A Sketching-Based Single-Pass Tensor Compression Algorithm

Anonymous Author(s)
Affiliation
Address
email

Abstract

Given limited storage resources, many higher order datasets from applications such as video, PDE, and weather benefit from being modeled as a streaming case, where the data are in a form of linear updates. Even with these linear updates, it may be only viable to store the sketches of the data given limited storage. This paper proposes a novel sketching-based low-rank tensor approximation algorithm that only passes through the original tensor once during the calculation. This algorithm also provides a theoretical approximation guarantee, as well as a computational speed comparable to existing non-streaming algorithms. Simulations as well as experiments on real-world weather data show the efficiency of this algorithm.

1 Introduction

1.1 Motivation

Over the last few decades, massive large-scale datasets with natural tensor (multi-dimensional array) representations have arisen from many modern applications. Sketching is commonly used to compress the tensor as the first step [10]. In many of these instances, data exhibits a low-rank structure, allowing further compression through decomposition methods such as CANDECOMP/PARAFAC (CP) and Tucker decomposition [13]. These methods have seen uses in various domains, including computer vision [22], neuroscience [5], data mining [14], with a variety of algorithms developed to scale and speed up the decomposition [1, 4, 18].

In practice, large tensors cannot fit into the memory entirely, thus requiring the storage of different slices in a decentralized storage system. The communication costs between the disk and memory then become nontrivial. Usually the main challenge for scaling up the tensor approximation algorithm is the communication cost between the main memory and hard disk instead of the algorithmic cost (flop counts). If the dataset cannot fit into the memory, we need to ship the input and output of the algorithm back and forth between the memory and the disk. Therefore, it is beneficial to develop "pass-efficient" algorithms that read the tensor into the main memory as few times as possible. Meanwhile, since the decentralized storage system requires passing the data to memory in a sequential manner, modeling this problem as a streaming model is more appropriate. Another advantage of the streaming model is its support for data presented as a sum of linear updates or as slices revealed one at a time. In the current literature, both streaming and pass-efficient algorithms in tensor approximation remain largely unexplored. In addition, an alternative way to adapt the tensor decomposition algorithm to large tensor is to use an distributed storage system [2, 3, 4]. So, we extend our method to the distributed setting with the input tensor stored in multiple nodes, and discuss its performance based on the message passing interface (MPI) from [2].

Our main contribution is to develop a one-pass streaming model for low-rank tensor approximation using sketching with a theoretical approximation guarantee. Our algorithm attains a speed comparable

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

to non-streaming algorithms. We validated our findings in both synthetic examples and in real world climate datasets.

The Streaming Model In the applications with streaming tensor data, we are given a large tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$. We need to update $\mathcal{X}$ through a sequence of linear updates:

$$\mathcal{X} \leftarrow \theta_1 \mathcal{X} + \theta_2 \mathcal{X}, \text{ where } \theta_i \in \mathbb{R}. \tag{1.1}$$

In most scenarios, $\mathcal{X}$ is low-rank, and sometimes sparse. Therefore, we extend the classical idea of sketching to the tensor case, where we only need to store the low-dimensional linear image of $\mathcal{X}$ along different dimensions.

1.2 Review of Previous Work

We first review the development of pass-efficient low-rank matrix approximation. Then we restate the formulation of Tucker Decomposition and review some development of fast and memory-efficient algorithms for tensor decomposition. Some notations will be formally introduced in the next section.

Pass-Efficient Low-Rank Matrix Approximation Randomized algorithms for matrix approximation were first explored by the theoretical computer science community in the early 2000s [9, 17], followed by numerical analysts who further developed more practical algorithms. Applying the sketch method and streaming model to matrices is also not new: in 2008, Woolfe [24] proposed one of the first sketching algorithms for low-rank matrix approximation. Clarkson [6] explicitly frames several problems including low rank matrix approximation under a streaming model. However, their focus is on the theoretical development, while [20] provides a more practical one-pass algorithm for low/fixed rank matrix approximation.

Tucker Decomposition Given a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and target rank $\mathbf{r} = (r_1, \ldots, r_N)$, the Tucker decomposition aims to factor $\mathcal{X}$ into a core tensor and orthogonal matrices multiplied along each mode:

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{U}_1 \times \cdots \times_N \mathbf{U}_N. \tag{1.2}$$

Finding the core tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times \cdots \times r_N}$ and arm factors $\mathbf{U}_n \in \mathbb{R}^{r_n \times I_n}, n \in [N]$ as the solution to the following optimization problem is an NP-hard optimization problem:

$$\min_{\mathcal{G}, \mathbf{U}_n} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{U}_1 \times \cdots \times_N \mathbf{U}_N\|_F, \qquad s.t. \qquad \mathbf{U}_n^\top \mathbf{U}_n = \mathbf{I}. \tag{1.3}$$

