TOWARDS SAMPLING DATA STRUCTURES FOR TENSOR PRODUCTS

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper studies the computational challenges of attention-based models in artificial intelligence by introducing innovative sampling methods to accelerate attention computation in large language models (LLM). Inspired by the recent progress of LLM in real-life applications, we introduces a streaming sampler question for attention setting. Our approach significantly reduces the computational burden of traditional attention mechanisms while maintaining or enhancing model performance. We demonstrate these methods' effectiveness from theoretical perspective, including space, update time. Additionally, our framework exhibits scalability and broad applicability across various model architectures and domains.

1 INTRODUCTION

In recent years, the field of artificial intelligence has witnessed a significant paradigm shift with the advent of attention-based models, particularly in the domains of natural language processing and computer vision (Vaswani et al., 2017; Devlin et al., 2018; Liu et al., 2019; Yang et al., 2019; Brown et al., 2020; Zhang et al., 2022; Chowdhery et al., 2022; Touvron et al., 2023a;b; Inc., 2023; Manyika, 2023). At the heart of these models lies the attention mechanism (Vaswani et al., 2017), which has proven to be a powerful tool in enhancing the performance of deep learning networks. It enables models to focus on relevant parts of the input data, thereby facilitating a more nuanced and context-aware processing. However, as these models scale in size and complexity, the computational demands of the attention mechanism increase exponentially, posing significant challenges in terms of efficiency and scalability.

Traditional attention mechanisms (Vaswani et al., 2017), such as those used in Transformer models, require the computation of attention weights across all elements of the input sequence, leading to a quadratic increase in computational complexity with respect to the sequence length (Alman & Song, 2023; Kacham et al., 2023; Han et al., 2023; Zandieh et al., 2023). This computational burden becomes particularly pronounced in large-scale applications, hindering the deployment of attention-based models in resource-constrained environments and limiting their real-time processing capabilities. Furthermore, the high computational cost also exacerbates the environmental impact of training and deploying these models, due to increased energy consumption and carbon footprint.

The core question we ask in this paper then is:

Instead of computing all the entries explicitly, can we quickly sample only some important coordinates?

To address these challenges, our research introduces innovative sampling methods aimed at accelerating attention computation in deep learning models. By strategically sampling key elements from the input data, our approach significantly reduces the computational overhead associated with the attention mechanism, while maintaining, or even enhancing, the model's performance. This paper presents a comprehensive exploration of our proposed sampling techniques, detailing the underlying principles, implementation strategies, and the resultant gains in computational efficiency.

Our contributions can be summarized as follows:

- For the softmax distribution ($\langle \exp(Ax), \mathbf{1}_n \rangle^{-1} \exp(Ax)$), we prove an $\Omega(n)$ space streaming sampler algorithm lower bound. (See Theorem 4.4)

- As the softmax distribution has a strong lower bound, we then provide upper bounds for polynomial type samplers, i.e., L_2 sampling from Ax. There are three settings for various updates of A and x, (see Theorem 5.3, Theorem 5.5)
- For updating both A and x, we provide an upper bound (see Theorem 5.7). In addition, we also provide a lower bound (see Theorem 6.2).
- Toward tensor generalization, we will sample $(i_1, i_2) = i \in [n^2]$ approximately according to the ℓ_2 sampling distribution via using $O(nd)$ space, $O(n)$ update time (see Theorem 7.6). Note that the trivial result takes $O(n^2)$ space.

2 Related Work

On sampling. Given a vector $v \in U^n$ whose coordinates are elements from a universe U and a non-negative weight function $W : U \to \mathbb{R}^{\geq 0}$, a fundamental goal is to return an index $i \in \{1, \ldots, n\}$ with probability proportional to $W(v_i)$. The definition of U permits settings such as $U = \mathbb{R}^d$, so that each coordinate is a row of a matrix or a d-dimensional point, or U may be a subset of the set of all matrices or tensors. In perhaps the most well-studied setting, each coordinate is a real number, so that $U = \mathbb{R}$ and the weight function is chosen from the class $W(x) = |x|^p$ for $p \geq 0$. The problem is particularly interesting when the vector $v \in U^n$ is implicitly defined through a data stream, i.e., a sequence of m updates to the coordinates of v, and the goal is to perform the sampling procedure using space sublinear in n and m, and the existence of such L_p *sampling* algorithms was asked by Cormode et al. (2005) in 2005.

Monemizadeh & Woodruff (2010) partially answered this question in the affirmative by giving an L_p sampler using polylogarithmic space for $p \in [1, 2]$, although the sampling probabilities were distorted by a multiplicative $(1 + \epsilon)$ factor and an additive $\frac{1}{\mathrm{poly}(n)}$ factor. The space requirements of the algorithm were subsequently improved (Andoni et al., 2011; Jowhari et al., 2011) and extended to other choices of index domain U and weight function W (Cohen & Geri, 2019; Mahabadi et al., 2020; 2022), while retaining a multiplicative distortion in the sampling probability. Surprisingly, Jayaram & Woodruff (2021) showed that it is possible to achieve no multiplicative distortion in the sampling probabilities while using polylogarithmic space, while conversely Jayaram et al. (2022) showed that removing the additive distortion would require linear space, essentially closing the line of work studying the space complexity of L_p samplers. It should be noted however, achieving such guarantees in sub-polynomial update time while retaining the space guarantees remains an intriguing open question (Jayaram et al., 2022). For a more comprehensive background on samplers, we refer to the survey by Cormode & Jowhari (2019).

On tensors. In the realm of tensor decomposition, the canonical polyadic (CP) decomposition, specifically the CANDECOMP/PARAFAC method, stands out for its unique ability to break down tensors into rank-1 tensors in a singular way, distinct from matrix decomposition (Harshman, 1970; Song et al., 2016). This method, having applications in computational neuroscience, data mining, and statistical learning (Wang et al., 2015), emphasizes the rigidity and uniqueness of tensor decomposition. Earlier studies (Tsourakakis, 2010; Phan et al., 2013; Choi & Vishwanathan, 2014; Huang et al., 2013; Kang et al., 2012; Wang et al., 2014; Bhojanapalli & Sanghavi, 2015) have delved into efficient tensor decomposition methods. Subsequent works introduced methods for fast orthogonal tensor decomposition using random linear sketching techniques (Wang et al., 2015) and explored symmetric orthogonally decomposable tensors' properties, integrating spectral theory (Robeva, 2016; Robeva & Seigal, 2017). Additionally, importance sampling for quicker decomposition was proposed (Song et al., 2016). (Deng et al., 2023a) studies the tensor cycle low rank approximation problem.

In algebraic statistics, tensor decompositions are linked to probabilistic models, particularly in determining latent variable models' identifiability through low-rank decompositions of specific moment tensors (Allman et al., 2009a;b; Rhodes & Sullivant, 2012). Kruskal's theorem (Kruskal, 1977) was pivotal in ascertaining the precision of model parameter identification. However, this approach, assuming an infinite sample size, falls short in providing minimum sample size information necessary for learning model parameters within given error bounds. A more robust uniqueness guarantee is needed, ensuring that the low-rank decomposition of an empirical moment tensor approximates that of an actual moment tensor, thus offering more insight into empirical moment tensors' decomposition.

On sketching. The application of sketching and sampling techniques in numerical linear algebra has been remarkably effective, revolutionizing a broad spectrum of core tasks. These methods are crucial in linear programming (LP), as evidenced by Cohen et al. (2019); Jiang et al. (2021); Ye (2020); Gu & Song (2022), and have significantly impacted tensor approximation (Song et al., 2019a; Mahankali et al., 2022; Deng et al., 2023a). Sketching and sampling techniques also have been widely applied in matrix completion (Gu et al., 2023), matrix sensing (Qin et al., 2023c; Deng et al., 2023c), submodular function maximization (Qin et al., 2023a), dynamic sparsification (Deng et al., 2022a), dynamic tensor product regression (Reddy et al., 2022), and semi-definite programming (Song et al., 2022b). Additionally, sketching has been pivotal in iterative sparsification problems (Song et al., 2022a), adversarial training (Gao et al., 2022), kernel density estimation (Qin et al., 2022b), solving the distance oracle problem (Deng et al., 2022b), and empirical risk minimization (Lee et al., 2019; Qin et al., 2023b). Its applications furthermore extends to relational databases (Qin et al., 2022a) and Large Language Models (LLMs) research (Deng et al., 2023d;c; Gao et al., 2023b; Li et al., 2023).

