for polynomial kernel approximation described in Section C.5. Besides, note that the error bound in Theorem 4.2 does not depend on δ but depends on δ'. The role of δ is to control a hidden constant term in the big O notation, i.e., increasing δ reduces the error by a small constant (Fact 2.5). In practice, we set δ as a small positive constant close to 0. Please refer to the Lemma D.6 for more details.*

4.4 Adaptive Query Data Structure

We adapt our DPTreeSoftmax to DPTreeSoftmaxAdaptive (Algorithm 8) to solve the adaptive query problem. By proving it can handle any query within the query space with a certain error, we ensure it effectively processes adaptive queries. We first boost the constant probability to high probability using the Chernoff bound (Lemma B.2). Employing an ϵ_0-net argument and the union bound, we bound all query points within the net. Finally, we use the Lipschitz property of the weighted Softmax distance function with an additive error to bound all points in the query space. The corresponding proofs can be found in Section H and Section I.

Theorem 4.4 (Adaptive query Softmax data structure, informal version of Theorem I.11)**.** *Let $R \geq 1$. Let $r \leq \binom{2s+2d}{2s}$ and $s = O(\max\{\frac{\log(1/\epsilon_s)}{\log(\log(1/\epsilon_s)/R)}, R^2\})$. Let $\Gamma_{R,s} := \max_{j\in[s]} \frac{R^j}{\sqrt{j!}}$ (Definition I.3). Let the accuracy parameter be $\epsilon_s \in (0, 0.1)$. Let $X \in [0, R]^{n\times d}$ be the dataset, $w \in [-R_w, R_w]^n$ be weights, $y \in [0, R]^d$ be the query, $\alpha \in (0, 1)$ be the relative error parameter and p_f be the failure probability parameter. Let $l = O(r \log(dR/(\epsilon_s p_f)))$. There is a data structure* DPTreeSoftmaxAdaptive *(Algorithm 8) that uses $O(lnr)$ spaces to solve the weighted Softmax query problem for the dataset $X \subset [0, R]^d$ and supports the following operations:*

- Init$(X \subset [0, R]^d, n \in \mathbb{N}_+, w \in [-R_w, R_w]^n, \epsilon \in (0, 1), \delta \in (0, 1), \delta' \in (0, 1), c \in (0, 0.1), \epsilon_s \in (0, 0.1), p_f \in (0, 0.01))$. *It takes $O(lnr)$ time to initialize the data structure.*
- DistanceQuery$(y \in [0, R]^d, \alpha \in (0, 1))$. *It takes $O(\alpha^{-1} lr \log^2 n)$ time to output a number z such that*
  - *the process of output z satisfies $(\epsilon, \delta + \delta')$-DP private, which computes $w^\top \exp(Xy/d)$,*
  - *the error bound satisfies $|z - w^\top \exp(Xy/d)| \leq (\alpha + \epsilon_s) \cdot w^\top \exp(Xy/d)$ $+ O(\epsilon^{-1}\alpha^{-1/2} l \Gamma_{R,s}^2 R_w r \sqrt{\log(l/\delta')} \cdot \log^{3/2} n)$,*
  - *it holds with probability at least $1 - p_f$ (where p_f is used in l),*
  - *it is robust to adaptive query.*

Remark 4.5. *We describe the parallelization of our algorithms. In the second for loop of* Init *and the for loop of* DistanceQuery *in Algorithm 3, the r* DPTreeDistance *data structures instantiated for each coordinate are independent of each other. In addition, the for loops in Algorithm 8 are also parallelizable since the $l = O(r \log(dR/(\epsilon_s p_f)))$ copies are independent. After parallelization, we have the final time complexity of* Init *to be $O(nr)$ and* DistanceQuery *to be $O(\alpha^{-1} \log^2 n)$ in Algorithm 8 with $O(lr)$ GPU process.*

5 Discussion

How do we extend to self-attention and other data structures? As self-attention is a more fundamental module in LGMs, we would like to extend our data structure to this setting. However, the challenge we faced was the dynamic update in tree nodes for each query for self-attention, which our current analysis does not support. How we can solve this challenge is crucial, and we leave it as our future direction.

Moreover, we observe that Li et al. (2015) introduces the DP matrix mechanism, which offers an alternative to our currently used binary tree data structure. A preliminary idea for extending this is as follows: consider $A = \exp(QK^\top/d)$ as defined in Definition 1.1, where Q of size $m \times d$ represents the query matrix with m linear queries, and K serves as the database. Leveraging the results from Li et al. (2015), we could design an alternative algorithm to enhance the current binary tree data structure, DPTree. We leave this exploration for future work.

Why not add noise to some other places? Where and how to add DP noises is an important problem to ask during the DP algorithm design. In this paper, we consider the problem of $\sum_{i=1}^n w_i \exp(\langle x_i, y\rangle/d)$ where $y, x_i \in [0, R]^d$ and $w \in [-R_w, R_w]^n$ (Definition 4.1). Notice that the only place where we add noises is in the most basic building block data structure DPTree (Algorihtm 2). From Lemma C.3 and the way we initialize DPTree in Algorithm 5, we see that the sensitivity Δ of this problem is $2R_w$.

A simple method for adding noise involves adding n noises to a length n array, with each item $w_i \exp(\langle x_i, y\rangle/d)$ for $i \in [n]$. However, this approach increases the error by a factor of n by basic composition (Fact B.8) and also makes the model dependent on the number of queries. Besides, it only supports a single query and requires rebuilding the tree for each new query, rendering it impractical. In contrast, our current noise-adding technique (Lines 9 and 15 of Algorithm 2) utilizes a summation tree such that the error only increases by a factor of $\mathrm{poly}\log n$. This method also supports multiple queries, eliminating the need to rebuild the tree each time.

How to remove the relative error parameter α? The relative error parameter α in Theorem 3.1 appears because of the $(1+\alpha)$-approximation introduced in Algorithm 5 (Remark F.3) to reduce the number of required iterations from naive $O(n)$ to $O(\log(n)/\alpha)$. However, we notice that a recent work (Liu et al., 2024a) does not utilize $(1+\alpha)$-approximation and still achieves $O(\log n)$ iteration number. They introduce a new tree node representation where each node stores the sum of distances from one point to multiple points, enabling the answer to be divided into only $\log n$ values, each combining two distance values, two count values, and y itself. Our DPTree algorithms can be integrated with their method, thus removing parameter α.

6 Conclusion

To our knowledge, we are the first work to provide differential privacy for cross-attention. This paper presents the DPTree data structures, which provide a differential privacy guarantee for the cross-attention module in large generative models. This is achieved by transforming the cross-attention mechanism into a weighted distance problem. Furthermore, our algorithm is robust to adaptive queries, allowing users to interact with the model arbitrarily without extracting sensitive information from the system prompts or RAG data. Our results may inspire more privacy algorithm design in large generative models.

References

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

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.

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

Josh Alman and Zhao Song. The fine-grained complexity of gradient computation for training large language models. arXiv preprint arXiv:2402.04497, 2024a.

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, 2024b.

Alexandr Andoni, Piotr Indyk, Sepideh Mahabadi, and Shyam Narayanan. Differentially private approximate near neighbor counting in high dimensions. In Advances in Neural Information Processing Systems (NeurIPS), pp. 43544–43562, 2023.

Anthropic. System prompts, 2024. https://docs.anthropic.com/en/docs/system-prompts.

Apple. Apple intelligence, 2024. https://www.apple.com/apple-intelligence/.

Arturs Backurs, Zinan Lin, Sepideh Mahabadi, Sandeep Silwal, and Jakub Tarnawski. Efficiently computing similarities to private datasets. arXiv preprint arXiv:2403.08917, 2024.

Rouzbeh Behnia, Mohammadreza Reza Ebrahimi, Jason Pacheco, and Balaji Padmanabhan. Ew-tune: A framework for privately fine-tuning large language models with differential privacy. In 2022 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 560–566. IEEE, 2022.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In International conference on machine learning, pp. 2206–2240. PMLR, 2022.

Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In Proceedings of the IEEE/CVF international conference on computer vision, pp. 357–366, 2021.

Justin Y Chen, Shyam Narayanan, and Yinzhan Xu. All-pairs shortest path distances with differential privacy: Improved algorithms for bounded and unbounded weights. arXiv preprint arXiv:2204.02335, 2022.

Yeshwanth Cherapanamjeri, Sandeep Silwal, David P Woodruff, Fred Zhang, Qiuyi Zhang, and Samson Zhou. Robust algorithms on adaptive inputs from bounded adversaries. arXiv preprint arXiv:2304.07413, 2023.

Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. The Annals of Mathematical Statistics, pp. 493–507, 1952.

Vincent Cohen-Addad, Alessandro Epasto, Vahab Mirrokni, Shyam Narayanan, and Peilin Zhong. Near-optimal private and scalable k-clustering. Advances in Neural Information Processing Systems, 35:10462–10475, 2022a.

Vincent Cohen-Addad, Chenglin Fan, Silvio Lattanzi, Slobodan Mitrovic, Ashkan Norouzi-Fard, Nikos Parotsidis, and Jakub M Tarnawski. Near-optimal correlation clustering with privacy. Advances in Neural Information Processing Systems, 35:33702–33715, 2022b.

Itai Dinur, Uri Stemmer, David P Woodruff, and Samson Zhou. On differential privacy and adaptive data analysis with bounded space. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 35–65. Springer, 2023.

Wei Dong, Zijun Chen, Qiyao Luo, Elaine Shi, and Ke Yi. Continual observation of joins under differential privacy. Proceedings of the ACM on Management of Data, 2(3):1–27, 2024.

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

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

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3, pp. 265–284. Springer, 2006.

Kennedy Edemacu and Xintao Wu. Privacy preserving prompt engineering: A survey. arXiv preprint arXiv:2404.06001, 2024.

Marek Eliáš, Michael Kapralov, Janardhan Kulkarni, and Yin Tat Lee. Differentially private release of synthetic graphs. In Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 560–578. SIAM, 2020.

Alessandro Epasto, Vahab Mirrokni, Shyam Narayanan, and Peilin Zhong. k-means clustering with distance-based privacy. Advances in Neural Information Processing Systems, 36, 2024.

Hossein Esfandiari, Vahab Mirrokni, and Shyam Narayanan. Tight and robust private mean estimation with few users. In International Conference on Machine Learning, pp. 16383–16412. PMLR, 2022.

Chenglin Fan and Ping Li. Distances release with differential privacy in tree and grid graph. In 2022 IEEE International Symposium on Information Theory (ISIT), pp. 2190–2195. IEEE, 2022.

Chenglin Fan, Ping Li, and Xiaoyun Li. k-median clustering via metric embedding: towards better initialization with differential privacy. Advances in Neural Information Processing Systems, 36, 2024.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. Advances in Neural Information Processing Systems, 35: 18343–18362, 2022.

Alireza Farhadi, MohammadTaghi Hajiaghayi, and Elaine Shi. Differentially private densest subgraph. In International Conference on Artificial Intelligence and Statistics, pp. 11581–11597. PMLR, 2022.

Yeqi Gao, Zhao Song, Xin Yang, and Yufa Zhou. Differentially private attention computation. In Neurips Safe Generative AI Workshop 2024, 2024.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997, 2023.

Quan Geng, Wei Ding, Ruiqi Guo, and Sanjiv Kumar. Tight analysis of privacy and utility tradeoff in approximate differential privacy. In International Conference on Artificial Intelligence and Statistics, pp. 89–99. PMLR, 2020.

Badih Ghazi, Pritish Kamath, Ravi Kumar, Pasin Manurangsi, and Kewen Wu. On differentially private counting on trees. In 50th International Colloquium on Automata, Languages, and Programming (ICALP 2023), volume 261, pp. 66. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023.

GitHub. Agentgpt, 2024a. https://github.com/reworkd/AgentGPT.

GitHub. Chatgpt system prompt, 2024b. https://github.com/LouisShark/chatgpt_system_prompt.

GitHub. Gpt researcher, 2024c. https://github.com/assafelovic/gpt-researcher.

GitHub. Superagi, 2024d. https://github.com/TransformerOptimus/SuperAGI.

Aditya Golatkar, Alessandro Achille, Yu-Xiang Wang, Aaron Roth, Michael Kearns, and Stefano Soatto. Mixed differential privacy in computer vision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8376–8386, 2022.

Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. Numerical composition of differential privacy. Advances in Neural Information Processing Systems, 34:11631–11642, 2021.

Sivakanth Gopi, Yin Tat Lee, and Daogao Liu. Private convex optimization via exponential mechanism. In Conference on Learning Theory, pp. 1948–1989. PMLR, 2022.

Sivakanth Gopi, Yin Tat Lee, Daogao Liu, Ruoqi Shen, and Kevin Tian. Private convex optimization in general norms. In Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 5068–5089. SIAM, 2023.

Adityanand Guntuboyina and Bodhisattva Sen. L1 covering numbers for uniformly bounded convex functions. In Conference on Learning Theory, pp. 12–1. JMLR Workshop and Conference Proceedings, 2012.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. arXiv preprint arXiv:2208.01626, 2022.

Sirui Hong, Yizhang Lin, Bangbang Liu, Binhao Wu, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Lingyao Zhang, Mingchen Zhuge, et al. Data interpreter: An llm agent for data science. arXiv preprint arXiv:2402.18679, 2024a.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In The Twelfth International Conference on Learning Representations, 2024b. URL https://openreview.net/forum?id=VtmBAGCN7o.

Samuel B Hopkins, Gautam Kamath, Mahbod Majid, and Shyam Narayanan. Robustness implies privacy in statistical estimation. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing, pp. 497–506, 2023.

Jerry Yao-Chieh Hu, Donglin Yang, Dennis Wu, Chenwei Xu, Bo-Yu Chen, and Han Liu. On sparse modern hopfield model. In Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS), 2023.

Jerry Yao-Chieh Hu, Pei-Hsuan Chang, Haozheng Luo, Hong-Yu Chen, Weijian Li, Wei-Po Wang, and Han Liu. Outlier-efficient hopfield layers for large transformer-based models. In Forty-first International Conference on Machine Learning (ICML), 2024a.

Jerry Yao-Chieh Hu, Bo-Yu Chen, Dennis Wu, Feng Ruan, and Han Liu. Nonparametric modern hopfield models. arXiv preprint arXiv:2404.03900, 2024b.

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), 2024c.

Tianhao Huang, Tao Yang, Ivan Habernal, Lijie Hu, and Di Wang. Private language models via truncated laplacian mechanism. arXiv preprint arXiv:2410.08027, 2024.

Zhiyi Huang and Aaron Roth. Exploiting metric structure for efficient private query release. In Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms, pp. 523–534. SIAM, 2014.

Ziyue Huang and Ke Yi. Approximate range counting under differential privacy. In 37th International Symposium on Computational Geometry (SoCG 2021). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.

Haibo Jin, Leyang Hu, Xinuo Li, Peiyan Zhang, Chonghan Chen, Jun Zhuang, and Haohan Wang. Jailbreakzoo: Survey, landscapes, and horizons in jailbreaking large language and vision-language models. arXiv preprint arXiv:2407.01599, 2024.

Christopher Jung, Katrina Ligett, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Moshe Shenfeld. A new analysis of differential privacy's generalization guarantees. arXiv preprint arXiv:1909.03577, 2019.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33: 9459–9474, 2020.

Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. The VLDB journal, 24: 757–781, 2015.

Chenyang Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Tianyi Zhou. Fourier circuits in neural networks: Unlocking the potential of large language models in mathematical reasoning and modular arithmetic. arXiv preprint arXiv:2402.09469, 2024a.

Haoran Li, Yulin Chen, Jinglong Luo, Yan Kang, Xiaojin Zhang, Qi Hu, Chunkit Chan, and Yangqiu Song. Privacy in large language models: Attacks, defenses and future directions. arXiv preprint arXiv:2310.10383, 2023a.

Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. Multi-step jailbreaking privacy attacks on chatgpt. In Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 4138–4153, 2023b.

Ninghui Li, Min Lyu, Dong Su, and Weining Yang. Differential privacy: From theory to practice. Springer, 2017.

Ping Li and Xiaoyun Li. Differential privacy with random projections and sign random projections. arXiv preprint arXiv:2306.01751, 2023a.

Ping Li and Xiaoyun Li. Smooth flipping probability for differential private sign random projection methods. Advances in Neural Information Processing Systems, 36, 2024.

Xiaoyu Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Junwei Yu. Fast john ellipsoid computation with differential privacy optimization. arXiv preprint arXiv:2408.06395, 2024b.

Xiaoyun Li and Ping Li. Differentially private one permutation hashing and bin-wise consistent weighted sampling. arXiv preprint arXiv:2306.07674, 2023b.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In International Conference on Learning Representations, 2021.

Xuechen Li, Daogao Liu, Tatsunori B Hashimoto, Huseyin A Inan, Janardhan Kulkarni, Yin-Tat Lee, and Abhradeep Guha Thakurta. When does differentially private learning not suffer in high dimensions? Advances in Neural Information Processing Systems, 35:28616–28630, 2022.

Yingyu Liang, Zhizhou Sha, Zhenmei Shi, and Zhao Song. Differential privacy mechanisms in neural tangent kernel regression. arXiv preprint arXiv:2407.13621, 2024a.

Yingyu Liang, Zhizhou Sha, Zhenmei Shi, Zhao Song, and Yufa Zhou. Multi-layer transformers gradient can be approximated in almost linear time. arXiv preprint arXiv:2408.13233, 2024b.

Yingyu Liang, Zhenmei Shi, Zhao Song, and Chiwun Yang. Toward infinite-long prefix in transformer. arXiv preprint arXiv:2406.14036, 2024c.

Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. Unraveling the smoothness properties of diffusion models: A gaussian mixture perspective. arXiv preprint arXiv:2405.16418, 2024d.

Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. Tensor attention training: Provably efficient learning of higher-order transformers. arXiv preprint arXiv:2405.16411, 2024e.

Erzhi Liu, Jerry Yao-Chieh Hu, Alex Reneau, Zhao Song, and Han Liu. Differentially private kernel density estimation. arXiv preprint arXiv:2409.01688, 2024a.

Zhihao Liu, Jian Lou, Wenjie Bao, Zhan Qin, and Kui Ren. Differentially private zeroth-order methods for scalable large language model finetuning. arXiv preprint arXiv:2402.07818, 2024b.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, et al. Bolaa: Benchmarking and orchestrating llm-augmented autonomous agents. arXiv preprint arXiv:2308.05960, 2023.

Peihua Mai, Ran Yan, Zhe Huang, Youjia Yang, and Yan Pang. Split-and-denoise: Protect large language model inference with local differential privacy. arXiv preprint arXiv:2310.09130, 2023.

Justus Mattern, Zhijing Jin, Benjamin Weggenmann, Bernhard Schölkopf, and Mrinmaya Sachan. Differentially private language models for secure data sharing. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 4860–4873. Association for Computational Linguistics, 2022.

Shyam Narayanan. Private high-dimensional hypothesis testing. In Conference on Learning Theory, pp. 3979–4027. PMLR, 2022.

Shyam Narayanan. Better and simpler lower bounds for differentially private statistical estimation. arXiv preprint arXiv:2310.06289, 2023.

OpenAI. Creating a gpt, 2024a. `https://help.openai.com/en/articles/8554397-creating-a-gpt`.

OpenAI. Video generation models as world simulators, 2024b. `https://openai.com/research/video-generation-models-as-world-simulators`.

Samet Oymak, Ankit Singh Rawat, Mahdi Soltanolkotabi, and Christos Thrampoulidis. On the role of attention in prompt-tuning. In International Conference on Machine Learning, pp. 26724–26768. PMLR, 2023.

Dylan Patel. Chatgpt system prompt is 1700 tokens?!, 2024. `https://x.com/dylan522p/status/1755086111397863777`.

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

Natalia Ponomareva, Hussein Hazimeh, Alex Kurakin, Zheng Xu, Carson Denison, H Brendan McMahan, Sergei Vassilvitskii, Steve Chien, and Abhradeep Guha Thakurta. How to dp-fy ml: A practical guide to machine learning with differential privacy. Journal of Artificial Intelligence Research, 77:1113–1201, 2023.

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.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10684–10695, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. Advances in neural information processing systems, 35:36479–36494, 2022.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, et al. The prompt report: A systematic survey of prompting techniques. arXiv preprint arXiv:2406.06608, 2024.

Weiyan Shi, Ryan Shea, Si Chen, Chiyuan Zhang, Ruoxi Jia, and Zhou Yu. Just fine-tune twice: Selective differential privacy for large language models. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 6327–6340, 2022a.

Zhenmei Shi, Jiefeng Chen, Kunyang Li, Jayaram Raghuram, Xi Wu, Yingyu Liang, and Somesh Jha. The trade-off between universality and label efficiency of representations from contrastive learning. In The Eleventh International Conference on Learning Representations, 2022b.

Zhenmei Shi, Junyi Wei, Zhuoyan Xu, and Yingyu Liang. Why larger language models do in-context learning differently? arXiv preprint arXiv:2405.19592, 2024.

Tanmay Singh, Harshvardhan Aditya, Vijay K Madisetti, and Arshdeep Bahga. Whispered tuning: Data privacy preservation in fine-tuning llms through differential privacy. Journal of Software Engineering and Applications, 17(1):1–22, 2024.

Zhao Song, Yitan Wang, Zheng Yu, and Lichen Zhang. Sketching for first order method: efficient algorithm for low-bandwidth channel and vulnerability. In International Conference on Machine Learning, pp. 32365–32417. PMLR, 2023a.

Zhao Song, Xin Yang, Yuanyuan Yang, and Lichen Zhang. Sketching meets differential privacy: fast algorithm for dynamic kronecker projection maintenance. In International Conference on Machine Learning (ICML), pp. 32418–32462. PMLR, 2023b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

Roman Vershynin. An introduction with applications in data science. Camb. Ser. Stat. Probab. Math, 47, 2017.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. arXiv preprint arXiv:2305.16291, 2023a.

Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, and Neel Joshi. Is a picture worth a thousand words? delving into spatial reasoning for vision language models. arXiv preprint arXiv:2406.14852, 2024a.

Yilin Wang, Zeyuan Chen, Liangjun Zhong, Zheng Ding, Zhizhou Sha, and Zhuowen Tu. Dolfin: Diffusion layout transformers without autoencoder. arXiv preprint arXiv:2310.16305, 2023b.

Yilin Wang, Haiyang Xu, Xiang Zhang, Zeyuan Chen, Zhizhou Sha, Zirui Wang, and Zhuowen Tu. Omnicontrolnet: Dual-stage integration for conditional image generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7436–7448, 2024b.

Zirui Wang, Zhizhou Sha, Zheng Ding, Yilin Wang, and Zhuowen Tu. Tokencompose: Grounding diffusion with token-level supervision. arXiv preprint arXiv:2312.03626, 2023c.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. arXiv preprint arXiv:2206.07682, 2022.

David Woodruff, Fred Zhang, and Samson Zhou. On robust streaming for learning with experts: algorithms and lower bounds. Advances in Neural Information Processing Systems, 36:79518–79539, 2023.

David P Woodruff. Sketching as a tool for numerical linear algebra. Foundations and Trends® in Theoretical Computer Science, 10(1–2):1–157, 2014.

Dennis Wu, Jerry Yao-Chieh Hu, Teng-Yun Hsiao, and Han Liu. Uniform memory retrieval with larger capacity for modern hopfield models. In Forty-first International Conference on Machine Learning (ICML), 2024a.

Dennis Wu, Jerry Yao-Chieh Hu, Weijian Li, Bo-Yu Chen, and Han Liu. STanhop: Sparse tandem hopfield model for memory-enhanced time series prediction. In The Twelfth International Conference on Learning Representations (ICLR), 2024b.

Chenwei Xu, Yu-Chao Huang, Jerry Yao-Chieh Hu, Weijian Li, Ammar Gilani, Hsi-Sheng Goan, and Han Liu. Bishop: Bi-directional cellular learning for tabular data with generalized sparse modern hopfield model. In Forty-first International Conference on Machine Learning (ICML), 2024a.

Zhuoyan Xu, Zhenmei Shi, Junyi Wei, Fangzhou Mu, Yin Li, and Yingyu Liang. Towards few-shot adaptation of foundation models via multitask finetuning. In The Twelfth International Conference on Learning Representations, 2023.

Zhuoyan Xu, Zhenmei Shi, and Yingyu Liang. Do large language models have compositional ability? an investigation into limitations and scalability. In ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models, 2024b.

Fei Yang, Shiqi Yang, Muhammad Atif Butt, Joost van de Weijer, et al. Dynamic prompt learning: Addressing cross-attention leakage for text-based image editing. Advances in Neural Information Processing Systems, 36, 2024.

Mengmeng Yang, Taolin Guo, Tianqing Zhu, Ivan Tjuawinata, Jun Zhao, and Kwok-Yan Lam. Local differential privacy and its applications: A comprehensive survey. Computer Standards & Interfaces, pp. 103827, 2023.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. Differentially private fine-tuning of language models. In International Conference on Learning Representations, 2021.

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, 2024.

Ying Zhao and Jinjun Chen. A survey on differential privacy for unstructured data content. ACM Computing Surveys (CSUR), 54(10s):1–28, 2022.

Chunyan Zheng, Keke Sun, Wenhao Zhao, Haibo Zhou, Lixing Jiang, Shaoyang Song, and Chunlai Zhou. Locally differentially private in-context learning. In LREC/COLING, 2024.

Appendix

CONTENTS

Roadmap. The appendix is organized as follows. In Section A, we provide more related works. In Section B, we give the preliminary of our paper. In Section C, we offer an outline of our proof techniques. In Section D, we give the analysis of the data structure DPTREE that can solve summation problem with DP and accuracy guarantee. In Section E, we show how to solve weighted distance problem. In Section F, we give our DPTREEDISTANCE data structure that can solve one dimensional ℓ_1 distance problem with DP and accuracy guarantee. In Section G, we present the analysis of our DPTREEHIGHDIM (Algorithm 7) data structure, which can address the high-dimensional ℓ_1 distance problem while ensuring differential privacy and accuracy guarantees. In Section H, we show how we can handle adaptive query. In Section I, we show how to extend our algorithm to Softmax activation and give the analysis of DPTREESOFTMAX (Algorithm 3) and DPTREESOFTMAXADAPTIVE (Algorithm 8).

A More Related Work

Differential Privacy Guarantee Analysis. Ever since Dwork et al. (2006) proposes the notion of differential privacy (DP), it has become one of the most essential standards of privacy protection in both theoretical and empirical ways (Dwork, 2008; Li et al., 2017; Zhao & Chen, 2022; Ponomareva et al., 2023; Yang et al., 2023). DP provides a powerful, robust, and quantifiable privacy definition, allowing algorithm design with concrete privacy and accuracy guarantee (Hay et al., 2009; Esfandiari et al., 2022; Andoni et al., 2023; Li & Li, 2023b; Huang & Yi, 2021; Ghazi et al., 2023; Backurs et al., 2024; Cohen-Addad et al., 2022a; Epasto et al., 2024; Chen et al., 2022; Hopkins et al., 2023; Narayanan, 2022; 2023; Jung et al., 2019; Li & Li, 2024; Fan & Li, 2022; Fan et al., 2024; Li & Li, 2023a; Cherapanamjeri et al., 2023; Cohen-Addad et al., 2022b; Dong et al., 2024; Farhadi et al., 2022; Gopi et al., 2021; 2023; Li et al., 2022; Gopi et al., 2022; Eliáš et al., 2020; Song et al., 2023b; Dinur et al., 2023; Woodruff et al., 2023; Song et al., 2023a; Gao et al., 2024; Liang et al., 2024a; Li et al., 2024b). Additionally, new mechanisms have been proposed beyond the traditional Laplace, Gaussian, and Exponential mechanisms (Dwork & Roth, 2014). For example, truncated Laplace mechanism (Geng et al., 2020) is proved to be the current tightest the lower and upper bounds on the minimum noise amplitude and power cross all (ϵ, δ)-DP distributions.

Cross-Attention in System Prompt, RAG, Stable Diffusion and More. Cross-attention (Vaswani et al., 2017), first introduced in language translation, is a widely used technique in many advanced AI systems. For example, Stable Diffusion (Rombach et al., 2022; Liang et al., 2024d; Wang et al., 2023b;c; 2024b) and SORA (OpenAI, 2024b) employ cross-attention as a core module for a text-to-image conditional generation. This technique is also utilized by other multimodal models (Liang et al., 2024e), including Imagen (Saharia et al., 2022) and Diffusion Transformer (Peebles & Xie, 2023). In the realm of text-to-image editing, Hertz et al. (2022) analyzes and controls the cross-attention module to enable editing without requiring additional training. Furthermore, Yang et al. (2024) tackles the issue of inaccurate cross-attention maps, enhancing fine-grained control over edited regions while preventing unintended changes to other areas. In addition, Retrieval Augmented Generation (RAG) (Lewis et al., 2020; Borgeaud et al., 2022; Gao et al., 2023), a technique that improves model responses by retrieving information from a knowledge base or external documents, extensively uses cross-attention as its core design module. Cross-attention also has other applications. Oymak et al. (2023) demonstrates that the prompt-tuning (Liang et al., 2024c) task can be formulated as cross-attention, while Chen et al. (2021) uses cross-attention to fuse multi-scale features in vision transformers, thereby reducing computation. Moreover, attention-based Transformer architecture makes LGMs equipping many emergent ability (Wei et al., 2022), such as spatial reasoning (Wang et al., 2024a), mathematical reasoning (Li et al., 2024a), in-context learning ability (Shi et al., 2024), compositional ability (Xu et al., 2024b), few-shot adaptation ability (Shi et al., 2022b; Xu et al., 2023), and so on. There are some other works that used cross attention in Hopfield Models (Hu et al., 2023; Wu et al., 2024b; Hu et al., 2024c; Xu et al., 2024a; Wu et al., 2024a; Hu et al., 2024a;b).

B More Preliminary

In Section B.1, we give the probability tools we use in the paper. In Section B.2, we provide the algebraic facts we use. In Section B.3, we give the DP facts we use in the paper. In Section B.4, we compare between popular DP mechanisms.

B.1 Probability Tools

In this section, we give several probability lemmas.

Lemma B.1 (Markov's inequality)**.** *If x is a nonnegative random variable and $t > 0$, we have*

$$\Pr[x \geq t] \leq \frac{\mathbb{E}[x]}{t}.$$

Lemma B.2 (Chernoff bound, (Chernoff, 1952))**.** *Let x_i be a Bernoulli random variable with probability p_i of being equal to 1 and $1 - p_i$ of being equal to 0, and all x_i for $i \in [n]$ are independent. Let $x = \sum_{i=1}^n x_i$. Let $\mu = \mathbb{E}[x] = \sum_{i=1}^n p_i$. Then, for all $\delta > 0$ we have*

$$\Pr[x \geq (1+\delta)\mu] \leq \exp(-\delta^2 \mu/3),$$

and for all $0 < \delta < 1$

$$\Pr[x \leq (1-\delta)\mu] \leq \exp(-\delta^2 \mu/2).$$

Lemma B.3 (Chebyshev's inequality)**.** *Let x (integrable) be a random variable with finite non-zero variance σ^2 (and thus finite expected value μ). Then for any real number $k > 0$,*

$$\Pr[|x - \mu| \geq k\sigma] \leq \frac{1}{k^2}.$$

B.2 Algebraic Facts

Fact B.4 (Upper bound of exponential, Fact C.9 in Liang et al. (2024d))**.** *For $a \in \mathbb{R}$, $b \in \mathbb{R}$, $a, b \leq R$, where $R \geq 0$, we have*

$$|\exp(a) - \exp(b)| \leq \exp(R)|a - b|.$$

B.3 DP Facts

In this section, we present several facts about differential privacy (DP).

We first define vector neighboring dataset and sensitivity.

Definition B.5 (Vector neighboring dataset)**.** *We define the two neighboring datasets as $X, X' \in \mathbb{R}^n$ such that $\|X - X'\|_1 \leq 1$, i.e., they differ on a single data point.*

Definition B.6 (Vector sensitivity)**.** *The sensitivity of a function $f : \mathbb{R}^n \to \mathbb{R}^d$ is defined by: $\Delta := \max_{X, X' \in \mathbb{R}^n, \|X - X'\|_1 = 1} \|f(X) - f(X')\|_1$.*

We state the post-processing property, which means, in an algorithm, if one step is DP, all the following steps are DP.

Fact B.7 (Post-processing, see Fact 2.1 in Ghazi et al. (2023))**.** *Let $\mathcal{A}_1$ be an (ϵ, δ)-DP algorithm and $\mathcal{A}_2$ be a (randomized) post-processing algorithm. Then the algorithm $\mathcal{A}(X) = \mathcal{A}_2(\mathcal{A}_1(X))$ is still an (ϵ, δ)-DP algorithm.*

If we have many DP algorithms, we need a composition rule. The most straightforward composition is the basic/sequential composition rule.

Fact B.8 (Basic composition, see Fact 2.3 in Ghazi et al. (2023))**.** *Let $\mathcal{A}_1$ be an (ϵ_1, δ_1)-DP algorithm and $\mathcal{A}_2$ be an (ϵ_2, δ_2)-DP algorithm. Then $\mathcal{A}(X) = (\mathcal{A}_1(X), \mathcal{A}_2(\mathcal{A}_1(X), X))$ is an $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP algorithm.*

We can do much better if we know that the inputs are disjoint.

Fact B.9 (Parallel composition, see Fact 2.4 in Ghazi et al. (2023))**.** *Let $\mathcal{A}_1$ be an (ϵ_1, δ_1)-DP algorithm and $\mathcal{A}_2$ be an (ϵ_2, δ_2)-DP algorithm. Assume $\mathcal{A}_1$ and $\mathcal{A}_2$ depend on disjoint subsets of input coordinates. Then the algorithm $\mathcal{A}(X) = (\mathcal{A}_1(X), \mathcal{A}_2(\mathcal{A}_1(X), X))$ is a $(\max\{\epsilon_1, \epsilon_2\}, \max\{\delta_1, \delta_2\})$-DP algorithm.*

In addition, we have the advanced composition, which improves the dependence of the number of DP algorithms to square root but compromises the term δ'.

Theorem B.10 (Advanced composition, see Theorem 3.20 in Dwork & Roth (2014))**.** *For all $\epsilon, \delta, \delta' \geq 0$, the class of (ϵ, δ)-differentially private mechanisms satisfies $(\epsilon', k\delta + \delta')$-differential privacy under k-fold adaptive composition for:*

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

B.4 Comparison of Truncated Laplace, Gaussian, and Laplace Mechanisms

We first define the Laplace mechanism as below:

Definition B.11 (Laplace distribution)**.** *We use $\mathrm{Lap}(b)$ to denote the pdf: $p(z) = \frac{1}{2b}\exp(-\frac{|z|}{b})$.*

Fact B.12. *For $z \sim \mathrm{Lap}(b)$, $\mathbb{E}[z] = 0$, and $\mathrm{Var}[z] = 2b^2$. Furthermore, if $b = \Delta/\epsilon$, we have $\mathrm{Var}[z] = 2\Delta^2/\epsilon^2$.*

In this paper, we use the Chebyshev inequality to bound the error, and from Geng et al. (2020), we know that the truncated Laplace mechanism has the current minimum variance across all (ϵ, δ)-DP distributions.