On theoretical attention. A comprehensive body of research, including studies (Child et al., 2019; Kitaev et al., 2020; Wang et al., 2020; Daras et al., 2020; Katharopoulos et al., 2020; Chen et al., 2021; 2022; Zandieh et al., 2023; Alman & Song, 2023; Brand et al., 2023; Deng et al., 2023d; Kacham et al., 2023; Alman & Song, 2024; Han et al., 2023; Awasthi & Gupta, 2023; Marcus et al., 2022), has progressively shed light on the complexities and optimization of attention matrix computation. This exploration has been further enriched by insights into the effectiveness of attention mechanisms in Transformers (Dehghani et al., 2018; Vuckovic et al., 2020; Zhang et al., 2020; Edelman et al., 2021; Snell et al., 2021; Wei et al., 2021; Deng et al., 2023e;b). Among these, Zhao et al. (2023) revealed the adeptness of mid-scale masked language models in identifying syntactic elements, paving the way for innovations like partial parse tree reconstructions. Inspired the exponential mechanism in attention structure, Gao et al. (2023a) provides an analysis which shows exponential regression within the over-parameterized neural tangent kernel framework can converge. In the over-constrained setting, several work show the convergence for attention inspired regression problem (Li et al., 2023; Deng et al., 2023c)

Roadmap. In Section 3, we provide some standard notations and definitions in literature. In Section 4, we study the exponential sampler. In Section 5, we study the streaming upper for the ℓ_2 sampling problem, i.e., sampling coordinates from a vector Ax, where A and x may be updated across a data stream. In Section 6, we present lower bounds for the same ℓ_2 sampling problem. Finally, in Section 7, we discuss the tensor sampling problem.

3 Preliminaries

For any positive integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. We use $\mathbb{E}[\cdot]$ to denote the expectation. We use $\Pr[\cdot]$ to denote the probability. We use $\mathbf{1}_n$ to denote a length-n vector where all the entries are ones. Given two length-n vector, we use $\langle x, y\rangle$ to denote the inner product between x and y, i.e, $\langle x, y\rangle := \sum_{i=1}^n x_i y_i$. For a vector $x \in \mathbb{R}^n$, we use $\exp(x) \in \mathbb{R}^n$ to denote a vector that has length n and the i-th entry is $\exp(x_i)$. For a matrix A, we use $\exp(A)$ to denote the matrix that (i, j)-th coordinate is $\exp(A_{i,j})$. For a vector x, we use $\|x\|_2 := (\sum_{i=1}^n x_i^2)^{1/2}$. We use $\|x\|_1 := \sum_{i=1}^n |x_i|$. We use $\|x\|_0$ to denote the ℓ_0 norm of x, which is the number of nonzero entries in x. We use $\|x\|_\infty$ to denote the ℓ_∞ norm of x, which is $\max_{i\in[n]} |x_i|$.

Let n_1, n_2, d_1, d_2 be positive integers. Let $A \in \mathbb{R}^{n_1\times d_1}$ and $B \in \mathbb{R}^{n_2\times d_2}$. We define the Kronecker product between matrices A and B, denoted $A\otimes B \in \mathbb{R}^{n_1 n_2\times d_1 d_2}$, as $(A\otimes B)_{(i_1-1)n_2+i_2,(j_1-1)d_2+j_2}$ is equal to $A_{i_1,j_1} B_{i_2,j_2}$, where $i_1 \in [n_1], j_1 \in [d_1], i_2 \in [n_2], j_2 \in [d_2]$.

We use $\mathrm{poly}(n)$ to denote n^C where $C > 1$ is some constant. For any function f, we use $\widetilde{O}(f)$ to denote $f \cdot \mathrm{poly}(\log f)$. For two sets A and B, we use $A \cap B$ to denote their intersection. We use $|A \cap B|$ to denote the cardinality of $A \cap B$. We use $A \cup B$ to denote the union of A and B.

3.1 TensorSketch

TensorSketch (Pagh, 2013) has been extensively used in many sketching and optimizations (Song et al., 2019b; Diao et al., 2018; 2019; Ahle et al., 2020; Song et al., 2021a;b; 2022a; Zhang, 2022;

Song et al., 2023). Song et al. (2022a) defined TensorSparse by compose Sparse embedding (Nelson & Nguyên, 2013; Cohen, 2016) with a tensor operation (Pagh, 2013).

Definition 3.1 (TensorSparse, see Definition 7.6 in Song et al. (2022a)). *Let $h_1, h_2 : [n] \times [s] \to [m/s]$ be $O(\log 1/\delta)$-wise independent hash functions and let $\sigma_1, \sigma_2 : [n] \times [s] \to \{\pm 1\}$ be $O(\log 1/\delta)$-wise independent random sign functions. Then, the degree two tensor sparse transform, $S : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$ is given as:*

$$R_{r,(i,j)} = \exists k \in [s] : \sigma_1(i,k)\sigma_2(j,k)/\sqrt{s} \cdot \mathbf{1}[((h_1(i,k) + h_2(j,k)) \text{ mod } m/s) + (k-1)m/s = r]$$

For $s = 1$, the above definition becomes TensorSketch (Pagh, 2013).

4 Exponential Sampler

In this section, we define and consider exponential samplers. We then show strong space lower bounds for achieving such a data structure when the input dataset arrives in a data stream.

Let us firstly describe the offline version:

Definition 4.1 (Exponential sampler). *Given matrix $A \in \mathbb{R}^{n \times d}$ and $x \in \mathbb{R}^d$, the goal is to sample index $i \sim [n]$ with probability $p_i = \langle \exp(Ax), \mathbf{1}_n \rangle^{-1} \cdot \exp(Ax)_i$, where $\mathbf{1}_n$ denotes a length-n vector, $\exp(Ax) \in \mathbb{R}^n$ denotes a length-n vector with $\exp(Ax)_i = \exp((Ax)_i)$, and $\exp(z)$ is the usual exponential function.*

Note that at the end of the stream, we only need to sample one index $i \in [n]$. On the other hand, there are three possibilities for streaming version:

- Both A and x arrive in streaming fashion
- A is fixed but x arrives in streaming fashion
- x is fixed but A arrives in streaming fashion

We consider each of these cases separately. Regardless, we use the following definition for each of the various cases:

Definition 4.2. *Let $C > 0$ be any fixed constant and let $C_0 \in [n^{-C}, n^C]$. Let y be a vector. Then the exponential sampler outputs an index j^* such that for all $i \in [n]$,*

$$\Pr[j^* = i] = C_0 \cdot \frac{\exp(y_i)}{\langle \exp(y), \mathbf{1}_n \rangle}.$$

We first recall the (two-party) set-disjointness communication problem $\mathsf{SetDisj}_n$, in which two parties Alice and Bob have subsets A and B, respectively, of $[n]$. Note that we can equivalently view A and B as binary vectors in n-dimensional space, serving as the indicator vector for whether each index $i \in [n]$ is in the player's input subset. The task for the players is to determine whether there exists a common element in their intersection, i.e., whether there exists $i \in [n]$ such that $i \in (A \cap B)$ or equivalently, $A_i = B_i = 1$. In fact, the problem promises that either the inputs are completely disjoint, $|A \cap B| = 0$ or the inputs contain only a single coordinate in their intersection, $|A \cap B| = 1$. We recall the following standard communication complexity result of set-disjointness.

Theorem 4.3 (Kalyanasundaram & Schnitger (1992); Razborov (1992); Bar-Yossef et al. (2004)). *Any protocol that solves the set-disjointness problem $\mathsf{SetDisj}_n$ with probability at least $\frac{3}{4}$ requires $\Omega(n)$ bits of total communication.*

We show that even a sampler that relaxes the probability distribution defined in Definition 4.2 up to a factor of n^C is infeasible in the streaming model.

Theorem 4.4. *Let $y \in \mathbb{R}^n$ that arrives as a data stream and let $C > 0$ be a constant. Then any algorithm that samples an index $i \in [n]$ with probability proportional to $p_i = \frac{\exp(y_j)}{\langle \exp(y), \mathbf{1}_n \rangle}$ must use $\Omega(n)$ bits of space, even if the sampling probabilities are allowed to be distorted by as large as n^C and even if $\|y\|_\infty = O(\log n)$.*

Proof. Let $A, B \in \{0,1\}^n$ be input vectors from the set disjointness problem, so that the goal is to determine whether there exists $i \in [n]$ such that $A_i = B_i = 0$. Observe that Alice and Bob can multiply A and B by $100C \log n$ for some constant $C > 0$. Now, note that in the disjoint case, we have that $\|A+B\|_\infty = 100C \log n$ and in the non-disjoint case, we have that $\|A+B\|_\infty = 200C \log n$. In particular, in the non-disjoint case, there exists $i \in [n]$ such that $A_i + B_i = 200C \log n$ and for all $j \neq i$, we have that $A_j + B_j \leq 100C \log n$. Hence, in the non-disjoint case, any exponential sampler will output i with probability proportional to $\exp(200C \log n)$ and output $j \neq i$ with probability proportional to $n \cdot \exp(100C \log n)$. Even if the sampling probabilities are distorted by a factor of n^C, any exponential sampler would output i with probability at least $\frac{3}{4}$.