The variance of Gaussian mechanism in Theorem 3.22 in Dwork & Roth (2014):

$$\mathrm{Var} = \frac{2\Delta^2\log(1.25/\delta)}{\epsilon^2}.$$

The variance of Laplace mechanism in Fact B.12:

$$\mathrm{Var} = \frac{2\Delta^2}{\epsilon^2}.$$

The variance of truncated Laplace mechanism in Fact 2.5, for $c \in (0, 1]$:

$$\mathrm{Var} = \frac{2\Delta^2 c}{\epsilon^2}.$$

Thus, since it has the minimum variance, we choose the truncated Laplace mechanism to design our algorithms among these popular mechanisms.

C Proof Outline

This section provides the proof outline of our paper. In Section C.1, we analyze our DPTree data structure. In Section C.2, we show the sensitivity of summation problem. In Section C.3, we explain the high-level idea behind the weighted ℓ_p^p distance query. In Section C.4, we show how to answer one-dimensional weighted ℓ_1 distance query. In Section C.5, we show how to answer Softmax distance query using previous algorithms. In Section C.6, we show how to handle adaptive query. By combining the results from these sections, we prove the main results in Section 4.

C.1 Summation Segment Tree

First, in order to solve the weighted distance problem, we need to have a basic DP algorithm (Algorithm 2) that can answer simple summation queries. After analyzing its DP and error in Section D, we state the data structure theorem.

Theorem C.1 (DPTree data structure, informal version of Theorem D.1)**.** *There is a data structure (see* DPTree *in Algorithm 2) that uses $O(n)$ spaces to support the following operations:*

- $\textsc{Init}(a \in \mathbb{R}^n, n \in \mathbb{N}_+, \Delta \in \mathbb{N}_+, \epsilon \in (0,1), \delta \in (0,1))$. *It takes $O(n)$ time to initialize the data structure.*
- $\textsc{Query}(x \in [n], y \in [n])$. *It takes $O(\log n)$ time to output a number z such that*
  - *the process of output z satisfies (ϵ, δ)-DP private, which computes $\sum_{i=x}^{y} a_i$,*
  - $|z - \sum_{i=x}^{y} a_i| \leq O(\epsilon^{-1} \Delta \log^{3/2} n)$,
  - *it holds with probability* 0.99.

During the design of the data structure, we found an interesting property based on the parallel composition rule of DP Fact B.9. We will now state the lemma, whose proof is provided in Section D.

Lemma C.2 (Weighted sum of disjoint interval queries, informal version of Lemma D.8)**.** *If the following conditions hold that:*

- *Let there be t disjoint intervals, i.e., S_j for $j \in [t]$, such that $S_j \cap S_k = \emptyset$ for all $j \neq k$.*
- *Let $\epsilon \in (0,1)$ and $\delta \in (0,1)$.*
- *Let a_j for $j \in [t]$ be a series that square converges to a, i.e., $\sum_{j=1}^{t} a_j^2 \leq a$.*

Then, we have Alg. 2 is (ϵ, δ)-DP and output $\sum_{j=1}^{t} a_j \textsc{Query}(S_j)$ with the error upper bounded by

$$O(a^{1/2} \epsilon^{-1} \Delta \log^{3/2} n)$$

with probability 0.99.

From Lemma C.2, we can see that if we have multiple disjoint interval queries, the error of the weighted sum of the intervals can be bounded independently of the number of queries, as long as the sum of squared weights is finite.

C.2 Sensitivity for Range Summation Problem

Our DP summation tree data structure DPTree (Algorithm 2) requires sensitivity parameter Δ. In this section, we show that for the summation problem, we have the sensitivity $\Delta = 2R$ if the input $X \in [-R, R]^n$.

Lemma C.3 (Sensitivity of summation)**.** *Let $X \in [-R, R]^n$. We have the sensitivity $\Delta = 2R$ for* DPTree.Init *in Algorithm 2.*

Proof. Let's say two neighboring datasets X and X' differ in x_i and x'_i for some i in the array X. Then for a summation problem, i.e. $f(X) := \sum_{i=1}^{n} x_i$, we have

$$\Delta = \max_{X, X'} |f(X) - f(X')| = \max_{X, X'} |x_i - x'_i| = 2R.$$

where the first step follows from Definition B.6, the second step follows from X, X' differ in x_i, x'_i, and the last step follows from each coordinate of the dataset is bounded in $[-R, R]$. $\square$

C.3 Weighted ℓ_p^p Distance Problem

In this section, we introduce the intuition behind the method for handling the weighted ℓ_p^p distance problem. The formal lemmas and proofs can be found in Section E.

Given a dataset and a query point in d dimensions, we round each coordinate of the data points and the query point to the nearest multiple of a small interval. We then aggregate the weights of data points that have been rounded to the same position. Finally, we compute the sum of these aggregated weights multiplied by the distances between the query point and the data points over the rounded positions. This approach makes the computation more efficient while maintaining sufficient accuracy.

We provide an example of weighted ℓ_1-distance of a one-dimensional dataset consisting of 10 data points, i.e., $X \in [0, 1]^{10}$ and a query $y = 0$ in Figure 1.

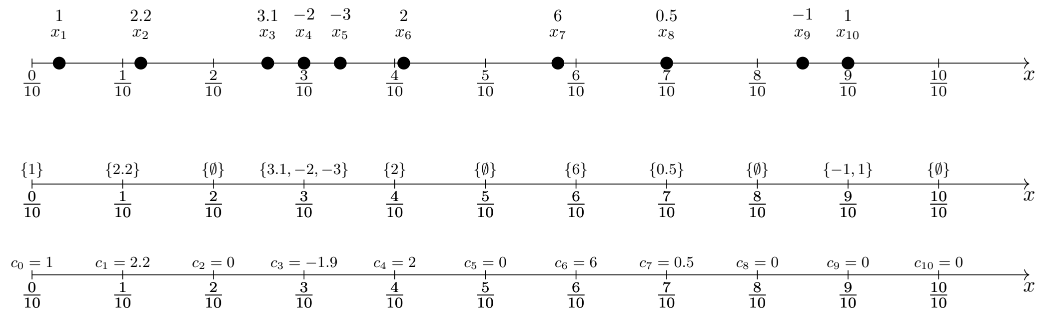

Figure 1: The visualization of how to build c_j of rounded dataset $X \in [0,1]^{10}$ and compute the weighted ℓ_1 distance. The number above each x_i is w_i. See Algorithm 5 for details. Suppose $y = 0$. Then $\sum_{i=1}^n w_i|y - x_i| = 0.1 \cdot 2.2 + 0.3 \cdot 3.1 + 0.3 \cdot (-2) + 0.3 \cdot (-3) + 0.4 \cdot 2 + 0.6 \cdot 6 + 0.7 \cdot 0.5 + 0.9 \cdot (-1) + 0.9 \cdot 1 = 4.4$. And $\sum_{j=0}^n |k - j| c_j / n = 0.1 \cdot 2.2 + 0.3 \cdot (-1.9) + 0.4 \cdot 2 + 0.6 \cdot 6 + 0.7 \cdot 0.5 = 4.4$. See details in Lemma E.1.

Lemma C.4 (Weighted ℓ_p^p-distance high dimension, informal version of Lemma E.2)**.** *If the following conditions hold:*

- *Let data $X \in [0,R]^{n \times d}$ and $x_i^\top \in [0,R]^d$ be the i-th row of x, weight $w \in \mathbb{R}^n$, query $y \in [0,R]^d$.*
- *We round each dimension of X and y to an integer multiple of R/n.*
- *Let $x_{i,k}, y_k$ denote the k-th coordinates of x_i, y for $k \in [d]$.*
- *Let $c_{j,k} := \sum_{j_0 \in S_{j,k}} w_{j_0}$ where the set $S_{j,k}$ is the set of index i such that the corresponding $x_{i,k}$ is rounded to jR/n for $j \in \{0,1,2,\dots,n\}$ for $k \in [d]$.*
- *After rounding, we assume that y_k is in the $l_k R/n$ position for $l_k \in \{0,1,2,\dots,n\}$ for $k \in [d]$.*

For the weighted problem, we have

$$\sum_{i=1}^n w_i \cdot \|y - x_i\|_p^p = \sum_{k=1}^d \sum_{j=0}^n (|l_k - j| R/n + O(R/n))^p c_{j,k}.$$

where $O(R/n)$ is the rounding error for each data point.

Remark C.5. *In Lemma C.4, we first round the dataset. This rounding simplifies the calculation by reducing the number of possible positions to consider, from real values in $[0,R]^d$ to the total $O(nd)$ spots. However, it also introduces an error $O(R/n)$ for one data point. Then, for one spot in the rounded dataset, we sum over the weights of that spot and multiply the corresponding distance raised to the power of p. Additionally, since we are dealing with ℓ_p^p distance, the rounding error is also raised to the power of p.*

C.4 One-Dimensional Weighted ℓ_1 Distance Data Structure

Based on previous discussions in Section C.1 and C.3, we can now describe our one-dimensional weighted ℓ_1 distance data structure, DPTreeDistance, presented in Algorithm 5 and 6, which generalizes the results from Backurs et al. (2024). Drawing from the intuition in Section C.3, the initialization process is as follows: first, we round each data point in the dataset to the nearest multiple of a small interval and build an array that aggregates the corresponding weights. This array is then fed into our DPTree data structure in Algorithm 2. At query time, we query the DPTree to obtain the aggregated weights within a small interval and multiply these weights by the distance to the query point. Furthermore, we also introduce a relative error parameter α to reduce the total number of queries to $O(\log(n)/\alpha)$ instead of querying all n positions. We also analyze the DP and the error bound; see details in Section F.

Theorem C.6 (DPTREEDISTANCE data structure, informal version of Theorem F.6)**.** *There is a data structure* DPTREEDISTANCE *(Algorithm 5 and 6) that uses $O(n)$ spaces to solve weighted ℓ_1-distance query problem for dataset $X \subset [0,R]$ and support the following operations:*

- INIT$(X \subset [0,R], n \in \mathbb{N}_+, w \in [-R_w, R_w]^n, \epsilon \in (0,1), \delta \in (0,1))$. *(Algorithm 5) It takes $O(n)$ time to initialize the data structure.*
- DISTANCEQUERY$(y \in [0,R], \alpha \in (0,1))$. *(Algorithm 6) It takes $O(\alpha^{-1}\log^2 n)$ time to output a number z such that*
  - *the process of output z satisfies (ϵ,δ)-DP, which computes $\sum_{i\in[n]} w_i|y-x_i|$,*
  - $|z - \sum_{i\in[n]} w_i|y-x_i|| \le \alpha \sum_{i\in[n]} w_i|y-x_i| + O(\epsilon^{-1}\alpha^{-1/2} R R_w \log^{3/2} n)$*,*
  - *it holds with probability* 0.99.

C.5 SOFTMAX ACTIVATION

We then describe how we extend the previous results to Softmax activation, i.e. exponential inner product function (Definition 4.1). From Alman & Song (2023), we know that Softmax activation can be approximated by polynomial kernel function $P(\cdot)$ with a certain error. The following lemma shows that we can transform weighted Softmax queries into polynomial kernels. More specifically, we have one term that computes the weighted ℓ_2^2 distance, which is the place where we add DP noises. Because of the decomposability of the ℓ_p^p distance, i.e. $\sum_{i\in[n]} w_i\|x_i - y\|_p^p = \sum_{j\in[d]}\sum_{i\in[n]} w_i|x_{i,j} - y_j|^p$, we can easily extend the results of Section C.4 to handle the ℓ_2^2 distance query. After that, we compute the term for the weighted ℓ_2^2 norms of approximation kernel exactly. Summing all these terms, with a certain error, we can answer the Softmax query. Related proofs can be found in Section I.

Lemma C.7 (Weighted Softmax approximation, informal version of Lemma I.6)**.** *Let the accuracy parameter be $\epsilon_s \in (0, 0.1)$. Let $R \ge 1$. Let $r \le \binom{2s+2d}{2s}$ and $s = O(\max\{\frac{\log(1/\epsilon_s)}{\log(\log(1/\epsilon_s)/R)}, R^2\})$. Let $\Gamma_{R,s} := \max_{j\in[s]} \frac{R^j}{\sqrt{j!}}$ (Definition I.3). Let $P(x) : [0,R]^d \to [0,\Gamma_{R,s}]^r$ be the s-th order polynomial kernel function defined in Lemma I.5. Then, we can approximate the exponential inner product using the polynomial kernel function:*

$$w^\top \exp(Xy/d) = -\frac{1}{2}\sum_{j\in[r]}\sum_{i\in[n]} w_i|P(x_i)_j - P(y)_j|^2 + \frac{1}{2}\sum_{i\in[n]} w_i(\|P(x_i)\|_2^2 + \|P(y)\|_2^2) + O(w^\top \exp(Xy/d)\cdot\epsilon_s).$$

Moreover, the vectors $P(\cdot)$ can be computed in $O(r)$ time.

C.6 ADAPTIVE QUERY

We introduce how we can modify our algorithm to solve the adaptive query problem using some tools in Qin et al. (2022). Our approach is based on proving that our algorithm can handle any query within the query space with a certain error. Since adaptive queries must lie within this space, our algorithm can effectively handle them. In Section C.5, we demonstrate our algorithm's capability to answer weighted Softmax distance queries with constant probability. We then use the Chernoff bound to boost the constant probability of our algorithm to a high probability. Next, we apply the notion of an ϵ_0-net to bound all query points within the net using the union bound. Finally, we bound all points in the query space by utilizing the Lipschitz property of the weighted Softmax distance function and introducing an additive error. See the proofs in Sections H and I.

Lemma C.8 (Adaptive Softmax, informal version of Lemma I.10)**.** *If the following conditions hold:*

- *Let N be the ℓ_∞ ϵ_0-net of $\mathcal{B}$, and let $|N|$ be the size of the net N.*
- *Let data set $X \in [0,R]^{n\times d}$, weights $w \in [-R_w, R_w]^n$, query $y \in [0,R]^d$.*
- *Let the relative error parameter $\alpha \in (0,1)$, the failure probability $p_f \in (0, 0.01)$.*

- *We create* $l = O(\log((R/\epsilon_0)^r/p_f))$ *independent copies of the data structure* $\{\text{DPTREESOFTMAX}_j\}_{j=1}^l$ *(Algorithm 3) and take the median of the outputs with each data structure instantiated with* $(\epsilon/l, (\delta+\delta')/l)$*-DP.*
- *Let* $f(y) := \mathrm{Median}(\{\text{DPTREESOFTMAX}_j.\text{DISTANCEQUERY}(y, \alpha)\}_{j=1}^l)$.
- *Let* $Z(y) := w^\top \exp(Xy/d)$.
- *Let* $B = O(\epsilon^{-1}\alpha^{-1/2} l \Gamma_{R,s}^2 R_w r \sqrt{\log(l/\delta')} \cdot \log^{3/2} n)$.

Then with probability $1 - p_f$*, for all query points* $q \in \mathcal{B}$*, there exists a point* $y \in N$ *which is the closest to* q*, we can have the process of outputting median of* l *responses is* $(\epsilon, \delta + \delta')$*-DP and the error satisfies*

$$|f(y) - Z(q)| \le (\alpha + \epsilon_s) Z(q) + B + 2n\sqrt{d} R R_w \exp(R^2)\epsilon_0.$$

D DPTREE ALGORITHM

In this section, we give the analysis of privacy, accuracy and runtime of our DPTREE (Algorithm 2). In Section D.1, we give the theorem (Theorem D.1) of our data structure that can answer summation problem. In Section D.2, we improve our data structure from constant probability to high probability by applying Chernoff bound. In Section D.3, we give the analysis. In Section D.4, we show some results of our data structure if the input queries are disjoint.

D.1 SINGLE DATA STRUCTURE

We give the theorem of our DPTREE data structure that can answer the summation problem with DP, accuracy, runtime guarantee.

Theorem D.1 (DPTREE data structure, formal version of Theorem C.1)**.** *There is a data structure (see* DPTREE *in Algorithm 2) that uses* $O(n)$ *spaces to support the following operations:*

- INIT$(a \in \mathbb{R}^n, n \in \mathbb{N}_+, \Delta \in \mathbb{N}_+, \epsilon \in (0,1), \delta \in (0,1))$. *It takes* $O(n)$ *time to initialize the data structure.*
- QUERY$(x \in [n], y \in [n])$. *It takes* $O(\log n)$ *time to output a number* z *such that*
  - *the process of output* z *satisfies* (ϵ, δ)*-DP private, which computes* $\sum_{i=x}^y a_i$*,*
  - $|z - \sum_{i=x}^y a_i| \le O(\epsilon^{-1}\Delta \log^{3/2} n)$*,*
  - *it holds with probability* 0.99*.*

Proof. The proofs follow from combining Lemma D.3 (running time of initialization), Lemma D.4 (running time of query), Lemma D.5 (DP of query), and Lemma D.6 (error of query) together. □

D.2 BOOST THE CONSTANT PROBABILITY TO HIGH PROBABILITY

We can use Chernoff bound to boost the high probability by repeating the data structure multiple times.

Theorem D.2 (High-probability)**.** *There is a data structure (see* DPTREEHIGHPROB *in Algorithm 4) that uses* $O(n\log(1/\delta_{\rm fail}))$ *spaces to support the following operations*

- INIT$(a \in \mathbb{R}^n, n \in \mathbb{N}_+, \Delta \in \mathbb{N}_+, \epsilon \in (0,1), \delta \in (0,1), \delta_{\rm fail} \in (0, 0.01))$. *It takes* $O(n\log(1/\delta_{\rm fail}))$ *time to initialize the data structure.*
- QUERY$(x \in [n], y \in [n])$. *It takes* $O(\log(n) \cdot \log(1/\delta_{\rm fail}))$ *time to output a number* z *such that*
  - *the process of output* z *satisfies* (ϵ, δ)*-DP private, which computes* $\sum_{i=x}^y a_i$*,*
  - $|z - \sum_{i=x}^y a_i| \le O(\epsilon^{-1}\Delta \log^{3/2}(n) \cdot \log(1/\delta_{\rm fail}))$*,*

– *it holds with probability* $1 - \delta_{\mathrm{fail}}$ *for failure probability* $\delta_{\mathrm{fail}} \in (0, 0.01)$.