Thus, Alice and Bob can use such a data structure to sample an index i and then check whether $A_i = B_i = 1$. In particular, Alice can first create a data stream encoding the vector A, run the sampling algorithm on the data stream, and then pass the state of the algorithm to Bob. Bob can then create another portion of the data stream encoding an addition of the vector B, take the state of the algorithm from Alice, run the sampling algorithm on the portion of the data stream, and query the algorithm for an index i. Bob can then take the index and pass it to Alice, and the two parties can finally communicate whether $A_i = B_i = 1$, thereby solving set-disjointness with probability at least $\frac{3}{4}$. Note that the communication of the protocol is the space used by the sampling algorithm. Therefore by Theorem 4.3, such a sampler must use $\Omega(n)$ bits of space. □

5 ℓ_2 Sampler Upper bound with A and x

In this section, we describe a standard data structure for ℓ_2 sampling. We start with providing the definition of ℓ_2 sampler as follows,

Definition 5.1. *Let n denote a positive integer. Let $\epsilon \geq 0$ denote a parameter. In ℓ_2 sampling, we receives y each coordinates online, it can be positive/negative, the goal is to at the end of the stream output an index $I \in [n]$ such that for each $j \in [n]$*

$$\Pr[I = j] = (1 \pm \epsilon) \cdot \frac{|y_j|^2}{\|y\|_2^2} + 1/\operatorname{poly}(n).$$

We describe various instantiations of the ℓ_2 sampler for sampling entries from a vector $Ax \in \mathbb{R}^n$, based upon whether the matrix $A \in \ltimes \times$ is updated during the data stream, whether the vector $x \in \mathbb{R}^d$ is updated during the data stream, or both.

5.1 A is updated during the streaming and x is fixed

In this section, we describe the construction of an ℓ_2 sampler for sampling coordinates of the vector $Ax \in \mathbb{R}^n$, in the setting where the vector $x \in \mathbb{R}^d$ is fixed, but the entries of $A \in \mathbb{R}^{n \times d}$ are evolving as the data stream progresses.

Definition 5.2 (Updating A and fixed x)**.** *In this setting, we assume $x \in \mathbb{R}^d$ is fixed, we receive updates to the entries of $A \in \mathbb{R}^{n \times d}$ in a turnstile data stream. Then for $y = Ax$, we want a data structure that produces the ℓ_2 sampling guarantee for y.*

We remark that a turnstile data stream means that each update of the data stream can increase or decrease a single entry of A.

In this work, we are interested in the regime of $n \gg d$. Then we have the following guarantee:

Theorem 5.3. *Suppose $y = Ax$, for $x \in \mathbb{R}^n$, which is fixed, and $A \in \mathbb{R}^{n \times d}$, which is defined by a turnstile stream. There exists an algorithm that uses $d \log n + \operatorname{poly}\left(\frac{1}{\epsilon}, \log n\right)$ bits of space and returns $I \in [n]$ such that $\Pr[I = j] = (1 \pm \epsilon) \cdot \frac{|y_j|^2}{\|y\|_2^2} + 1/\operatorname{poly}(n)$. The update time of the data structure is $d \operatorname{poly}\left(\frac{1}{\epsilon}, \log n\right)$.*

Proof. Recall that existing approximate ℓ_2 samplers, e.g., Algorithm 2 maintains a linear sketch Φy, where $\Phi \in \mathbb{R}^{m \times n}$, for $m = \operatorname{poly}\left(\frac{1}{\epsilon}, \log n\right)$. We have $y = Ax$, where $x \in \mathbb{R}^d$ is fixed but $A \in \mathbb{R}^{n \times d}$ is defined through turnstile updates. Nevertheless, we can maintain the state of $\Phi A x$. In particular, whenever we receive an update in $A_{i,j}$ by Δ, then we can compute $\Phi e_i e_j^\top \Delta x$ to update the sketch

ΦAx. To analyze the space complexity, observe that storing ΦAx requires $O(m)$ words of space and x requires d words of space, which is $d \log n + \mathrm{poly}\left(\frac{1}{\epsilon}, \log n\right)$ bits of space in total. Moreover, each update to $A_{i,j}$ can change all entries of ΦAx, so the update time is $O(md) = d\ \mathrm{poly}\left(\frac{1}{\epsilon}, \log n\right)$. □

5.2 x IS UPDATED DURING THE STREAMING AND A IS FIXED

We next consider the setting where the vector $x \in \mathbb{R}^d$ is updated as the data stream progress, but the entries of $A \in \mathbb{R}^{n\times d}$ are fixed.

Definition 5.4 (Fixed A and updating x)**.** *We assume $A \in \mathbb{R}^{n\times d}$ is fixed, we receive updates to $x \in \mathbb{R}^d$ in a turnstile data stream. Then for $y = Ax$, we want a data structure that produces the ℓ_2 sampling guarantee for y.*

We have the following algorithmic guarantees for this setting:

Theorem 5.5. *Suppose $y = Ax$, for $A \in \mathbb{R}^{n\times d}$, which is fixed, and $x \in \mathbb{R}^n$, which is defined by a turnstile stream. There exists an algorithm that uses $d\ \mathrm{poly}\left(\frac{1}{\epsilon}, \log n\right)$ bits of space and returns $I \in [n]$ such that $\Pr[I = j] = (1 \pm \epsilon) \cdot \frac{|y_j|^2}{\|y\|_2^2} + 1/\mathrm{poly}(n)$. The update time of the data structure is $O(1)$.*

Proof. Again recall that existing approximate ℓ_2 samplers, e.g., Algorithm 2 maintains a linear sketch Φy, where $\Phi \in \mathbb{R}^{m\times n}$, for $m = \mathrm{poly}\left(\frac{1}{\epsilon}, \log n\right)$. Since $y = Ax$, but $A \in \mathbb{R}^{n\times d}$ is too large to store, while $x \in \mathbb{R}^n$ is defined through turnstile updates, we can instead maintain the sketch ΦA and the vector x and compute $\Phi Ax = \Phi y$ after the stream concludes. Note that storing ΦA requires $O(md)$ words of space and x requires d words of space, which is $d\ \mathrm{poly}\left(\frac{1}{\epsilon}, \log n\right)$ bits of space in total. Moreover, each update to x changes a single entry, so the update time is $O(1)$. □

5.3 BOTH A AND x ARE UPDATED DURING THE STREAMING

Finally, we consider the setting where both the vector $x \in \mathbb{R}^d$ and the entries of $A \in \mathbb{R}^{n\times d}$ can be changed by updates from the data stream.

Definition 5.6 (Updating A and updating x)**.** *In this setting, we receive updates to both $A \in \mathbb{R}^{n\times d}$ and $x \in \mathbb{R}^d$ in a turnstile data stream. Then for $y = Ax$, we want a data structure that provides the ℓ_2 sampling guarantee for y.*

We have the following guarantees:

Lemma 5.7 (Upper Bound)**.** *Suppose $y = Ax$, for $A \in \mathbb{R}^{n\times d}$ and $x \in \mathbb{R}^n$, which are each defined in a stream through turnstile updates. There exists an algorithm that uses $d\ \mathrm{poly}\left(\frac{1}{\epsilon}, \log n\right)$ bits of space and returns $I \in [n]$ such that $\Pr[I = j] = (1 \pm \epsilon) \cdot \frac{|y_j|^2}{\|y\|_2^2} + 1/\mathrm{poly}(n)$. The update time is $\mathrm{poly}\left(\frac{1}{\epsilon}, \log n\right)$.*

Proof. As before, recall that existing approximate ℓ_2 samplers, e.g., Algorithm 2 maintains a linear sketch Φy, where $\Phi \in \mathbb{R}^{m\times n}$, for $m = \mathrm{poly}\left(\frac{1}{\epsilon}, \log n\right)$. Since $y = Ax$, but now both $A \in \mathbb{R}^{n\times d}$ and $x \in \mathbb{R}^n$ are defined through turnstile updates, we can instead maintain the sketch ΦA and the vector x and compute $\Phi Ax = \Phi y$ after the stream concludes. Observe that maintaining ΦA requires $O(md)$ words of space and x requires d words of space, which is $d\ \mathrm{poly}\left(\frac{1}{\epsilon}, \log n\right)$ bits of space in total. Each update to A can change all m entries of in a single column of ΦA, while each update to x changes a single entry. Hence, the update time is $\mathrm{poly}\left(\frac{1}{\epsilon}, \log n\right)$. □

6 ℓ_2 SAMPLER LOWER BOUND (WITH A AND x)

In this section, we give lower bounds for ℓ_2 sampling from a vector $y = A^{\otimes p}x$, as either A or x are updated in a data stream. We show that in any of these cases, the general problem is substantially more difficult than the previous case where $p = 1$.

We first recall the Index problem for one-way communication. In the INDEX_n problem, Alice receives a vector $v \in \{0,1\}^n$ and Bob receives a coordinate $i \in [n]$. The goal is for Bob to compute v_i with probability at least $\frac{3}{4}$, given some message Π from Alice. We recall the following communication complexity lower bounds for Index.

Theorem 6.1 (Kremer et al. (1999))**.** *Any protocol that solves INDEX_n with probability at least $\frac{3}{4}$ requires $\Omega(n)$ bits of communication.*

Lemma 6.2 (Lower Bound)**.** *Any streaming algorithm that solves problem defined as Definition 5.6 will require $\Omega(d)$ space.*

Proof. Suppose Alice receives a vector $v \in \{0,1\}^d$. Then Alice creates the diagonal matrix $M \in \{0,1\}^{d \times d}$ so that the j-th diagonal entry of A is v_j, for all $j \in [n]$. Finally, Alice creates $A \in \mathbb{R}^{(d+1)\times d}$ by appending the row consisting of $\frac{1}{10^{10}}$ in all of its d entries to M. Suppose Bob receives the coordinate $i \in [d]$ and wants to determine v_i. Then Bob can set x to be the elementary vector $e_i \in \mathbb{R}^d$, which has a 1 in its i-th coordinate and zeros elsewhere. Observe that by construction, Ax is the i-th column of A. If $v_i = 1$, then the i-th column of A consists of a 1 in the i-th entry, $\frac{1}{10^{10}}$ in the $(d+1)$-st entry, and zeros elsewhere. Hence, a sampler with the desired properties will output i with probability at least $\frac{3}{4}$. Similarly, if $v_i = 0$, then the i-th column of A consists of $\frac{1}{10^{10}}$ in the $(d+1)$-st entry and zeros elsewhere. Thus, the sampler with the desired properties will output $d+1$ with probability 1. Bob can therefore distinguish between these two cases with probability at least $\frac{3}{4}$, thereby solving INDEX_d with probability at least $\frac{3}{4}$. Therefore, by Theorem 6.1, such a sampler must use at least $\Omega(d)$ space. $\square$

In fact, we show that if $y = A^{\otimes p}x$, where $A \in \mathbb{R}^{n\times n}$ so that $A^{\otimes p} \in \mathbb{R}^{n^p \times n^p}$ denotes the p-wise self-tensor and $x \in \mathbb{R}^{n^p}$, then actually L_2 sampling from y uses $\Omega(n)$ bits of space.

Lemma 6.3. *Let $A \in \mathbb{R}^{n\times n}$ and $A^{\otimes p} \in \mathbb{R}^{n^p \times n^p}$ denote the p-wise self-tensor. Let $y = A^{\otimes p}x$, so that $x \in \mathbb{R}^{n^p}$. Then even if all the entries of x arrive in a data stream followed by all the entries of A, L_2 sampling from y requires $\Omega(n)$ bits of space.*

Proof. Let $S \in \{0,1\}^n$ be an instance of INDEX_n. Suppose Alice creates the diagonal matrix A with exactly S being the entries across its diagonal, i.e., $A_{1,1} = S_1, \ldots, A_{n,n} = S_n$. Bob has an index $i \in [n]$, and sets the vector x to be the elementary vector $\mathbf{e}_j$, where $j = i \cdot n^{p-1}$. Then by construction Ax is the all zeros vector if $S_i = 0$ and otherwise there is a nonzero entry, which allows Alice and Bob to solve INDEX_n. Hence, L_2 sampling from y requires $\Omega(n)$ bits of space. $\square$

7 The Tensor Version Problem

In this section, we further consider sampling from a tensor product. We provide the tensor notations and objects.

Definition 7.1. *Let $A_1 \in \mathbb{R}^{n\times d}$, let $A_2 \in \mathbb{R}^{n\times d}$, we define*

$$\mathsf{A} = A_1 \otimes A_2 \in \mathbb{R}^{n^2 \times d^2}.$$

Let $x \in \mathbb{R}^{d^2}$. Let $\mathsf{A}_i \in \mathbb{R}^{n\times d^2}$ denote the i-th block of A.

Definition 7.2 (fixed x, Streaming Sampler for one of A_1 and A_2 is updating.)**.** *One way, we assume $x \in \mathbb{R}^{d^2}$ is fixed. We assume that*

- *one of A_1 and A_2 is updating*
- *one of A_1 and A_2 is fixed*

Let $y = \mathsf{A}x$, we want ℓ_2 sampling guarantee for sampling one coordinate in $y_i \in \mathbb{R}^{n^2}$ for all $i \in [n^2]$.

We use the following formulation of Nisan's pseudorandom generator to derandomize our algorithm.

Theorem 7.3 (Nisan's PRG, Nisan (1992))**.** *Suppose $\mathcal{A}$ is an algorithm that requires $S = \Omega(\log n)$ bits of space and R random bits. Then there exists a pseudorandom generator for $\mathcal{A}$ that succeeds with probability $1 - \frac{1}{\mathrm{poly}(n)}$ and uses $O(S \log R)$ bits of space.*

Algorithm 1 We build on algorithm based on $S(x_1 \otimes x_2)$

```
procedure MAIN(x_1, x_2 ∈ ℝ^n)
    Suppose we use O(nd) space to store A_1 and A_2 (Avoid n^2 time/space)
    Suppose we receive an update q ∈ [2], i ∈ [n], j ∈ [d], Δ
    Suppose you have hash function g to access uniform number
    if q = 1 then
        p ← g(i(n−1)+1, ···, in)                                  ▷ p ∈ ℝ^n
        y ← y + ΦΔ(e_{[i(n−1)+1,in]} ∘ (A_2)_{*,j})/p             ▷ Φ_1 is decided by h_1, σ_1
    else
        y_2 ← y_2 + Φ_2 e_i Δ                                     ▷ Φ_2 is decided by h_2, σ_2
    end if

end procedure
```

In the following Lemma, we state a streaming algorithm to solve tensor related sampling problem. We consider the situation that one of A_1 and A_2 is fixed, and the other one is updated in streaming fashion. We show the following estimation guarantees using the standard CountSketch analysis, c.f., Charikar et al. (2004); Jowhari et al. (2011).

Lemma 7.4 (Tensor ℓ_2 Tail Estimation)**.** *Let $y = (A_1 \otimes A_2)x \in \mathbb{R}^{n^2}$. Let only one of A_1 and A_2 be updated in streaming. Let $w = \frac{y_i}{\sqrt{u_i}}$ for a constant $u_i \in [0,1]$ generated uniformly at random. There is an algorithm $\mathcal{A}$ that that uses $O(nd) + \mathrm{poly}\left(\frac{1}{\epsilon}, \log n\right)$ space, uses $O(n)$ update time, and estimates each element of w up to additive error $\epsilon \cdot \|z\|_2$, where z denotes the tail vector of w without the largest $\frac{1}{\epsilon^2}$ entries in magnitude. Specifically, for all $i \in [n^2]$, we have $|\widehat{w}_i - w_i| \le \epsilon \cdot \|z\|_2$.*

Proof. Consider hash function $h_1, h_2 : [n] \to [b]$. Consider random sign functions $\sigma_1, \sigma_2 : [n] \to \{-1, +1\}$. We consider a fixed index $i_1, i_2 \in [n]$. Let $j = h_1(i_1) + h_2(i_2) \pmod b$. Let $h^{-1}(j)$ denote the all the pairs $(i_1, i_2) \in [n] \times [n]$ such that $h_1(i_1) + h_2(i_2) \pmod b = j$. Note that $\widehat{y}_i$ induced by h is

$$\widehat{w}_i = w_i + \sum_{l \in h^{-1}(j) \setminus \{i\}} s_i s_l w_{l_1} w_{l_2},$$

For ease of presentation, we write $\sigma_i = \sigma_{1,i_1}\sigma_{2,i_2}$ and $\sigma_l = \sigma_{1,l_1}\sigma_{2,l_2}$.

$$\begin{aligned}
\mathbb{E}[\widehat{w}_i] &= \mathbb{E}\Big[w_i + \sum_{l \in h^{-1}(j) \setminus \{i\}} \sigma(i)\sigma(l) w_l\Big] \\
&= \mathbb{E}[w_i] + \sum_{l \in h^{-1}(j) \setminus \{i\}} \mathbb{E}[\sigma(i) \cdot \sigma(l)] \cdot w_l \\
&= w_i + \sum_{l \in h^{-1}(j) \setminus \{i\}} \mathbb{E}[\sigma(i)] \cdot \mathbb{E}[\sigma(l)] \cdot w_l = w_i,
\end{aligned}$$

where the first step follows from definition, the second step follows from linearity of expectation, the third step follows from $\sigma(i)$ and $\sigma(l)$ are independent, the forth step follows from $\mathbb{E}[\sigma(l)] = 0$.