Proof. Note that our data structure (Theorem D.1) succeeds with probability 0.99. The success of the algorithm (Theorem D.1) can be viewed as a Bernoulli random variable, to which we apply the Chernoff bound (Lemma B.2). By repeating the data structure $O(\log(1/\delta_{\mathrm{fail}}))$ times and taking the median of the outputs, we boost the success probability. The details are following.

To boost the success probability, we assume the query is repeated l times. Let $i \in [l]$, and let z_i denote the indicator random variable for the success of the i-th instance of the data structure for a single query. Let $z = \sum_{i=1}^{l} z_i$ be the total success times. Since $p = \Pr[z_i = 1] = 0.99$, we can have $\mu = \mathbb{E}[z] = \sum_{i=1}^{l} p = lp$. Note that $p = 0.99$. By setting $\delta = 0.1$ and using Chernoff bound from Lemma B.2, we can show

$$\Pr[z \le l/2] \le \Pr[z \le (1-\delta)lp] \le \exp(-\delta^2 lp/2).$$

Note that we want $z > l/2$ (since we want at least half to succeed so we could take the median),

$$\Pr[z > l/2] \ge 1 - \exp(-\delta^2 lp/2).$$

To ensure that failure probability is δ_{fail}, we have

$$\exp(-\delta^2 lp/2) = \delta_{\mathrm{fail}}.$$

We can make this hold by choosing $l = O(\log(1/\delta_{\mathrm{fail}}))$.

By the DP basic composition rule (Fact B.8), we need to choose $\epsilon = \epsilon'/O(\log(1/\delta_{\mathrm{fail}}))$ and $\delta = \delta'/O(\log(1/\delta_{\mathrm{fail}}))$ where ϵ', δ' are the ϵ, δ in Theorem D.1. □

Algorithm 4 Boost constant probability

```
1:  datastructure DPTreeHighProb                                         ▷ Theorem D.2
2:  members
3:      D_1, ..., D_{O(log(1/δ_fail))} : DPTree                          ▷ Alg. 2
4:  end members
5:  procedure Init(a ∈ R^n, n ∈ N_+, Δ ∈ N_+, ε ∈ (0,1), δ ∈ (0,1), δ_fail ∈ (0,0.01))
6:      for i = 1 → O(log(1/δ_fail)) do
7:          D_i.Init(a, n, Δ, ε/O(log(1/δ_fail)), δ/O(log(1/δ_fail)))
8:      end for
9:  end procedure
10: procedure Query(x ∈ [n], y ∈ [n])
11:     r ← 0^{O(log(1/δ_fail))}
12:     for i = 1 → O(log(1/δ_fail)) do
13:         r_i ← D_i.Query(x, y)
14:     end for
15:     return Median of r
16: end procedure
17: end datastructure
```

D.3 Algorithm of Data Structure

In this section, we analyze the accuracy, DP, and runtime of Algorithm 2.

We first analyze the runtime.

Lemma D.3 (Runtime of initialization, Algorithm 2). *For the initialization, we have the time complexity of Algorithm 2 is $O(n)$.*

Proof. All the computations are dominated by $O(n)$ time. □

Lemma D.4 (Runtime of query, Algorithm 2). *For each query, we have the time complexity of Algorithm 2 is $O(\log n)$.*

Proof. Due to the property of tree, we will use at most $2 \log n$ nodes in the tree, thus the running time is $O(\log n)$. $\square$

We now analyze the DP.

Lemma D.5 (Privacy of query, Algorithm 2)**.** *The output process of* QUERY *(see Algorithm 2) is* (ϵ, δ)*-DP.*

Proof. Suppose that our dataset is $X \in [-R, R]^n$. Note that we only add noise in the pre-processing stage. There is no noise in the query stage. Since the problem we care about is summation, if we change one leaf node, the sensitivity $\Delta = 2R$ (see Lemma C.3). Since we add noise to each node in the tree, and each leaf node count will contribute to $\log n$ nodes, it is equivalent to our output function being in $\log n$ dimension. We will then blow up the DP parameter by $\log n$ factor. Thus, using the basic composition rule (Fact B.8), the DP guarantee for the whole tree data structure is $((\epsilon/\log n) \cdot \log n, (\delta/\log n) \cdot \log n)$ which is (ϵ, δ)-DP. $\square$

We now analyze the accuracy.

Lemma D.6 (Accuracy of query, Algorithm 2)**.** *Let* $\epsilon \in (0, 1)$ *and* $\delta \in (0, 1)$*. Then, using Chebyshev's inequality and Fact 2.5, we have the error of* QUERY*(see Algorithm 2) output is upper bounded by:*

$$O(\epsilon^{-1} \Delta \log^{3/2} n).$$

with probability 0.99.

Proof. For an interval S, we define $\textsc{TrueQuery}(S)$ to be the output of DPTREE.TRUEQUERY in Algorithm 2. Let $\textsc{Query}(S)$ denote the noised interval query answer returned by DPTREE.QUERY in Algorithm 2. Let $z := \textsc{Query}(S) - \textsc{TrueQuery}(S)$, which from Algorithm 2 we can see this is the sum of $O(\log n)$ independent truncated Laplace random variables each with parameter $\mathrm{TLap}(\Delta, \epsilon/\log n, \delta/\log n)$. Thus,

$$z = \sum_{i=1}^{O(\log n)} z_i$$

where $z_i \sim \mathrm{TLap}(\Delta, \epsilon/\log n, \delta/\log n)$, and every z_i are independent to each other.

We know $\mu = \mathbb{E}[z] = 0$ since $\mathbb{E}[z_i] = 0$. From Fact 2.5, we know the variance for each z_i is $\mathrm{Var}[z_i] = c\epsilon^{-2}\Delta^2 \log^2 n$ where $0 < c \leq 2$ and $c = 2$ when $\delta = 0$.

Therefore, we can show

$$\begin{aligned} \mathrm{Var}[z] =&~ \mathrm{Var}[\sum_{i=1}^{O(\log n)} z_i] \\ =&~ \sum_{i=1}^{O(\log n)} \mathrm{Var}[z_i] \\ =&~ O(c\epsilon^{-2}\Delta^2 \log^3 n) \end{aligned} \tag{4}$$

where the first step follows from definition of z, the second step follows from every z_i are independent to each other, and the last step follows from $\mathrm{Var}[z_i] = O(c\epsilon^{-2}\Delta^2 \log^2 n)$.

Note that we wish to bound $|z| = |\textsc{Query}(S) - \textsc{TrueQuery}(S)|$ as our error.

Using Lemma B.3, we can have

$$\Pr[|z| \geq k\sigma] \leq \frac{1}{k^2}.$$

We know that $\sigma = \sqrt{\mathrm{Var}[z]} = O(c^{1/2}\epsilon^{-1}\Delta \log^{3/2} n)$. Picking $k = 10$, we have

$$\Pr[|z| < 10\sigma] \geq 0.99.$$

Thus, we conclude that error is bounded by $O(c^{1/2}\epsilon^{-1}\Delta \log^{3/2} n) = O(\epsilon^{-1}\Delta \log^{3/2} n)$ (since $c \in (0, 2]$) with probability 0.99. $\square$

D.4 DISJOINT INTERVALS

In this section, we show some interesting results for our DPTREE data structure if the input queries are disjoint.

Lemma D.7 (Sum of disjoint interval queries)**.** *If the following conditions hold that:*

- *Let there be t disjoint intervals, i.e., S_j for $j \in [t]$, such that $S_j \cap S_k = \emptyset$ for all $j \neq k$.*
- *Let $\epsilon \in (0,1)$ and $\delta \in (0,1)$.*

Then, we have Algorithm 2 is (ϵ, δ)-DP and outputs $\sum_{j=1}^{t} \mathrm{QUERY}(S_j)$ with the error upper bounded by

$$O(t^{1/2}\epsilon^{-1}\Delta \log^{3/2} n)$$

with probability 0.99.

Proof. From Lemma D.5, we know that DPTree.QUERY is (ϵ, δ)-DP. Then, from Fact B.9 and the disjoint intervals in Algorithm 6, we can conclude that the value returned is (ϵ, δ)-DP.

Let $\mathrm{TRUEQUERY}(S_j)$ denote the true interval query answer returned by DPTREE.TRUEQUERY in Algorithm 2 for interval S_j. Let $\mathrm{QUERY}(S_j)$ denote the noised interval query answer returned by DPTREE.QUERY in Algorithm 2 for interval S_j. Let $z_j := \mathrm{QUERY}(S_j) - \mathrm{TRUEQUERY}(S_j)$ and $z = \sum_{j=1}^{t} z_j$. From the proof of Lemma D.6, we know z_j is the sum of $O(\log n)$ independent truncated Laplace random variables each with parameter $\mathrm{TLap}(\Delta, \epsilon/\log n, \delta/\log n)$ and the variance is bounded by

$$\mathrm{Var}[z_j] = O(\epsilon^{-2}\Delta^2 \log^3 n)$$

Since the intervals S_j are disjoint, they are independent to each other. Then, we have

$$\begin{aligned}
\mathrm{Var}[z] = & ~ \mathrm{Var}[\sum_{j=1}^{t} z_j] \\
= & ~ \sum_{j=1}^{t} \mathrm{Var}[z_j] \\
= & ~ O(t\epsilon^{-2}\Delta^2 \log^3 n)
\end{aligned}$$

where the first step follows from definition of z, the second step follows from the intervals are disjoint, and the last step follows from $\mathrm{Var}[z_j] = O(\epsilon^{-2}\Delta^2 \log^2 n)$.

Note that we wish to bound $|z|$ as our error.

Using Lemma B.3, we can have error bounded by

$$O(t^{1/2}\epsilon^{-1}\Delta \log^{3/2} n)$$

with probability 0.99. □

Moreover, this can be generalized to weighted sum of queries.

Lemma D.8 (Weighted sum of disjoint interval queries, formal version of Lemma C.2)**.** *If the following conditions hold that:*

- *Let there be t disjoint intervals, i.e., S_j for $j \in [t]$, such that $S_j \cap S_k = \emptyset$ for all $j \neq k$.*
- *Let $\epsilon \in (0,1)$ and $\delta \in (0,1)$.*
- *Let a_j for $j \in [t]$ be a series that square converges to a, i.e., $\sum_{j=1}^{t} a_j^2 \leq a$.*

Then, we have Alg. 2 is (ϵ, δ)-DP and output $\sum_{j=1}^{t} a_j \textsc{Query}(S_j)$ with the error upper bounded by

$$O(a^{1/2}\epsilon^{-1}\Delta\log^{3/2} n)$$

with probability 0.99.

Proof. The DP proof is the same as in the proof of Lemma D.7.

Let $\textsc{TrueQuery}(S_j)$ and $\textsc{Query}(S_j)$ be same in the proof of Lemma D.7 Let $z_j := \textsc{Query}(S_j) - \textsc{TrueQuery}(S_j)$ and $z = \sum_{j=1}^{t} a_j z_j$. From the proof of Lemma D.7, we know the variance of z_j is bounded by

$$\mathrm{Var}[z_j] = O(\epsilon^{-2}\Delta^2\log^3 n)$$

Since the intervals S_j are disjoint, they are independent to each other. Then, we have

$$\begin{aligned}
\mathrm{Var}[z] = & \ \mathrm{Var}[\sum_{j=1}^{t} a_j z_j] \\
= & \sum_{j=1}^{t} \mathrm{Var}[a_j z_j] \\
= & \sum_{j=1}^{t} a_j^2 \,\mathrm{Var}[z_j] \\
= & \sum_{j=1}^{t} a_j^2 \cdot O(\epsilon^{-2}\Delta^2\log^3 n) \\
= & \ O(a\epsilon^{-2}\Delta^2\log^3 n)
\end{aligned}$$

where the first step follows from the definition of z, the second step follows from the intervals are disjoint, the third step follows from the $\mathrm{Var}[az] = a^2\,\mathrm{Var}[z]$ for a random variable z and a constant a, the fourth step follows from the $\mathrm{Var}[z_j] = O(\epsilon^{-2}\Delta^2\log^2 n)$, and the last step follows from $\sum_{j=1}^{t} a_j^2 \le a$.

Note that we wish to bound $|z|$ as our error.

Using Lemma B.3, we can have error bounded by

$$O(a^{1/2}\epsilon^{-1}\Delta\log^{3/2} n)$$

with probability 0.99. □

E WEIGHTED ℓ_p^p DISTANCE

In this section, we introduce how to handle weighted ℓ_p^p distance problem in the high level idea. In Section E.1, we show how to solve one dimensional weighted problem. In Section E.2, we show how to solve high dimensional weighted problem by decomposing each coordinate of the high dimensional dataset.

Suppose we have the original data $X \in [0, R]^n$ and weight $w \in \mathbb{R}^n$ and query $y \in [0, R]$. We want to compute the weighted ℓ_1-distance, i.e.

$$\sum_{i=1}^{n} w_i \cdot |y - x_i|.$$

For data in d-dimension, due to the decomposability of ℓ_p^p distance, our problem will be: given $x_i \in [0, R]^d$ and $w_i \in \mathbb{R}$ for $i \in [n]$, and $y \in [0, R]^d$, we can compute

$$\sum_{i=1}^{n} w_i \cdot \|y - x_i\|_p^p = \sum_{j=1}^{d}\sum_{i=1}^{n} w_i \cdot |y_j - x_{i,j}|^p$$

where $x_{i,j}, y_j$ means the j-th coordinates of x_i, y for $j \in [d]$.

Therefore, we can solve one dimension problem first, and then the high dimension case can be solved automatically.

E.1 One Dimensional Weighted Distance

Now we can give the lemma for weighted distance of dataset.

Lemma E.1 (Weighted distance one dimension)**.** *If the following conditions hold:*

- *Let data $X \in [0, R]^n$, weight $w \in \mathbb{R}^n$, query $y \in [0, R]$.*
- *We round X and y to an integer multiple of R/n.*
- *Let $c_j = \sum_{j_0 \in S_j} w_{j_0}$ where set S_j is the set of index i such that the corresponding x_i is rounded to jR/n for $j \in \{0, 1, 2, \dots, n\}$.*
- *After rounding, we assume y is in the kR/n position for $k \in \{0, 1, 2, \dots, n\}$.*

For the weighted problem, we have

$$\sum_{i=1}^n w_i \cdot |y - x_i| = \sum_{j=0}^n (|k-j|R/n + O(R/n))c_j.$$

Moreover, we have

$$\sum_{i=1}^n w_i \cdot |y - x_i|^p = \sum_{j=0}^n (|k-j|R/n + O(R/n))^p c_j$$

where $O(R/n)$ is the rounding error for each data point.

Proof. For each i, we have:

$$w_i \cdot |y - x_i| = w_i \cdot (\frac{|k-j|R}{n} + O(\frac{R}{n})).$$

where $O(R/n)$ is the rounding error introduced by each data point, since each data point will be at most $O(R/n)$ away from its true position.

We can construct c_j by

$$c_j = \sum_{j_0 \in S_j} w_{j_0}$$

where set S_j is the set of index i such that the corresponding x_i is rounded to jR/n. Moreover, c_j can be negative.

Summing over all i and grouping by j, we get:

$$\sum_{i=1}^n w_i \cdot |y - x_i| = \sum_{j=0}^n (\frac{|k-j|R}{n} + O(\frac{R}{n}))c_j.$$

The total rounding error will be $O(R)$ because we have n data points, each with an error of at most $O(R/n)$.

Moreover, we have

$$\sum_{i=1}^n w_i \cdot |y - x_i|^p = \sum_{j=0}^n (\frac{|k-j|R}{n} + O(\frac{R}{n}))^p c_j.$$

$\square$

E.2 HIGH DIMENSIONAL WEIGHTED DISTANCE

Finally, we can solve the problem of weighted distance for d-dimensional dataset.

Lemma E.2 (Weighted ℓ_p^p-distance high dimension, formal version of Lemma C.4)**.** *If the following conditions hold:*

- *Let data $X \in [0,R]^{n\times d}$ and $x_i^\top \in [0,R]^d$ be the i-th row of x, weight $w \in \mathbb{R}^n$, query $y \in [0,R]^d$.*
- *We round each dimension of X and y to an integer multiple of R/n.*
- *Let $x_{i,k}, y_k$ denote the k-th coordinates of x_i, y for $k \in [d]$.*
- *Let $c_{j,k} := \sum_{j_0 \in S_{j,k}} w_{j_0}$ where set $S_{j,k}$ is the set of index i such that the corresponding $x_{i,k}$ is rounded to jR/n for $j \in \{0,1,2,\dots,n\}$ for $k \in [d]$.*
- *After rounding, we assume y_k is in the $l_k R/n$ position for $l_k \in \{0,1,2,\dots,n\}$ for $k \in [d]$.*

For the weighted problem, we have

$$\sum_{i=1}^n w_i \cdot \|y - x_i\|_p^p = \sum_{k=1}^d \sum_{j=0}^n (|l_k - j|R/n + O(R/n))^p c_{j,k}$$

where $O(R/n)$ is the rounding error for each data point.

Proof. We can show

$$\begin{aligned}\sum_{i=1}^n w_i \cdot \|y - x_i\|_p^p &= \sum_{k=1}^d \sum_{i=1}^n w_i \cdot |y_k - x_{i,k}|^p \\ &= \sum_{k=1}^d \sum_{j=0}^n (|l_k - j|R/n + O(R/n))^p c_{j,k}\end{aligned}$$

where the first step follows from decomposability of ℓ_p^p-distance by dimension, the second step follows from Lemma E.1.

□

F ONE-DIMENSIONAL WEIGHTED ℓ_1 DISTANCE QUERY

In this section, we generalize the algorithms in Backurs et al. (2024) to weighted distance. Here, we compute the problem of one-dimensional weighted ℓ_1 distance query i.e. $\sum_{i\in[n]} w_i|y - x_i|$ for a given query $y \in [0,R]$, weights $w \in [-R_w, R_w]^n$ and dataset $X \subset [0,R]$ and $n = |X|$. In Section F.1, we analyze the runtime of our algorithm. In Section F.2, we analyze the DP and accuracy of our algorithm. In Section F.3, we give the theorem for our DPTREEDISTANCE data structure.

F.1 RUNTIME ANALYSIS

We first analyze the runtime.

Lemma F.1 (Runtime of initialization, Algorithm 5)**.** *For the initialization, we have the time complexity of* INIT *in Algorithm 5 is $O(n)$.*

Proof. In the initialization of INIT, the computations need $O(n)$ time to compute the count and $O(\log n)$ time to build the tree. Thus, total time is $O(n)$. □

Lemma F.2 (Runtime of DISTANCEQUERY, Algorithm 6)**.** *For the ℓ_1 distance query, we have the time complexity of* DISTANCEQUERY *in Algorithm 6 is $O(\alpha^{-1}\log^2 n)$.*

Algorithm 5 Pre-processing data structure

datastructure DPTreeDistance ▷ Theorem F.6
members
  $\mathcal{D}$: DPTree ▷ Alg. 2
  $X : [0, R]^n$
  $w : [-R_w, R_w]^n$
end members
procedure Init($X \subset [0, R]$, $n \in \mathbb{N}_+$, $w \in [-R_w, R_w]^n$, $\epsilon \in (0, 1)$, $\delta \in (0, 1)$) ▷ Lemma E.1
  $X, w, a \leftarrow X, w, 0^{n+1}$
  for $i = 1 \to n$ **do**
    $j \leftarrow$ Round(x_i, n) ▷ $x_i \in X$ for $i \in [n]$
    $a_j \leftarrow a_j + w_i$
  end for
  $\mathcal{D}$.Init$(a, n + 1, 2R_w, \epsilon, \delta)$ ▷ Alg. 2, Lemma C.3
end procedure
procedure Round($x \in [0, R]$, $n \in \mathbb{N}_+$)
  Let $j \in \{0, 1, 2, \dots n - 1\}$ denote the integer such that $jR/n \le x < (j + 1)R/n$
  if $|x - (j + 1)R/n| \le |x - jR/n|$ **then**
    $j \leftarrow j + 1$
  end if
  return j
end procedure
end datastructure

Algorithm 6 One dimensional weighted ℓ_1 distance query

1: **datastructure** DPTreeDistance ▷ Theorem F.6
2: **procedure** DistanceQuery($y \in [0, R]$, $\alpha \in (0, 1)$) ▷ Lemma F.2, Lemma F.4, Lemma F.5
3:   $y \leftarrow$ Round$(y, n) \cdot (R/n)$ ▷ Alg. 5
4:   Value $\leftarrow 0$
5:   **for** $j = 0, 1, ..., O(\log(n)/\alpha)$ **do**
6:     $l_j \leftarrow$ Round$(y + \frac{R}{(1+\alpha)^{j+1}}, n)$
7:     $r_j \leftarrow$ Round$(y + \frac{R}{(1+\alpha)^{j}}, n)$ ▷ Consider the points to the right of y
8:     Value $\leftarrow$ Value + $\mathcal{D}$.Query$(l_j, r_j) \cdot \frac{R}{(1+\alpha)^j}$ ▷ Alg. 2
9:   **end for**
10:   **for** $j = 0, 1, ..., O(\log(n)/\alpha)$ **do**
11:     $l_j \leftarrow$ Round$(y - \frac{R}{(1+\alpha)^{j}}, n)$
12:     $r_j \leftarrow$ Round$(y - \frac{R}{(1+\alpha)^{j+1}}, n)$ ▷ Consider the points to the left of y
13:     Value $\leftarrow$ Value + $\mathcal{D}$.Query$(l_j, r_j) \cdot \frac{R}{(1+\alpha)^j}$ ▷ Alg. 2
14:   **end for**
15:   **Return** Value
16: **end procedure**
17: **end datastructure**

Proof. In DistanceQuery, the computations need $O(\log n)$ time to compute one value from DPTree.Query and this process need to be repeated $O(\alpha^{-1} \log n)$ times. □

Remark F.3. *In Line 8 and 13 of Algorithm 6, we use $R/(1 + \alpha)^j$ to approximate the distance of each data point to the query in Lemma E.1, i.e. $|k - j|R/n$. This will introduce α relative error but also reduce the numbers of iteration from $O(n)$ to $O(\log(n)/\alpha)$.*

F.2 Privacy and Accuracy Analysis

We show the DP.

Lemma F.4 (Privacy of DistanceQuery, Algorithm 6)**.** *The output process of* DistanceQuery *(Algorithm 6) is (ϵ, δ)-DP.*

Proof. From Lemma D.5, we know that the process of $\mathrm{DPTree}.\textsc{Query}$ is (ϵ, δ)-DP. We observe that intervals in Algorithm 6 are disjoint. Then, following the same logic in the proof of Lemma D.8, we can conclude that the output process to return the value is (ϵ, δ)-DP. $\square$

We now analyze the accuracy of the algorithm.

Lemma F.5 (Accuracy of DISTANCEQUERY, Algorithm 6)**.** *If the following conditions are satisfied:*

- *Let $X \in [0, R]^n$ be a dataset consisting of n one-dimensional numbers, with weights $w \in [-R_w, R_w]^n$.*
- *Let $\alpha \in (0, 1)$ represent the relative error parameter utilized in Algorithm 6.*
- *Let $\widetilde{A}$ denote the output of the* DISTANCEQUERY *in Algorithm 6.*
- *Let $A_* := \sum_{i\in[n]} w_i|y - x_i|$ represent the true distance query value for a specific query y.*

Then with probability 0.99*, we have*

$$|\widetilde{A} - A_*| \le \alpha A_* + O(\epsilon^{-1}\alpha^{-1/2} R R_w \log^{3/2} n).$$

Proof. To simplify the explanation, we consider only the distance query for the points in X located to the right of y. The proof can be symmetrically applied to the case of points to the left of y. For an interval $S_j := (l_j, r_j)$ where l_j, r_j are defined in Algorithm 6, we define $\textsc{TrueQuery}(S_j)$ to be the output of DPTREE.TRUEQUERY in Algorithm 2. Let

$$\widehat{A} := \sum_{j=0}^{O(\log(n)/\alpha)} \frac{R}{(1+\alpha)^j} \cdot \textsc{TrueQuery}(S_j).$$

Since TRUEQUERY returns the sum of the corresponding weights, it aligns with the true answer $A_* := \sum_{i\in[n]} w_i|y - x_i|$. Thus, we have

$$|\widehat{A} - A_*| \le \alpha \cdot A_*,$$

because for all j, the distances between y and different points in X vary only by a multiplicative factor of $(1 + \alpha)$.

Next we show the additive error. Let $\textsc{Query}(S_j)$ denote the noised interval query answer returned by DPTREE.QUERY in Algorithm 2. Algorithm 6 outputs $\widetilde{A} = \sum_{j=0}^{O(\log(n)/\alpha)} \frac{R}{(1+\alpha)^j} \cdot \textsc{Query}(S_j)$. We wish to bound

$$|\widehat{A} - \widetilde{A}| \le |\sum_{j=0}^{O(\log(n)/\alpha)} \frac{R}{(1+\alpha)^j} \cdot (\textsc{TrueQuery}(S_j) - \textsc{Query}(S_j))|.$$

Let $z_j := \textsc{Query}(S_j) - \textsc{TrueQuery}(S_j)$, which from Algorithm 2 we can see this is the sum of $O(\log n)$ independent truncated Laplace random variables.

From Lemma D.8, we only need to show that the series $\frac{1}{(1+\alpha)^j}$ for $j \in \{0, 1, \ldots, O(\log(n)/\alpha)\}$ square converges to $1/\alpha$, since R is a constant.

We can show

$$\begin{aligned}
\sum_{j=0}^{O(\log(n)/\alpha)} \frac{1}{(1+\alpha)^{2j}} &\le \sum_{j=0}^{\infty} \frac{1}{(1+\alpha)^{2j}} \\
&\le \sum_{j=0}^{\infty} \frac{1}{(1+\alpha)^{j}} \\
&= \frac{1}{1 - \frac{1}{1+\alpha}}
\end{aligned}$$

$$= 1 + \frac{1}{\alpha}$$
$$= O(1/\alpha)$$

where the first step follows from we extend the finite sum to infinite sum, the second step follows from $\frac{1}{(1+\alpha)^{2j}} \leq \frac{1}{(1+\alpha)^j}$, the third step follows from the closed form of geometric sum, the fourth step follows from simple algebra, and the last step follows from $\alpha \in (0, 1)$.

Then from the proof of Lemma D.8, we can know that the variance is given by

$$O(\frac{R^2 R_w^2 \log^3 n}{\alpha \epsilon^2}) \tag{5}$$

since the sensitivity $\Delta = 2R_w$ from Lemma C.3.

Using Lemma B.3, we can have additive error bounded by

$$O(\frac{R \cdot R_w \log^{3/2} n}{\epsilon \sqrt{\alpha}}).$$

with probability 0.99. □

F.3 One Dimension Single Data Structure

We therefore have the data structure that can solve weighted ℓ_1-distance problem.

Theorem F.6 (DPTreeDistance data structure, formal version of Theorem C.6)**.** *There is a data structure* DPTreeDistance *(Algorithm 5,6) that uses $O(n)$ spaces to solve weighted ℓ_1-distance query problem for dataset $X \subset [0, R]$ and support the following operations:*

- Init$(X \subset [0, R], n \in \mathbb{N}_+, w \in [-R_w, R_w]^n, \epsilon \in (0, 1), \delta \in (0, 1))$. *(Algorithm 5) It takes $O(n)$ time to initialize the data structure.*
- DistanceQuery$(y \in [0, R], \alpha \in (0, 1))$. *(Algorithm 6) It takes $O(\alpha^{-1} \log^2 n)$ time to output a number z such that*
  - *the process of output z satisfies (ϵ, δ)-DP private, which computes $\sum_{i \in [n]} w_i |y - x_i|$,*
  - $|z - \sum_{i \in [n]} w_i |y - x_i|| \leq \alpha \sum_{i \in [n]} w_i |y - x_i| + O(\epsilon^{-1} \alpha^{-1/2} R R_w \log^{3/2} n)$,
  - *it holds with probability* 0.99.

Proof. The proofs follow from combining Lemma F.1 (running time of initialization), Lemma F.2 (running time of query), Lemma F.4 (DP of query), and Lemma F.5 (error of query) together. □

G High-Dimensional Weighted ℓ_1 Query

In this section, we show how we can solve the high dimensional weighted ℓ_1 distance problem, generalizing results from Backurs et al. (2024). In Section G.1, we give the analysis of Algorithm 7. In Section G.2, we give the theorem of our DPTreeHighDim data structure.

Algorithm 5,6 can be naturally extended to higher dimensions because of the decomposability of the ℓ_1 distance function. We construct d separate one-dimensional distance query data structures, each corresponding to a coordinate projection of the dataset.

G.1 Privacy and Accuracy Analysis for High Dimensional Weighted Distance

We now give the analysis of our Algorithm 7 for high dimensional weighted ℓ_1-distance query.

Algorithm 7 High-dimensional weighted ℓ_1 distance query

1: **datastrucutre** DPTreeHighDim ▷ Theorem G.3
2: **members**
3: $\quad \mathcal{D}_1, \dots, \mathcal{D}_d$: DPTreeDistance ▷ Alg. 5
4: $\quad X : [0, R]^{n \times d}$
5: $\quad w : [-R_w, R_w]^n$
6: **end members**
7: **procedure** Init($X \subset [0, R]^d$, $n \in \mathbb{N}_+$, $w \in [-R_w, R_w]^n$, $\epsilon \in (0, 1)$, $\delta \in (0, 1)$, $\delta' \in (0, 1)$, $c \in (0, 0.1)$) ▷ Lemma E.2
8: $\quad X \leftarrow X$
9: $\quad w \leftarrow w$
10: $\quad$ **for** $i = 1 \to d$ **do**
11: $\quad\quad \mathcal{D}_i$.Init($X_{:,i}$, n, w, $c\epsilon/\sqrt{d \log(1/\delta')}$, δ/d) ▷ Alg. 5
12: $\quad$ **end for**
13: **end procedure**
14: **procedure** DistanceQuery($y \in [0, R]^d$, $\alpha \in (0, 1)$) ▷ Lemma G.1, Lemma G.2
15: $\quad$ Value $\leftarrow 0$
16: $\quad$ **for** $i = 1 \to d$ **do**
17: $\quad\quad$ Value $\leftarrow$ Value + $\mathcal{D}_i$.DistanceQuery(y_i, α) ▷ Alg. 6
18: $\quad$ **end for**
19: $\quad$ **return** Value
20: **end procedure**
21: **end datastrucutre**

Lemma G.1 (Privacy of DistanceQuery, Algorithm 7)**.** *If the following conditions hold*

- *Let data set $X \in [0, R]^{n \times d}$, weights $w \in [-R_w, R_w]^n$, query $y \in [0, R]^d$.*
- *Let $\epsilon \in (0, 1)$, $\delta \in (0, 1)$, $\delta' \in (0, 1)$.*
- *Let $c \in (0, 0.1)$ be a small constant and A be the output of* DistanceQuery *in Algorithm 7, where each one-dimensional algorithm is configured to be $(c\epsilon/\sqrt{d \log(1/\delta')}, \delta/d)$-DP (see Line 11).*
- *Let $A_* = \sum_{i \in [n]} w_i \|y - x_i\|_1$ represent the true distance query value.*
- *Let $\epsilon = O(\log(1/\delta'))$.*

Then, we have the output process of DistanceQuery *(Algorithm 7) is $(\epsilon, \delta + \delta')$-DP.*

Proof. The $(\epsilon, \delta + \delta')$-DP guarantee follows from the approximate DP advanced composition result Theorem B.10. Our algorithm instantiate each one-dimensional data structure with $(c\epsilon/\sqrt{d \log(1/\delta')}, \delta/d)$-DP total d times.

From advanced composition in Theorem B.10, for a sufficient small parameter ϵ and constant c, we have the final privacy loss parameter be:

$$O(c\epsilon\sqrt{2d \log(1/\delta')}/\sqrt{d \log(1/\delta')}) = O(\epsilon)$$

and the final failure probability parameter be:

$$d\delta/d + \delta' = \delta + \delta'.$$

□

Lemma G.2 (Accuracy of DistanceQuery, Algorithm 7)**.** *If the following conditions hold*

- *Let data set $X \in [0, R]^{n \times d}$, weights $w \in [-R_w, R_w]^n$, query $y \in [0, R]^d$.*
- *Let $\epsilon \in (0, 1)$, $\delta \in (0, 1)$, $\delta' \in (0, 1)$.*

- *Let* $c \in (0, 0.1)$ *be a small constant and* A *be the output of* DISTANCEQUERY *in Algorithm 7, where each one-dimensional algorithm is configured to be* $(c\epsilon/\sqrt{d \log(1/\delta')}, \delta/d)$*-DP (see Line 11).*
- *Let* $A_* = \sum_{i \in [n]} w_i \|y - x_i\|_1$ *represent the true distance query value.*

With probability 0.99*, we have*

$$|A - A_*| \le \alpha A_* + O(\epsilon^{-1} \alpha^{-1/2} R R_w d \sqrt{\log(1/\delta')} \cdot \log^{3/2} n).$$

Proof. Let A_i be the i-th dimension output returned by $\mathcal{D}_i$ in Algorithm 7. Let $A_{*,i}$ be the true distance query value in the i-th dimension. Observe that $A_* = \sum_{i=1}^d A_{*,i}$ and $A = \sum_{i=1}^d A_i$.

We follow the similar idea in the proof of Lemma F.5. Let $z_{j,i}$ be the random variables that represent z_j (used in the proof of Lemma F.5) for the i-th coordinate. We can observe that the overall error across d coordinates can be upper bounded by

$$|\sum_{i=1}^d \sum_{j=0}^{O(\log(n)/\alpha)} \frac{R z_{j,i}}{(1+\alpha)^j}|$$

where each $z_{j,i}$ is the sum of $O(\log n)$ truncated Laplace random variables independent to others. With ϵ scaled down by $c\epsilon/\sqrt{d \log(1/\delta')}$ and δ scaled down by δ/d, the variance of each individual dimension is given by (see Eq. (5))

$$O(\alpha^{-1} \epsilon^{-2} d R^2 R_w^2 \log(1/\delta') \log^3 n).$$

Thus, the total variance for d instantiated data structures is then

$$O(\alpha^{-1} \epsilon^{-2} d^2 R^2 R_w^2 \log(1/\delta') \log^3 n).$$

Finally, from Lemma B.3, we have the additive error given by

$$O(\alpha^{-1/2} \epsilon^{-1} d R R_w \sqrt{\log(1/\delta')} \cdot \log^{3/2} n).$$

□

G.2 HIGH DIMENSION SINGLE DATA STRUCTURE

We have the data structure that can solve weighted ℓ_1-distance problem in d-dimensional data.