We now upper bound the variance of $\widehat{w}_i - y_i$ by analyzing $\mathbb{E}[(\widehat{y}_i)^2]$. Let $\mathcal{H}$ be the set of the top $\frac{1}{\epsilon^2}$ items and let $\mathcal{E}$ be the event that none of the items in $\mathcal{H}$ are mapped to $h(i)$, i.e., $h(a) \neq h(i)$ for all $a \in \mathcal{H}$.

Observe that for $b = \frac{100}{\epsilon^2}$, we have that $\Pr[\mathcal{E}] \ge 0.9$. Then we have:

$$\begin{aligned}
\mathbb{E}[(\widehat{w}_i - w_i)^2 \mid \mathcal{E}] &= \mathbb{E}[(\sum_{l \in [n]^2 \setminus \mathcal{H}, l \in h^{-1}(j)} \sigma(i)\sigma(l) w_l)^2] \\
&= \mathbb{E}\Big[\sum_{l \in [n]^2 \setminus \mathcal{H}, l \in h^{-1}(j)} w_l^2\Big]
\end{aligned}$$

$$= \frac{1}{b} \cdot \sum_{l \in [n]^2 \setminus \mathcal{H}, l \in h^{-1}(j)} w_l^2$$
$$\leq \frac{1}{b} \cdot (w_1^2 + \ldots + w_{n^2}^2 - \sum_{l \in \mathcal{H}} w_l^2)$$
$$= 100\epsilon^2 \cdot \|z\|_2^2,$$

for $b = \frac{100}{\epsilon^2}$, since z is the vector corresponding to y that removes the entries in $\mathcal{H}$. By Chebyshev's inequality, we have that

$$\Pr[|\widehat{w}_i - w_i| \geq \epsilon \cdot \|z\|_2 \mid \mathcal{E}] \leq \frac{1}{10}.$$

Since $\Pr[\mathcal{E}] \geq 0.9$, then

$$\Pr|\widehat{w}_i - w_i| \geq \epsilon \cdot \|z\|_2 \leq 0.2,$$

for a fixed hash function h. By taking the median of $O(\log n)$ estimations corresponding to $O(\log n)$ different hash functions h, we have that

$$\Pr[|\widehat{w}_i - w_i| \geq \epsilon \cdot \|z\|_2] \leq \frac{1}{n^{10}}.$$

Thus by a union bound over $i \in [n] \times [n]$, we have that with probability at least $1 - \frac{1}{n^5}$, we have for all $i \in [n]$, $|\widehat{w}_i - w_i| \geq \epsilon \cdot \|z\|_2$. $\square$

We state the following lemma as a structural property that will allow us to achieve our tenor product sampler. We remark that the proof is a simple adaptation of existing proofs for approximate ℓ_p sampling (Jowhari et al., 2011). Thus we defer the proof to Appendix B.

Lemma 7.5. *Let $y = (A_1 \otimes A_2)x \in \mathbb{R}^{n^2}$ and let $w \in \mathbb{R}^{n^2}$ so that $w_i = \frac{y_i}{\sqrt{u_i}}$ for a constant $u_i \in [0, 1]$ generated uniformly at random. Let z denote the tail vector of w without the largest $\frac{1}{\epsilon^2}$ entries in magnitude. Let $\widehat{Z}$ be a 2-approximation to $\|z\|_2$ and $\widehat{Y}$ be a 2-approximation to $\|y\|_2$. Then*

$$\Pr\left[\widehat{Z} > \sqrt{\frac{C \log n}{\epsilon}} \cdot \widehat{Y}\right] \leq O(\epsilon) + \frac{1}{\text{poly}(n)}.$$

Finally, we describe the guarantees of our tensor-based sampler, deferring the proof to Appendix C.

Theorem 7.6. *Let $y = (A_1 \otimes A_2)x \in \mathbb{R}^{n^2}$ and let $w \in \mathbb{R}^{n^2}$ so that for each $i \in [n^2]$, $w_i = \frac{y_i}{\sqrt{u_i}}$ for a constant $u_i \in [0, 1]$ generated uniformly at random. Let z denote the tail vector of w without the largest $\frac{1}{\epsilon^2}$ entries in magnitude. Suppose there exists:*

1. *An algorithm $\mathcal{A}_1$ that provides a 2-approximation to $\|y\|_2$ with probability $1 - \frac{1}{n^2}$.*
2. *An algorithm $\mathcal{A}_2$ that provides a 2-approximation to $\|z\|_2$ with probability $1 - \frac{1}{n^2}$.*
3. *An algorithm $\mathcal{A}_3$ that estimates each element of w up to additive error $\epsilon \cdot \|z\|_2$, $|\widehat{w_i} - w_i| \leq \epsilon \cdot \|z\|_2$, for all $i \in [n^2]$.*

Then there exists a data structure that uses $\text{poly}\left(\frac{1}{\epsilon}, \log n\right)$ bits of space and outputs each index i with probability p_i, such that

$$(1 - \epsilon) \cdot \frac{y_i^2}{\|y\|_2^2} - \frac{1}{\text{poly}(n)} \leq p_i \leq (1 + \epsilon) \cdot \frac{y_i^2}{\|y\|_2^2} + \frac{1}{\text{poly}(n)}.$$

We remark that the algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ in the context of Theorem 7.6 can be achieved using the standard AMS ℓ_2 norm estimator (Alon et al., 1999). Moreover, algorithm $\mathcal{A}_3$ in the context of Theorem 7.6 can be achieved using the standard CountSketch algorithm (Charikar et al., 2004).

8 Conclusion

Our research introduces a transformative approach to enhancing the efficiency of attention-based deep learning models, crucial in fields like natural language processing and computer vision. By developing an innovative sampling framework, we've effectively reduced the computational demands while preserving or enhancing model performance. This balance is a significant advancement, particularly for deploying complex models in resource-limited environments.

Our methods not only lower computational needs but also maintain or improve model accuracy and robustness. These results highlight the practicality and adaptability of our approach across different architectures and data types. Additionally, our framework's scalability ensures its relevance in the face of ever-growing model and dataset sizes.

In summary, our contribution addresses a key challenge in attention-based AI models, opening the door to more efficient, scalable, and sustainable AI technologies. This work lays the groundwork for future advancements in the field, catering to the increasing computational demands of modern AI applications.

References

Thomas D Ahle, Michael Kapralov, Jakob BT Knudsen, Rasmus Pagh, Ameya Velingker, David P Woodruff, and Amir Zandieh. Oblivious sketching of high-degree polynomial kernels. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 141–160. SIAM, 2020.

Elizabeth S. Allman, Catherine Matias, and John A. Rhodes. Identifiability of parameters in latent structure models with many observed variables. *The Annals of Statistics*, 37(6A), dec 2009a. doi: 10.1214/09-aos689. URL https://doi.org/10.1214%2F09-aos689.

Elizabeth S. Allman, Sonja Petrović, John A. Rhodes, and Seth Sullivant. Identifiability of 2-tree mixtures for group-based models, 2009b. URL https://arxiv.org/abs/0909.1854.

Josh Alman and Zhao Song. Fast attention requires bounded entries. In *NeurIPS*. arXiv preprint arXiv:2302.13214, 2023.

Josh Alman and Zhao Song. How to capture higher-order correlations? generalizing matrix softmax attention to kronecker computation. In *ICLR*. arXiv preprint arXiv:2310.04064, 2024.

Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*, pp. 363–372, 2011.

Pranjal Awasthi and Anupam Gupta. Improving length-generalization in transformers via task hinting. *arXiv preprint arXiv:2310.00726*, 2023.

Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.

Srinadh Bhojanapalli and Sujay Sanghavi. A new sampling technique for tensors. In *arXiv preprint*. https://arxiv.org/pdf/1502.05023, 2015.

Jan van den Brand, Zhao Song, and Tianyi Zhou. Algorithm and hardness for dynamic attention maintenance in large language models. *arXiv preprint arXiv:2304.02207*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems (NeurIPS)*, 34: 17413–17426, 2021.

Beidi Chen, Tri Dao, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Re. Pixelated butterfly: Simple and efficient sparse training for neural network models. In *International Conference on Learning Representations (ICLR)*, 2022.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Joon Hee Choi and S. Vishwanathan. Dfacto: Distributed factorization of tensors. In *NIPS*, pp. 1296–1304, 2014.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Edith Cohen and Ofir Geri. Sampling sketches for concave sublinear functions of frequencies. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, pp. 1361–1371, 2019.

Michael B. Cohen. Nearly tight oblivious subspace embeddings by trace inequalities. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Arlington, VA, USA, January 10-12, 2016*, pp. 278–287, 2016.

Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*. https://arxiv.org/pdf/1810.07896.pdf, 2019.

Graham Cormode and Hossein Jowhari. l_p samplers and their applications: A survey. *ACM Comput. Surv.*, 52(1):16:1–16:31, 2019.

Graham Cormode, S. Muthukrishnan, and Irina Rozenbaum. Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 25–36. ACM, 2005.

Giannis Daras, Nikita Kitaev, Augustus Odena, and Alexandros G Dimakis. Smyrf-efficient attention using asymmetric clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 6476–6489, 2020.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

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

Yichuan Deng, Zhao Song, Omri Weinstein, and Ruizhe Zhang. Fast distance oracles for any symmetric norm. *arXiv preprint arXiv:2205.14816*, 2022b.

Yichuan Deng, Yeqi Gao, and Zhao Song. Solving tensor low cycle rank approximation. *arXiv preprint arXiv:2304.06594*, 2023a.

Yichuan Deng, Zhihang Li, Sridhar Mahadevan, and Zhao Song. Zero-th order algorithm for softmax attention optimization. *arXiv preprint arXiv:2307.08352*, 2023b.

Yichuan Deng, Zhihang Li, and Zhao Song. Attention scheme inspired softmax regression. *arXiv preprint arXiv:2304.10411*, 2023c.

Yichuan Deng, Sridhar Mahadevan, and Zhao Song. Randomized and deterministic attention sparsification algorithms for over-parameterized feature dimension. *arxiv preprint: arxiv 2304.03426*, 2023d.

Yichuan Deng, Zhao Song, and Shenghao Xie. Convergence of two-layer regression with nonlinear units. *arXiv preprint arXiv:2308.08358*, 2023e.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Huaian Diao, Zhao Song, Wen Sun, and David P. Woodruff. Sketching for kronecker product regression and p-splines. In *AISTATS*. https://arxiv.org/pdf/1712.09473, 2018.

Huaian Diao, Rajesh Jayaram, Zhao Song, Wen Sun, and David Woodruff. Optimal sketching for kronecker product regression and low rank approximation. *Advances in neural information processing systems*, 32, 2019.

Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms. *arXiv preprint arXiv:2110.10090*, 2021.

Yeqi Gao, Lianke Qin, Zhao Song, and Yitan Wang. A sublinear adversarial training algorithm. *arXiv preprint arXiv:2208.05395*, 2022.

Yeqi Gao, Sridhar Mahadevan, and Zhao Song. An over-parameterized exponential regression. *arXiv preprint arXiv:2303.16504*, 2023a.

Yeqi Gao, Zhao Song, and Junze Yin. An iterative algorithm for rescaled hyperbolic functions regression. *arXiv preprint arXiv:2305.00660*, 2023b.

Yuzhou Gu and Zhao Song. A faster small treewidth sdp solver. *arXiv preprint arXiv:2211.06033*, 2022.

Yuzhou Gu, Zhao Song, Junze Yin, and Lichen Zhang. Low rank matrix completion via robust alternating minimization in nearly linear time. *arXiv preprint arXiv:2302.11068*, 2023.

Insu Han, Rajesh Jarayam, Amin Karbasi, Vahab Mirrokni, David P. Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time. *arXiv preprint arXiv:2310.05869*, 2023.

Richard A Harshman. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. 1970.

Furong Huang, Niranjan U. N, Mohammad Umar Hakeem, Prateek Verma, and Animashree Anandkumar. Fast detection of overlapping communities via online tensor methods on gpus. *CoRR*, abs/1309.0787, 2013.

Adobe Inc. Adobe firefly. https://www.adobe.com/sensei/generative-ai/firefly.html, 2023.

Rajesh Jayaram and David P. Woodruff. Perfect l_p sampling in a data stream. *SIAM J. Comput.*, 50 (2):382–439, 2021.

Rajesh Jayaram, David P. Woodruff, and Samson Zhou. Truly perfect samplers for data streams and sliding windows. In *PODS '22: International Conference on Management of Data*, pp. 29–40, 2022.

Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. In *STOC*. arXiv preprint arXiv:2004.07470, 2021.

Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pp. 49–58, 2011.

Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. Polysketchformer: Fast transformers via sketches for polynomial kernels. *arXiv preprint arXiv:2310.01655*, 2023.

Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discret. Math.*, 5(4):545–557, 1992.

U. Kang, Evangelos E. Papalexakis, Abhay Harpale, and Christos Faloutsos. Gigatensor: scaling tensor analysis up by 100 times - algorithms and discoveries. In *KDD*, pp. 316–324, 2012.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv: 2001.04451*, 2020.

Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Comput. Complex.*, 8(1):21–49, 1999.

Joseph B. Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and its Applications*, 18(2):95–138, 1977. ISSN 0024-3795. doi: https://doi.org/10.1016/0024-3795(77)90069-6. URL https://www.sciencedirect.com/science/article/pii/0024379577900696.

Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *COLT*. https://arxiv.org/pdf/1905.04447.pdf, 2019.

Zhihang Li, Zhao Song, and Tianyi Zhou. Solving regularized exp, cosh and sinh regression problems. *arXiv preprint arXiv:2303.15725*, 2023.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Sepideh Mahabadi, Ilya P. Razenshteyn, David P. Woodruff, and Samson Zhou. Non-adaptive adaptive sampling on turnstile streams. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pp. 1251–1264, 2020.

Sepideh Mahabadi, David P. Woodruff, and Samson Zhou. Adaptive sketches for robust regression with importance sampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pp. 31:1–31:21, 2022.

Arvind V Mahankali, David P Woodruff, and Ziyu Zhang. Near-linear time and fixed-parameter tractable algorithms for tensor decompositions. *arXiv preprint arXiv:2207.07417*, 2022.

James Manyika. An overview of bard: an early experiment with generative ai. Technical report, Tech. rep., Technical report, Google AI, 2023.

Gary Marcus, Ernest Davis, and Scott Aaronson. A very preliminary analysis of dall-e 2. *arXiv preprint arXiv:2204.13807*, 2022.

Morteza Monemizadeh and David P. Woodruff. 1-pass relative-error l_p-sampling with applications. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pp. 1143–1160. SIAM, 2010.

Jelani Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 ieee 54th annual symposium on foundations of computer science*, pp. 117–126. IEEE, 2013.

Noam Nisan. Pseudorandom generators for space-bounded computation. *Comb.*, 12(4):449–461, 1992.

Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–17, 2013.

Anh-Huy Phan, Petr Tichavsky, and Andrzej Cichocki. Low complexity damped gauss–newton algorithms for candecomp/parafac. *SIAM Journal on Matrix Analysis and Applications*, 34(1):126–147, 2013.

Lianke Qin, Rajesh Jayaram, Elaine Shi, Zhao Song, Danyang Zhuo, and Shumo Chu. Adore: Differentially oblivious relational database operators. *VLDB*, 2022a.

Lianke Qin, Aravind Reddy, Zhao Song, Zhaozhuo Xu, and Danyang Zhuo. Adaptive and dynamic multi-resolution hashing for pairwise summations. *arXiv preprint arXiv:2212.11408*, 2022b.

Lianke Qin, Zhao Song, and Yitan Wang. Fast submodular function maximization. *arXiv preprint arXiv:2305.08367*, 2023a.

Lianke Qin, Zhao Song, Lichen Zhang, and Danyang Zhuo. An online and unified algorithm for projection matrix vector multiplication with application to empirical risk minimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 101–156. PMLR, 2023b.

Lianke Qin, Zhao Song, and Ruizhe Zhang. A general algorithm for solving rank-one matrix sensing. *arXiv preprint arXiv:2303.12298*, 2023c.

Alexander A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106 (2):385–390, 1992.

Aravind Reddy, Zhao Song, and Lichen Zhang. Dynamic tensor product regression. *arXiv preprint arXiv:2210.03961*, 2022.

John A Rhodes and Seth Sullivant. Identifiability of large phylogenetic mixture models. *Bulletin of mathematical biology*, 74:212–231, 2012.

Elina Robeva. Orthogonal decomposition of symmetric tensors. *SIAM Journal on Matrix Analysis and Applications*, 37(1):86–102, 2016.

Elina Robeva and Anna Seigal. Singular vectors of orthogonally decomposable tensors. *Linear and Multilinear Algebra*, 65(12):2457–2471, 2017.