Theorem G.3 (DPTREEHIGHDIM data structure)**.** *There is a data structure* DPTREEHIGHDIM *(Algorithm 7) that uses* $O(nd)$ *spaces to solve weighted* ℓ_1*-distance query problem for dataset* $X \subset [0, R]^d$ *and support the following operations:*

- INIT$(X \subset [0, R]^d, n \in \mathbb{N}_+, w \in [-R_w, R_w]^n, \epsilon \in (0, 1), \delta \in (0, 1), \delta' \in (0, 1), c \in (0, 0.1))$. *(Algorithm 7) It takes* $O(nd)$ *time to initialize the data structure.*
- DISTANCEQUERY$(y \in [0, R]^d, \alpha \in (0, 1))$. *(Algorithm 7) It takes* $O(\alpha^{-1} d \log^2 n)$ *time to output a number* z *such that*
  - *the process of output* z *satisfies is* $(\epsilon, \delta + \delta')$*-DP private, which computes* $\sum_{i \in [n]} w_i \|y - x_i\|_1$*,*
  - $|z - \sum_{i \in [n]} w_i \|y - x_i\|_1| \le \alpha \sum_{i \in [n]} w_i \|y - x_i\|_1 + O(\epsilon^{-1} \alpha^{-1/2} R R_w d \sqrt{\log(1/\delta')} \cdot \log^{3/2} n)$*,*
  - *it holds with probability* 0.99*.*

Proof. For the runtime analysis, since we loop data structure DPTREEDISTANCE d times, an additional d factor will appear for both initialization and query time complexity. The DP is proved by Lemma G.1. The accuracy is proved by Lemma G.2. □

H Adaptive Query

In this section, we introduce how we can solve the adaptive query problem by our algorithm, using some tools from Qin et al. (2022). Our idea is that, if we can prove that our algorithm can solve any query in the query space with certain error. Then, since adaptive query must lie in this space, we can handle adaptive query. In Section H.1, we show how we can boost the constant probability of our algorithm to high probability. In Section H.2, we show how we can apply the notion of ϵ_0-net and bound all query points in net. In Section H.3, we show how we can bound all points in the query space by introducing an additive error. In Section H.4, we examine the effects of different norms on our adaptive query proof.

First, from Theorem G.3, given query $y \in [0, R]^d, \alpha \in (0, 1)$ we have $\textsc{DistanceQuery}(y, \alpha)$ that can solve d-dimension weighted ℓ_1-distance problem with constant probability 0.99. Now we show how to improve it to solve adaptive query problem.

H.1 Boost the Constant Probability to High Probability

We can repeat the data structure multiple times and take the median to boost the constant probability using Chernoff bound from Lemma B.2.

Lemma H.1 (Using Chernoff bound to boost the probability)**.** *If the following conditions hold:*

- *Let data set $X \in [0, R]^{n \times d}$, weights $w \in [-R_w, R_w]^n$, query $y \in [0, R]^d$.*
- *Let relative error parameter $\alpha \in (0, 1)$, the failure probability $p_f \in (0, 0.01)$.*
- *We create $l = O(\log(1/p_f))$ independent copies of data structure* $\textsc{DPTreeHighDim}$ *and take the median of the outputs with each data structure instantiated with $(\epsilon/l, (\delta + \delta')/l)$-DP.*
- *Let $A_* = \sum_{i \in [n]} w_i \|y - x_i\|_1$ be the true answer.*
- *Let $B = O(\epsilon^{-1}\alpha^{-1/2} l R R_w d \sqrt{\log(l/\delta')} \cdot \log^{3/2} n)$.*

Then for each fixed query point y, we can have the process of outputting the median of l responses is $(\epsilon, \delta + \delta')$-DP and the error is upper bounded by $\alpha A_* + B$ with probability $1 - p_f$.

Proof. By basic composition Fact B.8, we prove the DP. Similar to the proof of Theorem D.2, we prove the error by Chernoff bound (Lemma B.2). $\square$

H.2 From Each Fixed Query Point to All On-net Points

In this section, we build ϵ_0-net and generalize from each fixed query point to all on-net points.

Definition H.2 (ℓ_p ϵ_0-net, see Definition 4.2.1 in Vershynin (2017))**.** *We define N be ℓ_p ϵ_0-net of $\mathcal{B} := \{q \in [0, R]^d\}$ such that, for every point q in $\mathcal{B}$, there exists $y \in N$ satisfying $\|y - q\|_p \le \epsilon_0$.*

Fact H.3 (ℓ_∞ ϵ_0-net)**.** *Let N be the ℓ_∞ ϵ_0-net of $\mathcal{B}$, and $|N|$ be the size of net N. We have $|N| \le (5R/\epsilon_0)^d$.*

Fact H.4 (ℓ_2 ϵ_0-net, see Lemma 5 in Woodruff (2014))**.** *Let N be the ℓ_2 ϵ_0-net of $\mathcal{B}$, and $|N|$ be the size of net N. We have $|N| \le (5R/\epsilon_0)^d$.*

Fact H.5 (ℓ_1 ϵ_0-net, see Theorem 2 in Guntuboyina & Sen (2012))**.** *Let N be the ℓ_1 ϵ_0-net of $\mathcal{B}$, and $|N|$ be the size of net N. We have $|N| \le (5R\sqrt{d}/\epsilon_0)^d$.*

Lemma H.6 (From for each query point to for all points in net)**.** *If the following conditions hold:*

- *Let N be the ℓ_∞ ϵ_0-net of $\mathcal{B}$, and $|N|$ be the size of net N.*
- *Let data set $X \in [0, R]^{n \times d}$, weights $w \in [-R_w, R_w]^n$, query $y \in [0, R]^d$.*
- *Let relative error parameter $\alpha \in (0, 1)$, the failure probability $p_f \in (0, 0.01)$.*

- *We create $l = O(\log(|N|/p_f))$ independent copies of data structure* DPTreeHighDim *and take the median of the outputs with each data structure instantiated with $(\epsilon/l, (\delta + \delta')/l)$-DP.*
- *Let $A_* = \sum_{i \in [n]} w_i \|y - x_i\|_1$ be the true answer.*
- *Let $B = O(\epsilon^{-1} \alpha^{-1/2} l R R_w d \sqrt{\log(l/\delta')} \cdot \log^{3/2} n)$.*

Then with probability $1 - p_f$, for all query points $y \in N$, we can have the process of outputting the median of l responses is $(\epsilon, \delta + \delta')$-DP and the error is upper bounded by $\alpha A_* + B$.

Proof. By basic composition Fact B.8, we prove the DP. From Lemma H.1, we know for each $y \in N$, the error is upper bounded by $\alpha A_* + B$ with probability $1 - p_f/|N|$.

Then, by union bound, with probability $1 - p_f$, the error of all $|N|$ query points in the net $y \in N$ is upper bounded by $\alpha A_* + B$. $\square$

H.3 From Net Points to All Points

In this section, we show how to generalize points from net to all points in the query space. Since adaptive query must lie in this space, we complete the proof of adaptive query.

Lemma H.7 (Lipschitz of query function)**.** *If the following conditions hold:*

- *Let data set $X \in [0, R]^{n \times d}$, weights $w \in [-R_w, R_w]^n$, query $y \in [0, R]^d$.*
- *Let $Z(y) := \sum_{i \in [n]} w_i \|y - x_i\|_1$.*
- *Let $L = nR_w$.*

Then, we have $Z(y)$ is L-Lipschitz (note that we have ℓ_1 Lipschitz here).

Proof. We can show

$$\begin{aligned} |Z(y) - Z(\widetilde{y})| &= |\sum_{i \in [n]} w_i \|y - x_i\|_1 - \sum_{i \in [n]} w_i \|\widetilde{y} - x_i\|_1| \\ &\leq \sum_{i \in [n]} |w_i| \cdot |\|y - x_i\|_1 - \|\widetilde{y} - x_i\|_1| \\ &\leq \sum_{i \in [n]} |w_i| \cdot \|y - \widetilde{y}\|_1 \\ &= nR_w \cdot \|y - \widetilde{y}\|_1 \end{aligned}$$

where the first step follows from definition of $Z(y)$, the second step follows from triangular inequality, the third step follows from reverse triangular inequality, the fourth step follows from $w \in [-R_w, R_w]^n$. $\square$

Lemma H.8 (From points in net to all points in query space)**.** *If the following conditions hold:*

- *Let N be the ℓ_∞ ϵ_0-net of $\mathcal{B}$, and $|N|$ be the size of net N.*
- *Let data set $X \in [0, R]^{n \times d}$, weights $w \in [-R_w, R_w]^n$, query $y \in [0, R]^d$.*
- *Let relative error parameter $\alpha \in (0, 1)$, the failure probability $p_f \in (0, 0.01)$.*
- *We create $l = O(\log((R/\epsilon_0)^d/p_f))$ independent copies of data structure $\{$DPTreeHighDim$_j\}_{j=1}^l$ and take the median of the outputs with each data structure instantiated with $(\epsilon/l, (\delta + \delta')/l)$-DP.*
- *Let $f(y) := \mathrm{Median}(\{\text{DPTreeHighDim}_j.\text{DistanceQuery}(y, \alpha)\}_{j=1}^l)$.*
- *Let $Z(y) := \sum_{i \in [n]} w_i \|y - x_i\|_1$, where $Z(y)$ is L-Lipschitz with $L = nR_w$.*

- *Let* $B = O(\epsilon^{-1}\alpha^{-1/2} l R R_w d \sqrt{\log(l/\delta')} \cdot \log^{3/2} n)$.

Then with probability $1 - p_f$*, for all query points* $q \in \mathcal{B}$*, there exists a point* $y \in N$ *which is the closest to* q*, we can have the process of outputting the median of* l *responses is* $(\epsilon, \delta + \delta')$*-DP and the error satisfy*

$$|f(y) - Z(q)| \le \alpha Z(q) + B + 2Ld\epsilon_0.$$

Proof. By basic composition Fact B.8, we prove the DP.

We define an event E such that:

$$\begin{aligned} &\forall y \in N \\ &|f(y) - Z(y)| \le \alpha Z(y) + B. \end{aligned}$$

From Lemma H.1, with $l = O(\log(|N|/p_f))$ we know

$$\Pr[\text{event } E \text{ holds}] \ge 1 - p_f$$

We can show

$$\begin{aligned} l &= O(\log(|N|/p_f) \\ &= O(\log((R/\epsilon_0)^d/p_f) \end{aligned}$$

where the first step follows from definition of l, the second step follows from Fact H.3.

We condition on event E to be held. Then, by definition of ℓ_∞ ϵ_0-net (see Definition H.2), for each $q \notin N$, there exists $y \in N$ such that

$$\|y - q\|_\infty \le \epsilon_0 \tag{6}$$

We know

$$\begin{aligned} |Z(y) - Z(q)| &\le L \cdot \|y - q\|_1 \\ &\le L \cdot d\|y - q\|_\infty \\ &\le L \cdot d\epsilon_0 \end{aligned} \tag{7}$$

where the first step follows from Lemma H.7, the second step follows from $\|x\|_1 \le d\|x\|_\infty$ for $x \in \mathbb{R}^d$, and the last step follows from Eq. (6).

Using the on-net query y to answer the off-net query q, for any $q \notin N$, we have

$$\begin{aligned} |f(y) - Z(q)| &\le |f(y) - Z(y)| + |Z(q) - Z(y)| \\ &\le |f(y) - Z(y)| + L \cdot d \cdot \epsilon_0 \\ &\le \alpha Z(y) + B + L \cdot d \cdot \epsilon_0 \\ &\le \alpha Z(q) + B + 2L \cdot d \cdot \epsilon_0 \end{aligned} \tag{8}$$

where the first step follows from triangular inequality, the second step follows from Eq. (7), the third step follows from Lemma H.6, and the last step follows from Eq. (7).

Thus, we complete the proof. $\square$

Therefore, even adaptive queries can be answered accurately, since any adaptive query can be assumed in $\mathcal{B}$.

H.4 Effect of Different Norms on the Result

In the above proof, we have two different measure spaces, i.e. ℓ_∞ distance of ϵ_0-net (Definition H.2) and ℓ_1 Lipschitz (Lemma H.7).

One might ask, will the norm we choose in two spaces have an impact on the final result? We can show that the norm we choose currently is sufficient to use.

For different norms, the only differences in the proofs will be Lipschitz smoothness in Eq. (7) and the cardinality of ϵ_0-net, i.e. $|N|$ in Fact H.3.

Lemma H.9. *If we use ℓ_∞ ϵ_0-net and use ℓ_1 Lipschitz in Lemma H.8, we have copies of data structure $l = O(d\log(nR/p_f))$.*

Proof. If we use ℓ_∞ to bound the distance to net, Eq. (7) is:

$$\begin{aligned}|Z(y) - Z(q)| \leq & ~ nR_w \cdot \|y-q\|_1 \\ \leq & ~ nR_w \cdot d\|y-q\|_\infty \\ \leq & ~ nR_w \cdot d\epsilon_0\end{aligned}$$

where the first step follows from Lemma H.7, the second step follows from $\|x\|_1 \leq d\|x\|_\infty$ for $x \in \mathbb{R}^d$, and the last step follows from ℓ_∞ ϵ_0-net.

Then, Eq. (8) is

$$|f(y) - Z(q)| \leq \alpha Z(q) + B + 2nR_w \cdot d \cdot \epsilon_0$$

For ℓ_∞ distance, we have $|N| \leq (5R/\epsilon_0)^d$ in Fact H.3.

We can choose $\epsilon_0 = \Theta(1/n)$ to hide $nR_w \cdot d \cdot \epsilon_0$ term in B in Lemma H.8. Thus,

$$\begin{aligned}l = & ~ O(\log(|N|/p_f) \\ = & ~ O(\log((R/\epsilon_0)^d/p_f) \\ = & ~ O(\log((nR)^d/p_f)) \\ = & ~ O(d\log(nR/p_f))\end{aligned}$$

where the last step follows from $\log(a^d/b) = O(d\log(a/b))$ for any $a > 1, 0 < b < 1, d > 1$. $\square$

Lemma H.10. *If we use ℓ_2 ϵ_0-net and use ℓ_1 Lipschitz in Lemma H.8, we have copies of data structure $l = O(d\log(nR/p_f))$.*

Proof. If we use ℓ_2 to bound the distance to net, Eq. (7) changes to be:

$$\begin{aligned}|Z(y) - Z(q)| \leq & ~ nR_w \cdot \|y-q\|_1 \\ \leq & ~ nR_w \cdot \sqrt{d} \cdot \|y-q\|_2 \\ \leq & ~ nR_w \cdot \epsilon_0\sqrt{d}\end{aligned}$$

where the first step follows from Lemma H.7, the second step follows from $\|x\|_1 \leq \sqrt{d} \cdot \|x\|_2$ for $x \in \mathbb{R}^d$, and the last step follows from ℓ_2 ϵ_0-net.

Then, Eq. (8) changes to be

$$|f(y) - Z(q)| \leq \alpha Z(q) + B + 2nR_w \cdot \epsilon_0\sqrt{d}$$

For ℓ_2 distance, we also have $|N| \leq (5R/\epsilon_0)^d$ in Fact H.4.

We can choose $\epsilon_0 = \Theta(1/n)$ to hide $nR_w \cdot \sqrt{d} \cdot \epsilon_0$ term in B in Lemma H.8. Thus,

$$\begin{aligned}l = & ~ O(\log(|N|/p_f) \\ = & ~ O(\log((R/\epsilon_0)^d/p_f) \\ = & ~ O(\log((nR)^d/p_f)) \\ = & ~ O(d\log(nR/p_f))\end{aligned}$$

where the last step follows from $\log(a^d/b) = O(d\log(a/b))$ for any $a > 1, 0 < b < 1, d > 1$. $\square$

Lemma H.11. *If we use ℓ_1 ϵ_0-net and use ℓ_1 Lipschitz in Lemma H.8, we have copies of data structure $l = O(d\log(ndR/p_f))$.*

Proof. If we use ℓ_1 to bound the distance to net, Eq. (7) changes to be:

$$\begin{aligned}|Z(y) - Z(q)| &\leq nR_w \cdot \|y - q\|_1 \\ &\leq nR_w \cdot \epsilon_0\end{aligned}$$

where the first step follows from Lemma H.7, and the last step follows from ℓ_1 ϵ_0-net.

Then, Eq. (8) changes to be

$$|f(y) - Z(q)| \leq \alpha Z(q) + B + 2nR_w \cdot \epsilon_0$$

For ℓ_1 distance, we have $|N| \leq (5R\sqrt{d}/\epsilon_0)^d$.

We can choose $\epsilon_0 = \Theta(1/n)$ to hide $nR_w \cdot \epsilon_0$ term in B in Lemma H.8. Thus,

$$\begin{aligned}l &= O(\log(|N|/p_f) \\ &= O(\log((R\sqrt{d}/\epsilon_0)^d/p_f) \\ &= O(\log((nR\sqrt{d})^d/p_f)) \\ &= O(d\log(nRd/p_f))\end{aligned}$$

where the last step follows from $\log(a^d/b) = O(d\log(a/b))$ for any $a > 1, 0 < b < 1, d > 1$. $\square$

From the above analysis, we can show that ℓ_∞ or ℓ_2 ϵ_0-net is slightly better than ℓ_1 ϵ_0-net.

- ℓ_∞ ϵ_0-net, Lemma H.9: we have $l = O(d\log(nR/p_f))$.
- ℓ_2 ϵ_0-net, Lemma H.10: we have $l = O(d\log(nR/p_f))$.
- ℓ_1 ϵ_0-net, Lemma H.11: we have $l = O(d\log(nRd/p_f))$.

Thus, the norm we choose for ϵ_0-net is sufficient good.

I SOFTMAX ACTIVATION

In this section, we introduce how we extend previous ℓ_1 distance results to the Softmax activation function, which is the most widely used distance measure in attention mechanism based models.

In Section I.1, we show how to extend to the Softmax distance function in Lemma I.6. In Section I.2, we show how to adjust our algorithms. In Section I.3, we extend our algorithm to be robust to adaptive query. In Section I.4, we give the proof of our main result Theorem 3.1.