Charlie Snell, Ruiqi Zhong, Dan Klein, and Jacob Steinhardt. Approximating how single head attention learns. *arXiv preprint arXiv:2103.07601*, 2021.

Zhao Song, David P. Woodruff, and Huan Zhang. Sublinear time orthogonal tensor decomposition. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NIPS) 2016, December 5-10, 2016, Barcelona, Spain*, pp. 793–801, 2016.

Zhao Song, David P Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *SODA*. arXiv preprint arXiv:1704.08246, 2019a.

Zhao Song, David P Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2772–2789. SIAM, 2019b.

Zhao Song, David Woodruff, Zheng Yu, and Lichen Zhang. Fast sketching of polynomial kernels of polynomial degree. In *International Conference on Machine Learning*, pp. 9812–9823. PMLR, 2021a.

Zhao Song, Lichen Zhang, and Ruizhe Zhang. Training multi-layer over-parametrized neural network in subquadratic time. *arXiv preprint arXiv:2112.07628*, 2021b.

Zhao Song, Zhaozhuo Xu, and Lichen Zhang. Speeding up sparsification using inner product search data structures. *arXiv preprint arXiv:2204.03209*, 2022a.

Zhao Song, Xin Yang, Yuanyuan Yang, and Lichen Zhang. Sketching meets differential privacy: Fast algorithm for dynamic kronecker projection maintenance. *arXiv preprint arXiv:2210.11542*, 2022b.

Zhao Song, Mingquan Ye, and Lichen Zhang. Streaming semidefinite programs: $o(\sqrt{n})$ passes, small space and fast runtime. *Manuscript*, 2023.

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

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

Charalampos E. Tsourakakis. MACH: fast randomized tensor decompositions. In *SDM*, pp. 689–700, 2010.

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.

James Vuckovic, Aristide Baratin, and Remi Tachet des Combes. A mathematical theory of attention. *arXiv preprint arXiv:2007.02876*, 2020.

Chi Wang, Xueqing Liu, Yanglei Song, and Jiawei Han. Scalable moment-based inference for latent dirichlet allocation. In *ECML-PKDD*, pp. 290–305, 2014.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Yining Wang, Hsiao-Yu Tung, Alexander J Smola, and Anima Anandkumar. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 991–999. https://arxiv.org/pdf/1506.04448, 2015.

Colin Wei, Yining Chen, and Tengyu Ma. Statistically meaningful approximation: a case study on approximating turing machines with transformers. *arXiv preprint arXiv:2107.13163*, 2021.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

Guanghao Ye. Fast algorithm for solving structured convex programs. *The University of Washington, Undergraduate Thesis*, 2020.

Amir Zandieh, Insu Han, Majid Daliri, and Amin Karbasi. Kdeformer: Accelerating transformers via kernel density estimation. In *ICML*. arXiv preprint arXiv:2302.02451, 2023.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

Lichen Zhang. Speeding up optimizations via data structures: Faster search, sample and maintenance. Master's thesis, Carnegie Mellon University, 2022.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Haoyu Zhao, Abhishek Panigrahi, Rong Ge, and Sanjeev Arora. Do transformers parse while predicting the masked word? *arXiv preprint arXiv:2303.08117*, 2023.

Roadmap. In Section A, we briefly discuss the background on ℓ_2 sampler. In Section B, we show that how to use the tail bound to obtain sampling result. In Section C, we present the tensor sampling result.

A ℓ_2 SAMPLER

We give the full details of the standard L_2 sampler from Jowhari et al. (2011); Mahabadi et al. (2020) in Algorithm 2. The proof of correctness is verbatim from Jowhari et al. (2011); Mahabadi et al. (2020). The challenge is how to implement the data structures of y, which is implicitly defined as $(A_1 \otimes A_2)x$. By comparison, in the standard setting of ℓ_2 samplers Monemizadeh & Woodruff (2010); Andoni et al. (2011); Jowhari et al. (2011); Jayaram & Woodruff (2021); Mahabadi et al. (2020), y is given as a data stream.

Algorithm 2 Standard ℓ_2 Sampler, e.g., extension of Jowhari et al. (2011) to $p = 2$

1: For each $i \in [n]$, let $u_i \in [0, 1]$ be chosen uniformly at random
2: $w_i \leftarrow \frac{y_i}{\sqrt{u_i}}$
3: Let z denote the tail vector of w without the largest $\frac{1}{\epsilon^2}$ entries in magnitude
4: Let $\widehat{Y}$ be a 2-approximation of $\|y\|_2$
5: Let $\widehat{Z}$ be a 2-approximation of $\|z\|_2$
6: $i \leftarrow \text{argmax}_{i\in[n]} |\widehat{w_i}|$
7: Let $C > 0$ be a large constant determined by the additive faliure probability $\frac{1}{\mathrm{poly}(n)}$
8: **if** $\widehat{Z} > \sqrt{\frac{C \log n}{\epsilon}} \cdot \widehat{Y}$ or $|w_i| < \sqrt{\frac{C \log n}{\epsilon}} \cdot \widehat{Y}$ **then**
9: Return FAIL
10: **else**
11: Return i with estimate $\sqrt{u_i} \cdot \widehat{w_i}$
12: **end if**

B FROM TAIL TO SAMPLING

Lemma B.1 (Restatement of Lemma 7.5). *Let $y = (A_1 \otimes A_2)x \in \mathbb{R}^{n^2}$ and let $w \in \mathbb{R}^{n^2}$ so that $w_i = \frac{y_i}{\sqrt{u_i}}$ for a constant $u_i \in [0, 1]$ generated uniformly at random. Let z denote the tail vector of w without the largest $\frac{1}{\epsilon^2}$ entries in magnitude. Let $\widehat{Z}$ be a 2-approximation to $\|z\|_2$ and $\widehat{Y}$ be a 2-approximation to $\|y\|_2$. Then*

$$\Pr\left[\widehat{Z} > \sqrt{\frac{C \log n}{\epsilon}} \cdot \widehat{Y}\right] \le O(\epsilon) + \frac{1}{\mathrm{poly}(n)}.$$

Proof. Let $\mathcal{E}_1$ denote the event that $\widehat{Z}$ is a 2-approximation to $\|z\|_2$ and $\widehat{Y}$ is a 2-approximation to $\|y\|_2$, so that

$$\Pr[\mathcal{E}_1] \ge 1 - \frac{1}{\mathrm{poly}(n)}.$$

Conditioned on $\mathcal{E}_1$, it suffices to bound the probability that

$$4\|z\|_2 > \sqrt{\frac{C \log n}{\epsilon}} \cdot \|y\|_2.$$

Let $j \in [n^2]$ be a fixed index and let u_j be fixed.

Let $T = \sqrt{\epsilon} \cdot \|y\|_2$ and for each $i \in [n^2]$, we define the indicator random variable $W_i = 1$ if $|w_i| > T$ and $W_i = 0$ otherwise, if $|w_i| \le T$. Note that W_i is an indicator random variable for whether the coordinate w_i in the vector w is “heavy” in magnitude.

We then define
$$Z_i = \frac{w_i^2}{T^2} \cdot (1 - W_i)$$
to be the scaled contribution of the small entries of z, and observe that $Z_i \in [0, 1]$.

Let
$$W = \sum_{i \in [n^2], i \neq j} w_i$$
denote the total number of heavy indices besides possibly index j and $Z = \sum_{i \in [n^2], i \neq j} Z_i$ denote the total scaled contribution of the light indices besides possibly index j. Let v denote the vector containing the heavy indices, so that $v_i = w_i$ for $W_i = 1$ and $v_i = 0$ otherwise for $W_i = 0$. Note that v has sparsity at most $Y + 1$ and moreover $U^2 Z = \|w - v\|_2^2$. We also have that $\|z\|_2 \le \|w - v\|_2$ unless $W \ge \frac{2}{\epsilon^2}$.

Let $\mathcal{E}_2$ denote the event that $W \ge \frac{2}{\epsilon^2}$ and let $\mathcal{E}_3$ denote the event that $Z \ge \frac{C \log n}{16 T^2 \epsilon} \cdot \|y\|_2^2$. Observe that if neither $\mathcal{E}_2$ nor $\mathcal{E}_3$ occur, then we have $4\|z\|_2 \le \sqrt{\frac{C \log n}{\epsilon}} \cdot \|y\|_2$, as desired. Thus it remains to bound the probability of the failure events $\mathcal{E}_2$ and $\mathcal{E}_3$.

We have $\mathbb{E}[W_i] = \frac{\|w\|_2^2}{T^2}$, so that $\mathbb{E}[W] \le \frac{1}{\epsilon}$. By Markov's inequality, we have that $\Pr[\mathcal{E}_2] \le \frac{\epsilon}{2}$.