I.1 EXPONENTIAL INNER PRODUCT

In this section, we show how we obtain the Softmax distance using ℓ_2^2 distance query. First, we provide some helpful results from Alman & Song (2023).

Definition I.1 (Definition 3.1 in Alman & Song (2023))**.** *Let $r \geq 1$ denote a positive integer. Let $\epsilon \in (0, 0.1)$ denote an accuracy parameter. Given a matrix $A \in \mathbb{R}_{\geq 0}^{n\times n}$, we say $\widetilde{A} \in \mathbb{R}_{\geq 0}^{n\times n}$ is an (ϵ, r)-approximation of A if*

- *$\widetilde{A} = U_1 \cdot U_2^\top$ for some matrices $U_1, U_2 \in \mathbb{R}^{n\times r}$ (i.e., $\widetilde{A}$ has rank at most r), and*
- *$|\widetilde{A}_{i,j} - A_{i,j}| \leq \epsilon \cdot A_{i,j}$ for all $(i, j) \in [n]^2$.*

Lemma I.2 (Lemma 3.4 in Alman & Song (2023))**.** *Suppose $Q, K \in \mathbb{R}^{n\times d}$, with $\|Q\|_\infty \leq R$, and $\|K\|_\infty \leq R$. Let $A := \exp(QK^\top/d) \in \mathbb{R}^{n\times n}$. For accuracy parameter $\epsilon \in (0, 0.1)$, there is a positive integer s bounded above by*

$$s = O\Big(\max\Big\{\frac{\log(1/\epsilon)}{\log(\log(1/\epsilon)/R)}, R^2\Big\}\Big), \tag{9}$$

and a positive integer r bounded above by

$$r \leq \binom{2s+2d}{2s} \tag{10}$$

such that: There is a matrix $\widetilde{A} \in \mathbb{R}^{n \times n}$ that is an (ϵ, r)-approximation (Definition I.1) of $A \in \mathbb{R}^{n \times n}$. Furthermore, the matrices U_1 and U_2 defining $\widetilde{A}$ can be computed in $O(n \cdot r)$ time.

Here we consider the vector version of Lemma I.2.

Definition I.3. *We define $\Gamma_{R,s} := \max_{j \in [s]} \frac{R^j}{\sqrt{j!}}$.*

Then, we have $P(x) : [0, R]^d \to [0, \Gamma_{R,s}]^r$ where $P(\cdot)$ is polynomial kernel function defined in Alman & Song (2023).

Remark I.4. *We use $\Gamma_{R,s}$ to denote the value range of our polynomial kernel methods function, i.e., $P(x) : [0, R]^d \to [0, \Gamma_{R,s}]^r$. The factorial term in $\Gamma_{R,s}$ comes from Taylor approximation coefficients. We take the maximum overall s order approximation terms to get the upper bound of our value range.*

We use the polynomial approximation method, which has been applied to accelerate Transformer model extensively Alman & Song (2023; 2024a;b); Liang et al. (2024e;b).

Lemma I.5 (Polynomial approximation)**.** *For any accuracy parameter $\epsilon_s \in (0, 0.1)$, let $R \geq 1$, and let $P(x) : [0, R]^d \to [0, \Gamma_{R,s}]^r$ be the s-th order polynomial kernel function defined in Alman & Song (2023) where $r \leq \binom{2s+2d}{2s}$ and $s = O(\max\{\frac{\log(1/\epsilon_s)}{\log(\log(1/\epsilon_s)/R)}, R^2\})$. Then, for any $x, y \in [0, R]^d$, we have*

$$|P(x)^\top P(y) - \exp(x^\top y/d)| \leq \epsilon_s \cdot \min\{\exp(x^\top y/d), P(x)^\top P(y)\}$$

Furthermore, the vectors $P(x)$ and $P(y)$ can be computed in $O(r)$ time.

Proof. Let $n = 1$. The proof follows from directly applying Lemma I.2. $\square$

Using the results from Alman & Song (2023) above, we can extend our results to Softmax activation.

Lemma I.6 (Weighted Softmax approximation, formal version of Lemma C.7)**.** *Let accuracy parameter be $\epsilon_s \in (0, 0.1)$. Let $R \geq 1$. Let $r \leq \binom{2s+2d}{2s}$ and $s = O(\max\{\frac{\log(1/\epsilon_s)}{\log(\log(1/\epsilon_s)/R)}, R^2\})$. Let $P(x) : [0, R]^d \to [0, \Gamma_{R,s}]^r$ be the s-th order polynomial kernel function defined in Lemma I.5. Then we can approximate exponential inner product using polynomial kernel function:*

$$\begin{aligned} w^\top \exp(Xy/d) = & -\frac{1}{2} \sum_{j \in [r]} \sum_{i \in [n]} w_i |P(x_i)_j - P(y)_j|^2 + \frac{1}{2} \sum_{i \in [n]} w_i (\|P(x_i)\|_2^2 + \|P(y)\|_2^2) \\ & + O(w^\top \exp(Xy/d) \cdot \epsilon_s) \end{aligned}$$

Moreover, the vectors $P(\cdot)$ can be computed in $O(r)$ time.

Proof. From Lemma I.5, we can use polynomial kernel to approximate the Softmax function:

$$w^\top \exp(Xy/d) = \sum_{i \in [n]} w_i P(x_i)^\top P(y) + O(w^\top \exp(Xy/d) \cdot \epsilon_s).$$

The proof of approximation error and time complexity of constructing $P(\cdot)$ follows from Lemma I.5.

Then, we can show

$$\begin{aligned} 2 \sum_{i \in [n]} w_i P(x_i)^\top P(y) = & -\sum_{i \in [n]} w_i \|P(x_i) - P(y)\|_2^2 + \sum_{i \in [n]} w_i (\|P(x_i)\|_2^2 + \|P(y)\|_2^2) \\ = & -\sum_{j \in [r]} \sum_{i \in [n]} w_i |P(x_i)_j - P(y)_j|^2 + \sum_{i \in [n]} w_i (\|P(x_i)\|_2^2 + \|P(y)\|_2^2) \end{aligned}$$

where the first step follows from $\|x - y\|_2^2 = \|x\|_2^2 + \|y\|_2^2 - 2\langle x, y \rangle$, and the second step follows $\|x\|_2^2 = \sum_{j=1}^d |x_j|^2$ for $x \in \mathbb{R}^d$. $\square$

I.2 ALGORITHM MODIFICATIONS

Based on Lemma I.6, we can now extend our DP algorithms to handle Softmax activation. First, we need to construct $P(y)$ and $P(x_i)$ for $i \in [n]$, each costing $O(r)$ time. Then, for the second term in Lemma I.6, i.e. $\frac{1}{2}\sum_{i\in[n]} w_i(\|P(x_i)\|_2^2 + \|P(y)\|_2^2)$, we don't need to add DP noises in it; instead, we calculate this term exactly, preprocess it, and store the results in the algorithm. For the first term, $-\frac{1}{2}\sum_{j\in[r]}\sum_{i\in[n]} w_i|P(x_i)_j - P(y)_j|^2$, we can adjust our high dimensional DP distance query algorithm to solve it.

Due to the decomposability of ℓ_p^p norm, i.e.

$$\sum_{i\in[n]} w_i\|x_i - y\|_p^p = \sum_{j\in[d]}\sum_{i\in[n]} w_i|x_{i,j} - y_j|^p,$$

we can compute ℓ_2^2 norm easily (see details in Lemma E.2). We then show how to extend our one dimensional ℓ_1 distance algorithm (Algorithm 5 and 6) to ℓ_2^2 distance with minor modifications.

Theorem I.7 (DPTREEDISTANCE ℓ_2^2 distance)**.** *With α scaled down by a factor of 2 and all* QUERY *instead multiplied by $R^2/(1+\alpha/2)^{2j}$ in Lines 8 and 13 of Algorithm 6, i.e., from*

- *Lines 8 and 13: Value $\leftarrow$ Value + $\mathcal{D}$.*QUERY*$(l_j, r_j) \cdot \frac{R}{(1+\alpha)^j}$*

to

- *Lines 8 and 13: Value $\leftarrow$ Value + $\mathcal{D}$.*QUERY*$(l_j, r_j) \cdot \frac{R^2}{(1+\alpha/2)^{2j}}$.*

The data structure DPTREEDISTANCE *(Algorithm 5,6) uses $O(n)$ spaces to solve weighted ℓ_2^2-distance query problem for dataset $X \subset [0, R]$ and support the following operations:*

- INIT$(X \subset [0, R], n \in \mathbb{N}_+, w \in [-R_w, R_w]^n, \epsilon \in (0, 1), \delta \in (0, 1))$. *(Algorithm 5) It takes $O(n)$ time to initialize the data structure.*
- DISTANCEQUERY$(y \in [0, R], \alpha \in (0, 1))$. *(Algorithm 6)*

  It takes $O(\alpha^{-1}\log^2 n)$ time to output a number z such that

  - *the process of output z satisfies (ϵ, δ)-DP private, which computes $\sum_{i\in[n]} w_i|y - x_i|^2$,*
  - *$|z - \sum_{i\in[n]} w_i|y - x_i|^2| \le \alpha\sum_{i\in[n]} w_i|y - x_i|^2 + O(\epsilon^{-1}\alpha^{-1/2}R^2R_w\log^{3/2} n)$,*
  - *it holds with probability 0.99.*

Proof. The proof is similar to that of Theorem F.6, except that now our additive error includes R increased by a power of 2, i.e., from $O(\epsilon^{-1}\alpha^{-1/2}RR_w\log^{3/2} n)$ to $O(\epsilon^{-1}\alpha^{-1/2}R^2R_w\log^{3/2} n)$. □

Now we can give our result that can answer Softmax query.

Theorem I.8 (Softmax query, formal version of Theorem 4.2)**.** *Let $R \ge 1$. Let $r \le \binom{2s+2d}{2s}$ and $s = O(\max\{\frac{\log(1/\epsilon_s)}{\log(\log(1/\epsilon_s)/R)}, R^2\})$. Let $\Gamma_{R,s}$ be defined in Definition I.3. Let accuracy parameter be $\epsilon_s \in (0, 0.1)$. There is a data structure* DPTREESOFTMAX *(Algorithm 3) that uses $O(nr)$ spaces to solve Softmax query problem for dataset $X \subset [0, R]^d$ and support the following operations:*

- INIT$(X \subset [0, R]^d, n \in \mathbb{N}_+, w \in [-R_w, R_w]^n, \epsilon \in (0, 1), \delta \in (0, 1), \delta' \in (0, 1), c \in (0, 0.1), \epsilon_s \in (0, 0.1))$. *(Algorithm 3) It takes $O(nr)$ time to initialize the data structure.*
- DISTANCEQUERY$(y \in [0, R]^d, \alpha \in (0, 1))$. *(Algorithm 3) It takes $O(\alpha^{-1}r\log^2 n)$ time to output a number z such that*

  - *the process of output z satisfies $(\epsilon, \delta + \delta')$-DP private, which computes $w^\top \exp(Xy/d)$,*

– $|z - w^\top \exp(Xy/d)| \leq (\alpha + \epsilon_s) \cdot w^\top \exp(Xy/d)$
$+ O(\epsilon^{-1}\alpha^{-1/2}\Gamma_{R,s}^2 R_w r\sqrt{\log(1/\delta')} \cdot \log^{3/2} n)$,

– *it holds with probability* 0.99.

Proof. Let $P_{wx} := \sum_{i\in[n]} w_i \|P(x_i)\|_2^2$ and $s_w := \sum_{i\in[n]} w_i$. Observe that $P_{wx} = \sum_{i\in[n]} w_i \|P(x_i) - 0\|_2^2$, meaning we can calculating P_{wx} using query 0. Similarly, $s_w = \sum_{i\in[n]} w_i \|\mathbf{1}_n - 0\|_2^2$, meaning we can calculating s_w using weight $\mathbf{1}_n$ and query 0. Thus, we compute P_{wx}, s_w in Line 20 and 23 in Algorithm 3 in this way.

From the privacy proof of Lemma G.1 and the way we choose privacy parameters, similarly we get the output process of calculating P_{wx} and Value is $(\epsilon/3, \delta/3 + \delta'/2)$-DP. Also, the output process of calculating s_w is $(\epsilon/3, \delta/3)$-DP. Then, by Fact B.8, overall process is $(\epsilon, \delta + \delta')$-DP in Line 32 of Algorithm 3.

We then show the time complexity. From Lemma I.6, we know that constructing $P(\cdot)$ requires $O(r)$ time. In the first for loop of INIT, the dominating time consumption is $O(nr)$. The second for loop also has a time complexity of $O(nr)$. Therefore, the total time complexity for INIT is $O(nr)$. In the DISTANCEQUERY function, constructing $P(y)$ takes $O(r)$ time. Within the for loop, it requires $O(\alpha^{-1} r \log^2 n)$. Thus, the total time complexity for DISTANCEQUERY is $O(\alpha^{-1} r \log^2 n)$.

The space complexity is $O(nr)$, since storing the $n \times r$ matrix P is the dominating factor.

The proof of the error follows from the triangle inequality by combining the errors in Lemma I.6 and Theorem I.7. Here, we omit the constant factors of 2 and 3 used for the privacy guarantee in Algorithm 3, incorporating it into the big-O notation for the error analysis. □

I.3 ADAPTIVE SOFTMAX

In this section, we show how to make Algorithm 3 robust to adaptive query. We follow the same idea from Section H. We notice that, in the Softmax activation, we have query function $Z(y) := w^\top \exp(Xy/d)$ different from the ℓ_1-distance in Section H. Therefore, we need to re-calculate Lipschitz constant first.

Lemma I.9 (Lipschitz of weighted Softmax)**.** *If the following conditions hold:*

- *Let data set $X \in [0, R]^{n\times d}$, weights $w \in [-R_w, R_w]^n$, query $y \in [0, R]^d$.*
- *Let $Z(y) := w^\top \exp(Xy/d)$.*
- *Let $L = nd^{-1/2} R R_w \exp(R^2)$.*

Then, we have $Z(y)$ is L-Lipschitz (note that we have ℓ_1 Lipschitz here).

Proof. We can show

$$\begin{aligned}
|Z(y) - Z(\widetilde{y})| = & |\sum_{i\in[n]} w_i \exp(x_i^\top y/d) - \sum_{i\in[n]} w_i \exp(x_i^\top \widetilde{y}/d)| \\
\leq & \sum_{i\in[n]} |w_i| \cdot |\exp(x_i^\top y/d) - \exp(x_i^\top \widetilde{y}/d)| \\
\leq & \sum_{i\in[n]} |w_i| \exp(R^2) |x_i^\top y/d - x_i^\top \widetilde{y}/d| \\
\leq & \sum_{i\in[n]} |w_i| \exp(R^2) \|x_i\|_2 \cdot \|y - \widetilde{y}\|_2/d \\
\leq & nR_w \exp(R^2)\sqrt{d}R \cdot \|y - \widetilde{y}\|_2/d \\
\leq & nd^{-1/2} R R_w \exp(R^2) \|y - \widetilde{y}\|_1
\end{aligned}$$

where the first step follows from definition of $Z(y), Z(\widetilde{y})$, the second step follows from triangular inequality, the third step follows from Fact B.4, the fourth step follows from Cauchy–Schwarz inequality $|u^\top v| \leq \|u\|_2 \cdot \|v\|_2$ for $u, v \in \mathbb{R}^d$, the fifth step follows from $w_i \in [-R_w, R_w]$ and $x_i \in [0, R]^d$, and the last step follows from $\|u\|_2 \leq \|u\|_1$ for $u \in \mathbb{R}^d$. $\square$

Then we can show how to extend our algorithm to be robust to adaptive query.

Lemma I.10 (Adaptive Softmax, formal version of Lemma C.8)**.** *If the following conditions hold:*

- *Let N be the ℓ_∞ ϵ_0-net of $\mathcal{B}$, and $|N|$ be the size of net N.*
- *Let data set $X \in [0, R]^{n \times d}$, weights $w \in [-R_w, R_w]^n$, query $y \in [0, R]^d$.*
- *Let relative error parameter $\alpha \in (0, 1)$, the failure probability $p_f \in (0, 0.01)$.*
- *We create $l = O(\log((R/\epsilon_0)^r / p_f))$ independent copies of data structure $\{\text{DPTreeSoftmax}_j\}_{j=1}^l$ (Algorithm 3) and take the median of the outputs with each data structure instantiated with $(\epsilon/l, (\delta + \delta')/l)$-DP.*
- *Let $f(y) := \mathrm{Median}(\{\text{DPTreeSoftmax}_j.\text{DistanceQuery}(y, \alpha)\}_{j=1}^l)$.*
- *Let $Z(y) := w^\top \exp(Xy/d)$, where $Z(y)$ is L-Lipschitz with $L = nd^{-1/2} R R_w \exp(R^2)$.*
- *Let $B = O(\epsilon^{-1} \alpha^{-1/2} l \Gamma_{R,s}^2 R_w r \sqrt{\log(l/\delta')} \cdot \log^{3/2} n)$.*

Then with probability $1 - p_f$, for all query points $q \in \mathcal{B}$, there exists a point $y \in N$ which is the closest to q, we can have the process of outputting the median of l responses is $(\epsilon, \delta + \delta')$-DP and the error satisfies

$$|f(y) - Z(q)| \leq (\alpha + \epsilon_s) Z(q) + B + 2n\sqrt{d} R R_w \exp(R^2) \epsilon_0.$$

Proof. The proof follows from the same idea as the proof of Lemma H.8, except that we use Theorem I.8 and the Lipschitz in Lemma I.9. $\square$

Theorem I.11 (Adaptive query Softmax data structure, formal version of Theorem 4.4)**.** *Let $R \geq 1$. Let $r \leq \binom{2s+2d}{2s}$ and $s = O(\max\{\frac{\log(1/\epsilon_s)}{\log(\log(1/\epsilon_s)/R)}, R^2\})$. Let $\Gamma_{R,s}$ be defined in Definition I.3. Let accuracy parameter be $\epsilon_s \in (0, 0.1)$. Let $X \in [0, R]^{n \times d}$ be the dataset, $w \in [-R_w, R_w]^n$ be weights, $y \in [0, R]^d$ be the query, $\alpha \in (0, 1)$ be the relative error parameter, and p_f be the failure probability parameter. Let $l = O(r \log(dR/(\epsilon_s p_f)))$. There is a data structure* DPTreeSoftmaxAdaptive *(Algorithm 8) that uses $O(lnr)$ spaces to solve weighted Softmax query problem for dataset $X \subset [0, R]^d$ and support the following operations:*

- Init$(X \subset [0, R]^d, n \in \mathbb{N}_+, w \in [-R_w, R_w]^n, \epsilon \in (0, 1), \delta \in (0, 1), \delta' \in (0, 1), c \in (0, 0.1), \epsilon_s \in (0, 0.1), p_f \in (0, 0.01))$. *(Algorithm 8) It takes $O(lnr)$ time to initialize the data structure.*
- DistanceQuery$(y \in [0, R]^d, \alpha \in (0, 1))$. *(Algorithm 8) It takes $O(\alpha^{-1} lr \log^2 n)$ time to output a number z such that*
  - *the process of output z satisfies $(\epsilon, \delta + \delta')$-DP private, which computes $w^\top \exp(Xy/d)$,*
  - $|z - w^\top \exp(Xy/d)| \leq (\alpha + \epsilon_s) \cdot w^\top \exp(Xy/d)$ $+\ O(\epsilon^{-1} \alpha^{-1/2} l \Gamma_{R,s}^2 R_w r \sqrt{\log(l/\delta')} \cdot \log^{3/2} n)$,
  - *it holds with probability $1 - p_f$ (where p_f is used in l),*
  - *it is robust to adaptive query.*

Proof. We only need to show how to pick ϵ_0 in the parameter l, because everything else is the same as Lemma I.10. We know the additive error introduced by adaptive query is $E_a := O(n\sqrt{d} R R_w \exp(R^2) \epsilon_0)$ and the relative error introduced by polynomial kernel approximation is $E_p := w^\top \exp(Xy/d) \cdot \epsilon_s$. It can be shown that:

$$E_p := w^\top \exp(Xy/d) \cdot \epsilon_s$$

$$\leq \epsilon_s \|w\|_2 \cdot \|\exp(Xy/d)\|_2$$
$$= O(nR_w \epsilon_s \exp(R^2))$$

where the first step follows from definition of E_p, the second step follows from Cauchy–Schwarz inequality, and the last step follows from $w \in [-R_w, R_w]^n$, $X \in [0,R]^{n \times d}$, and $y \in [0,R]^d$.

Picking $\epsilon_0 = \Theta(\frac{\epsilon_s}{\sqrt{d}R})$, we can hide the error of adaptive query E_a in E_p. Thus, we have

$$\begin{aligned} l &= O(\log((R/\epsilon_0)^r/p_f)) \\ &= O(\log((\sqrt{d}R^2/\epsilon_s)^r/p_f)) \\ &= O(r\log(dR/(\epsilon_s p_f))) \end{aligned}$$

where the first step comes from the definition of l, the second step comes from picking $\epsilon_0 = \Theta(\frac{\epsilon_s}{\sqrt{d}R})$, and the last step follows from $\log(a^d/b) = O(d\log(a/b))$ for any $a > 1, 0 < b < 1, d > 1$. $\square$

Algorithm 8 Adaptive query data structure

datastructure DPTreeSoftmaxAdaptive ▷ Theorem 4.4
members
  $\mathcal{D}_1, \dots, \mathcal{D}_{O(r\log(dR/(\epsilon_s p_f)))}$: DPTreeSoftmax ▷ Algorithm 3
end members
procedure Init($X \subset [0,R]^d, n \in \mathbb{N}_+, w \in [-R_w, R_w]^n, \epsilon \in (0,1), \delta \in (0,1), \delta' \in (0,1), c \in (0,0.1)), \epsilon_s \in (0,0.1), p_f \in (0,0.01)$)
  $l \leftarrow O(r\log(dR/(\epsilon_s p_f)))$
  for $i = 1 \to l$ **do**
    $\mathcal{D}_i$.Init$(X, n, w, \epsilon/l, \delta/l, \delta'/l, c, \epsilon_s)$
  end for
end procedure
procedure DistanceQuery($y \in [0,R]^d, \alpha \in (0,1)$)
  $l \leftarrow O(r\log(dR/(\epsilon_s p_f)))$
  $r \leftarrow 0^l$
  for $i = 1 \to l$ **do**
    $r_i \leftarrow \mathcal{D}_i$.DistanceQuery$(y, \alpha)$
  end for
  return Median of r
end procedure
end datastructure

I.4 Proof of Main Result

In this section, we give the proof of our main result of Theorem 3.1.

Theorem I.12 (Softmax cross-attention, formal version of Theorem 3.1)**.** *Let Q, K, V, Attn be defined in Definition 1.1. Let $\alpha \in (0,1)$ be the relative error parameter and p_f be the probability of failure parameter. Let r, s, ϵ_s be parameters of polynomial kernel methods (Lemma C.7). Let $\Gamma_{R,s} := \max_{j \in [s]} \frac{R^j}{\sqrt{j!}}$ (Definition I.3). Let $l = O(r\log(dR/(\epsilon_s p_f)))$. There is a data structure* DPTreeCrossAttention *(Algorithm 1) that uses $O(lnrd)$ spaces to ensure cross-attention DP and supports the following operations:*

- Init$(K, V, \epsilon \in (0,1), \delta \in (0,1), \delta' \in (0,1), c \in (0,0.1), \epsilon_s \in (0,0.1), p_f \in (0,0.01))$ *(Algorithm 1). It takes $O(lnrd)$ time to initialize.*
- *At query time, for user input Q, we process one token at a time by passing the i-th row of Q, denoted $Q_i \in [0,R]^d$, to* Query$(Q_i, \alpha \in (0,1))$ *(Algorithm 1) for each $i \in [m]$. It takes $O(\alpha^{-1} ldr \log^2 n)$ time to output an entry z in $\mathrm{Attn}(Q,K,V)$ such that*
  - *the process of output z satisfies $(\epsilon, \delta + \delta')$-DP,*

  - *the process of output z has error*
    $$\widetilde{O}(n^{-1}\epsilon^{-1}R\exp(R^2+2R\epsilon^{-1})((1+\alpha+\epsilon_s)\cdot(AV)_{i,k}+\epsilon^{-1}\alpha^{-1/2}l\Gamma_{R,s}^2R_wr\sqrt{\log(l/\delta')}))$$
    where $\widetilde{O}$ hide logarithm dependecy on n,
  - *it holds with probability $1-p_f$ (where p_f is used in l),*
  - *it is robust to adaptive query.*

Proof. We first prove the privacy and then prove error for each coordinate of the output O of Algorithm 1.

Proof of Privacy:

From Theorem I.11, $\mathcal{D}_k$.DISTANCEQUERY for $k\in\{0,1,\dots,d\}$ in Algorithm 1 answer $(\epsilon/2,\delta/2+\delta'/2)$-DP queries that are robust to adpative queries. By Fact B.8, the procedure for calculating each coordinate of vector O is $(\epsilon,\delta+\delta')$-DP in Line 15 of Algorithm 1.

Proof of Error:

We prove the error bound of the cross-atteniton module. We omit the constant factor of 2 used for the privacy guarantee in Algorithm 1, incorporating it into the big-O notation for the error analysis. Let AV be the true value and $\widetilde{AV}$ be the noisy value. Let D be the true value and $\widetilde{D}$ be the noisy value. First, we use triangular inequality to decompose the error:

$$\begin{aligned}&|(D^{-1}AV)_{i,k}-(\widetilde{D}^{-1}\widetilde{AV})_{i,k}|\\ \leq&|(D^{-1}AV)_{i,k}-(D^{-1}\widetilde{AV})_{i,k}|+|(D^{-1}\widetilde{AV})_{i,k}-(\widetilde{D}^{-1}\widetilde{AV})_{i,k}| \end{aligned}\tag{11}$$

We now prove for each term.

Part 1: Error bound for AV

From Section 3, we know that we can ensure matrix AV in cross-attention computation satisfies DP. Next, from Theorem 4.4, for $i\in[m],j\in[n],k\in[d]$, we have $(AV)_{i,k}$ is $(\epsilon,\delta+\delta')$-DP and also robust to adaptive query.

Let $\zeta:=\epsilon^{-1}\alpha^{-1/2}l\Gamma_{R,s}^2R_wr\sqrt{\log(l/\delta')}\cdot\log^{3/2}n$ denote the additive error. Then, from Theorem I.11, we have

$$|(AV)_{i,k}-(\widetilde{AV})_{i,k}|\leq(\alpha+\epsilon_s)\cdot(AV)_{i,k}+O(\zeta)\tag{12}$$

For $D_{i,i}$, we can show

$$D_{i,i}=(A\cdot\mathbf{1}_n)_i=\sum_{j=1}^n\exp(\langle Q_i,K_j\rangle/d)\geq n\tag{13}$$

because $\langle Q_i,K_j\rangle\geq0$ for bounded Q,K.

Finally, we can show the error of first term in Eq. (11) is bounded by

$$\begin{aligned}|(D^{-1}AV)_{i,k}-(D^{-1}\widetilde{AV})_{i,k}|&=|D_{i,i}^{-1}((AV)_{i,k}-(\widetilde{AV})_{i,k})|\\&=|D_{i,i}^{-1}|\cdot|((AV)_{i,k}-(\widetilde{AV})_{i,k})|\\&\leq n^{-1}(\alpha+\epsilon_s)\cdot(AV)_{i,k}+O(n^{-1}\zeta)\end{aligned}$$

where the first step follows from definition, the second step follows from simple algebra, and the last step follows from Eq. (12) and (13).

Part 2: Error bound for D

We initialize one DPTREESOFTMAXADAPTIVE with INIT$(K,n,\mathbf{1}_n,\epsilon,\delta,\delta',c,\epsilon_s,p_f)$. Let us name this data structure D-DPTREE. Notice that we input $\mathbf{1}_n$ as the third argument.

We wish to bound $\|K_j-\widetilde{K}_j\|_2$ for $j\in[n]$. Observe that D-DPTREE only add noises in K, so $\|K_j-\widetilde{K}_j\|_2$ is the ℓ_2-norm of a d-dimension vector where corrdinates are independent truncated laplace noises. Similar to the proof of Lemma D.6, we prove the bound using Chebyshev's inequality. The variance of the sum of d independent truncated laplace noises is give by $\widetilde{O}(d\epsilon^{-2}\Delta^2)$. Then by Lemma B.3, with high probabiltiy, we have $\|K_j - \widetilde{K}_j\|_2 \leq \|K_j - \widetilde{K}_j\|_1 \leq \sqrt{d}/\epsilon$, where the sensitivity for D-DPTREE is 1 since we initialize D-DPTREE with $\mathbf{1}_n$ as third argument. Therefore, we have $\|K_j - \widetilde{K}_j\|_2 \leq \sqrt{d}/\epsilon$.

We show the lower bound of $\langle Q_i, \widetilde{K}_j \rangle$ below:

$$\begin{aligned}
\langle Q_i, \widetilde{K}_j \rangle =& \langle Q_i, \widetilde{K}_j - K_j + K_j \rangle \\
=& \langle Q_i, \widetilde{K}_j - K_j \rangle + \langle Q_i, K_j \rangle \\
\geq& \langle Q_i, \widetilde{K}_j - K_j \rangle \\
\geq& - \|Q_i\|_2 \cdot \|\widetilde{K}_j - K_j\|_2 \\
\geq& - \|Q_i\|_2 \cdot \sqrt{d}/\epsilon \\
\geq& - Rd/\epsilon
\end{aligned}$$

where the first step follows from simple algebra, the second step follows from linearity of inner product, the third step follows from $\langle Q_i, K_j \rangle \geq 0$ for bounded $Q_i, K_j \in [0, R]^d$, the fourth step follows from $\|K_j - \widetilde{K}_j\|_2 \leq \sqrt{d}/\epsilon$, the fifth step follows from Cauchy-Schwarz inequality, and the last step follows from $Q_i \in [0, R]^d$.

Thus, from above we have

$$\widetilde{D}_{i,i} = (\widetilde{A} \cdot \mathbf{1}_n)_i = \sum_{j=1}^n \exp(\langle Q_i, \widetilde{K}_j \rangle / d) \geq \sum_{j=1}^n \exp(-Rd/(d\epsilon)) = n \exp(-R/\epsilon). \tag{14}$$

Similarly, the upper bound of $\langle Q_i, \widetilde{K}_j \rangle$ is

$$\begin{aligned}
\langle Q_i, \widetilde{K}_j \rangle &= \langle Q_i, \widetilde{K}_j - K_j \rangle + \langle Q_i, K_j \rangle \\
&\leq \|Q_i\|_2 \cdot \|\widetilde{K}_j - K_j\|_2 + \|Q_i\|_2 \cdot \|K_j\|_2 \\
&\leq Rd/\epsilon + R^2 d = dR(R + 1/\epsilon)
\end{aligned}$$

Then, we can show the upper bound of $|D_{i,i} - \widetilde{D}_{i,i}|$ is

$$\begin{aligned}
|D_{i,i} - \widetilde{D}_{i,i}| &= |\sum_{j=1}^n \exp(\langle Q_i, K_j \rangle / d) - \sum_{j=1}^n \exp(\langle Q_i, \widetilde{K}_j \rangle / d)| \\
&\leq \sum_{j=1}^n |\exp(\langle Q_i, K_j \rangle / d) - \exp(\langle Q_i, \widetilde{K}_j \rangle / d)| \\
&\leq \sum_{j=1}^n \exp(dR(R + 1/\epsilon)/d) \cdot |\langle Q_i, K_j \rangle / d - \langle Q_i, \widetilde{K}_j \rangle / d)| \\
&\leq \sum_{j=1}^n \exp(R(R + 1/\epsilon)) \cdot \|Q_i\|_2 \cdot \|\widetilde{K}_j - K_j\|_2 / d \\
&\leq \sum_{j=1}^n \exp(R(R + 1/\epsilon)) \cdot Rd/(d\epsilon) \\
&= \epsilon^{-1} n R \exp(R(R + 1/\epsilon))
\end{aligned}$$

where the first step follows from Definition of D, the second step follows from triangular inequality, the third step follows from Fact B.4 and the upper bound of $\langle Q_i, \widetilde{K}_j \rangle$ above, the fourth step follows from Cauchy-Schwarz inequality, the fifth step follows from the upper bounds of $\|Q_i\|_2$ and $\|\widetilde{K}_j - K_j\|_2$, and the last step follows from simple algebra.

Then, we can show

$$\begin{aligned}
|D_{i,i}^{-1} - \widetilde{D}_{i,i}^{-1}| &= \frac{|D_{i,i} - \widetilde{D}_{i,i}|}{D_{i,i} \cdot \widetilde{D}_{i,i}} \\
&\leq \frac{|D_{i,i} - \widetilde{D}_{i,i}|}{n^2 \cdot \exp(-R/\epsilon)} \\
&\leq \frac{\epsilon^{-1} n R \exp(R(R + 1/\epsilon))}{n^2 \cdot \exp(-R/\epsilon)} \\
&= n^{-1}\epsilon^{-1} R \exp(R^2 + 2R/\epsilon)
\end{aligned}$$

where the first step follows from simple algebra, the second step follows from Eq.(13) and (14), the third step follows from the upper bound of $|D_{i,i} - \widetilde{D}_{i,i}|$ above, and the last step follows from simple algebra.

From Eq. (12), we have

$$|(\widetilde{AV})_{i,k}| \leq (1 + \alpha + \epsilon_s) \cdot (AV)_{i,k} + O(\zeta)$$

We consider the second term in Eq.(11). Then,

$$\begin{aligned}
|(D^{-1}\widetilde{AV})_{i,k} - (\widetilde{D}^{-1}\widetilde{AV})_{i,k}| &= |D_{i,i}^{-1}(\widetilde{AV})_{i,k} - \widetilde{D}_{i,i}^{-1}(\widetilde{AV})_{i,k}| \\
&= |D_{i,i}^{-1} - \widetilde{D}_{i,i}^{-1}| \cdot |(\widetilde{AV})_{i,k}| \\
&\leq n^{-1}\epsilon^{-1} R \exp(R^2 + 2R\epsilon^{-1})((1 + \alpha + \epsilon_s) \cdot (AV)_{i,k} + O(\zeta))
\end{aligned}$$

Part 3: Final error bound

Combining results from **Part 1 and 2**, the final error bound is

$$\begin{aligned}
&|(D^{-1}AV)_{i,k} - (\widetilde{D}^{-1}\widetilde{AV})_{i,k}| \\
\leq\ &|(D^{-1}AV)_{i,k} - (D^{-1}\widetilde{AV})_{i,k}| + |(D^{-1}\widetilde{AV})_{i,k} - (\widetilde{D}^{-1}\widetilde{AV})_{i,k}| \\
=\ &n^{-1}(\alpha + \epsilon_s) \cdot (AV)_{i,k} + O(n^{-1}\zeta) \\
&+\ n^{-1}\epsilon^{-1} R \exp(R^2 + 2R\epsilon^{-1})((1 + \alpha + \epsilon_s) \cdot (AV)_{i,k} + O(\zeta)) \\
\leq\ &n^{-1}\epsilon^{-1} R \exp(R^2 + 2R\epsilon^{-1})((1 + \alpha + \epsilon_s) \cdot (AV)_{i,k} + O(\zeta))
\end{aligned}$$

Therefore, we prove the error bound. $\square$