We now upper bound $\Pr[\mathcal{E}_3]$. Recall that $Z_i = \frac{w_i^2}{T^2} \cdot (1 - W_i) = \frac{w_i^2}{T u_i^2} \cdot (1 - W_i)$, since $w_i = \frac{y_i}{\sqrt{u_i}}$. Observe that $Z_i > 0$ only if $|w_i| < T$, i.e., if $u_i \ge \frac{y_i^2}{\epsilon \cdot \|y\|_2^2}$, since $T = \sqrt{\epsilon} \cdot \|y\|_2$. For $\epsilon \in (0, 1)$, we thus have
$$\begin{aligned} \mathbb{E}[Z_i] &\le \int_{y_i^2 / \|y\|_2^2}^{1} z_i \mathrm{d}u_i \\ &= \int_{y_i^2 / \|y\|_2^2}^{1} \frac{y_i^2}{u_i} \frac{1}{T^2} \mathrm{d}u_i. \end{aligned}$$
Now, let $\mathcal{E}_4$ be the event that $u_i \ge \frac{1}{n^{C/2}}$ for all $i \in [n^2]$, so that $\Pr[\mathcal{E}_4] \ge 1 - \frac{1}{n^{C/2-2}}$.

Then
$$\begin{aligned} \mathbb{E}[Z_i \mid \mathcal{E}_4] &\le \frac{1}{1 - \frac{1}{n^{C/2-2}}} \int_{1/n^{C/2}}^{1} \frac{y_i^2}{u_i} \frac{1}{T^2} \mathrm{d}u_i \\ &\le \frac{C \log n}{T^2} y_i^2. \end{aligned}$$
Thus, we have
$$\begin{aligned} \mathbb{E}[Z \mid \mathcal{E}_4] &= \sum_{i \in [n^2]} \mathbb{E}[Z_i \mid \mathcal{E}_4] \\ &= \sum_{i \in [n^2]} \frac{C \log n}{T^2} y_i^2 \\ &\le \sum_{i \in [n^2]} \frac{C \log n}{\epsilon} \frac{y_i^2}{\|y\|_2^2} \\ &= \frac{C \log n}{\epsilon}. \end{aligned}$$
Thus by Markov's inequality, the probability that Z is larger than $\frac{C \log n}{16 T^2 \epsilon} \cdot \|y\|_2^2 = \frac{C \log n}{16 \epsilon^2}$ is at most $\frac{\epsilon}{16}$. The claim then follows from taking a union bound over the events $\mathcal{E}_1, \neg\mathcal{E}_2, \neg\mathcal{E}_3, \neg\mathcal{E}_4$. $\square$

C TENSOR SAMPLING

Theorem C.1 (Restatement of Theorem 7.6). *Let $y = (A_1 \otimes A_2)x \in \mathbb{R}^{n^2}$ and let $w \in \mathbb{R}^{n^2}$ so that for each $i \in [n^2]$, $w_i = \frac{y_i}{\sqrt{u_i}}$ for a constant $u_i \in [0, 1]$ generated uniformly at random. Let z denote the tail vector of w without the largest $\frac{1}{\epsilon^2}$ entries in magnitude. Suppose there exists:*

1. *An algorithm* $\mathcal{A}_1$ *that provides a* 2*-approximation to* $\|y\|_2$ *with probability* $1-\frac{1}{n^2}$.
2. *An algorithm* $\mathcal{A}_2$ *that provides a* 2*-approximation to* $\|z\|_2$ *with probability* $1-\frac{1}{n^2}$.
3. *An algorithm* $\mathcal{A}_3$ *that estimates each element of* w *up to additive error* $\epsilon\cdot\|z\|_2$,
$$|\widehat{w_i}-w_i|\le\epsilon\cdot\|z\|_2,$$
for all $i\in[n^2]$.

Then there exists a data structure that uses $\operatorname{poly}\left(\frac{1}{\epsilon},\log n\right)$ *bits of space and outputs each index* i *with probability* p_i*, such that*

$$(1-\epsilon)\cdot\frac{y_i^2}{\|y\|_2^2}-\frac{1}{\operatorname{poly}(n)}\le p_i\le(1+\epsilon)\cdot\frac{y_i^2}{\|y\|_2^2}+\frac{1}{\operatorname{poly}(n)}.$$

Proof. Let i be fixed and let $\mathcal{E}$ denote the event that $u_i<\frac{\epsilon}{C\log n}\frac{y_i^2}{\widehat{Y}^2}$, so that $|w_i|>\sqrt{\frac{C\log n}{\epsilon}}\cdot\widehat{Y}$.

Let $\mathcal{E}_1$ denote the event that $\widehat{Y}$ is a 2-approximation to $\|y\|_2$, $\widehat{Z}$ is a 2-approximation to $\|z\|_2$, and $|\widehat{w_i}-w_i|\le\epsilon\cdot\|z\|_2$ for all $i\in[n]$. Let $\mathcal{E}_2$ denote the event that $\widehat{Z}>\sqrt{\frac{C\log n}{\epsilon}}\cdot\widehat{Y}$ and let $\mathcal{E}_3$ denote the event that multiple indices j satisfy $|w_j|>\sqrt{\frac{C\log n}{\epsilon}}\cdot\widehat{Y}$. Finally, let $\mathcal{E}_4$ denote the event that $|\widehat{w_i}|<\sqrt{\frac{C\log n}{\epsilon}}\cdot\widehat{Y}$.

Intuitively, $\mathcal{E}_1$ is a good event, i.e., correctness of the data structures, which we would like to hold. On the other hand, $\mathcal{E}_2,\mathcal{E}_3,\mathcal{E}_4$ are bad events that distort the sampling probabilities, which we would like to avoid.

We first note that $\mathcal{E}_1$ holds with high probability due to the correctness of the CountSketch and L_2-norm estimation data structures. We next note that by Lemma 7.5, the probability that $\mathcal{E}_2$ occurs is $O(\epsilon)$.

Next, note that the probability that for a fixed $j\in[n]$, u_j satisfies $\frac{y_j^2}{u_j}\ge\frac{C\log n}{\epsilon}\cdot\widehat{Y}$ is at most $\frac{\epsilon}{C'\log n}\frac{y_j^2}{\|y\|_2^2}$ for some constant C'. Thus summing over all $j\in[n]$, the probability that there exist an additional $j\in[n]$ for which $|w_j|>\sqrt{\frac{C\log n}{\epsilon}}\cdot\widehat{Y}$ is $O(\epsilon)$. Thus the probability that $\mathcal{E}_3$ occurs is $O(\epsilon)$.

Finally, conditioned on $\neg\mathcal{E}_2$, we have that $\widehat{Z}\le\sqrt{\frac{C\log n}{\epsilon}}\cdot\widehat{Y}$. Then conditioning on $\mathcal{E}_1$, we have $\|z\|_2\le\widehat{Z}$ and thus $|\widehat{w_i}-w_i|\le\epsilon\widehat{Z}\le\sqrt{C\epsilon\log n}\widehat{Y}$, so that $\mathcal{E}_4$ can only occur for $\sqrt{\frac{C\log n}{\epsilon}}\cdot\widehat{Y}\le|w_i|\le\sqrt{\frac{C\log n}{\epsilon}}\cdot\widehat{Y}$, which is at most probability $O\left(\frac{\epsilon^2}{C\log n}\frac{y_i^2}{\widehat{Y}^2}\right)$, over the randomness of u_i.

In summary, we observe that conditioned on some value being output, the probability that item i is selected is proportional to the event that the events $\mathcal{E}$ and $\mathcal{E}_1$ occur, and none of the events $\mathcal{E}_2,\mathcal{E}_3,\mathcal{E}_4$ occur. The probability that $\mathcal{E}$ occurs is $\frac{\epsilon}{C\log n}\frac{y_i^2}{\widehat{Y}^2}$, which u_i is chosen uniformly at random. Due to the event $\mathcal{E}_1$, the sampling probability is distorted additively by $\frac{1}{\operatorname{poly}(n)}$, while due to the events $\mathcal{E}_2,\mathcal{E}_3,\mathcal{E}_4$, the sampling probability is distorted multiplicatively by $(1+\epsilon)$. Thus conditioned on the event that some index is returned, the probability p_i that index i is returned satisfies

$$(1-\epsilon)\cdot\frac{y_i^2}{\|y\|_2^2}-\frac{1}{\operatorname{poly}(n)}\le p_i\le(1+\epsilon)\cdot\frac{y_i^2}{\|y\|_2^2}+\frac{1}{\operatorname{poly}(n)},$$

as desired. □