The best rank $\mathbf{r}$ Tucker approximation is $\mathcal{G}^\star \times_1 \mathbf{U}_1^\star \times \cdots \times_N \mathbf{U}_N^\star$ assuming $\mathcal{G}^\star, \mathbf{U}_n^\star, n \in [N]$ are the solution for (1.3). In the following paper, for theoretical development, we usually let $[\![\cdot]\!]_\mathbf{r}$ denote the operator returning the best rank $\mathbf{r}$ Tucker approximation; while when we refer to fix rank $\mathbf{r}$ algorithms, we let $[\![\cdot]\!]_\mathbf{r}$ denote the result got from certain algorithms. One most widely-used algorithm is the Higher-Order Orthogonal Iteration method (HOOI)[7], where higher-order SVD (HOSVD) serves as the starting point before applying the Alternating Least Squares (ALS) to reach a local optima of (1.3) [15]. We refer to HOOI in our algorithm implementation.

Pass-Efficient Tucker Decomposition One natural way to make this optimization problem pass-efficient is to replace the ALS typically performed after HOSVD with an one-pass algorithm for SVD along each mode similar to the methods in [6, 19]. Some other memory efficient Tucker Decomposition methods have been developed: [14] devises a way to efficiently update the SVD for each mode under limited memory; [23] suggests a method with linear memory cost and shows statistical guarantees. In addition, randomized algorithms have been applied to tensor decomposition: [21] uses the Monte Carlo method from [9] in the SVD steps of finding the Tucker Decomposition; [8] applies sketching methods [10] to the Canonical Decomposition. However, an one-pass algorithm for Tucker Decomposition has yet to be developed.

2 Methodology

We will describe our main algorithms for computing tensor low-rank and fixed-rank approximations in two stages: sketching and recovery. We start by creating a streaming model. Then, we develop the two-pass and one-pass sketching algorithms with their theoretical guarantees. In the end, we will discuss their computational complexity and storage cost for both sparse and dense input tensors.

2.1 Notation

Our paper follows the notation of [13]. We denote the *scalar*, *vector*, *matrix*, and *tensor*, respectively by lowercase letters, (x) boldface lowercase letters ($\mathbf{x}$) boldface capital letters ($\mathbf{X}$) and Euler script letters ($\mathcal{X}$). For matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$, $\mathbf{X}^\dagger \in \mathbb{R}^{n\times m}$ denotes its *Moore-Penrose pseudoinverse*. In particular, $\mathbf{X}^\dagger = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$, if $m \geq n$ and $\mathbf{X}$ has full column rank; $\mathbf{X}^\dagger = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$, if $m < n$ and $\mathbf{X}$ has full row rank. We let $[N]$ be the set containing $1, \ldots, N$.

For a tensor $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_N}$, its *mode* or *order* is the number of dimensions N. If $I = I_1 = \cdots I_N$, we denote $\mathbb{R}^{I_1\times\cdots\times I_N}$ as $\mathbb{R}^{I^N}$. The inner product of two tensors $\mathcal{X}, \mathcal{Y}$ is defined as $\langle \mathcal{X}, \mathcal{Y}\rangle = \sum_{i_1=1}^{I_1}\cdots\sum_{i_N=1}^{I_n} \mathcal{X}_{i_1\ldots i_N}\mathcal{Y}_{i_1\ldots i_N}$. The *Frobenius norm* of $\mathcal{X}$ is denoted by $\|\mathcal{X}\|_F = \sqrt{\langle \mathcal{X}, \mathcal{X}\rangle}$. Let $\bar{I} = \Pi_{j=1}^N I_j$ and $I_{(-n)} = \Pi_{j\neq n} I_j$. We denote the *mode-n unfolding* of $\mathcal{X}$ as $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n\times I_{(-n)}}$ and the *mode-n rank* as the rank of the mode-n unfolding. We define the *rank* of $\mathcal{X}$ as $\mathbf{r}(\mathcal{X}) = (r_1, \ldots, r_N)$ if its *mode-n rank* is r_n for all $n \in [n]$. The tensor with all entries equal to 0 except for the entries with the same indices is called a *superdiagonal* tensor.

We define a fix rank operator $[\![\mathcal{X}]\!]_{\mathbf{r}}$ which maps a tensor $\mathcal{X}$ to a rank $\mathbf{r}$ approximation of $\mathcal{X}$. For example, it refers to the rank $\mathbf{r}$ Tucker decomposition in this paper. $\mathcal{X} \times_n \mathbf{U}$ denotes the *mode-n (matrix) product* of $\mathcal{X}$ with $\mathbf{U} \in \mathbb{R}^{J\times I_n}$, with size $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$, that is:

$$\mathcal{Y} = \mathcal{X} \times_n \mathbf{U} \iff \mathbf{G}^{(n)} = \mathbf{U}\mathbf{X}^{(n)}. \tag{2.1}$$

For each mode-n unfolding $\mathbf{X}^{(n)}$, we use $(\tau_\rho^{(n)})^2$ to denote its *ρth tail energy* as

$$(\tau_\rho^{(n)})^2 = \sum_{k>\rho}^{\min(I_n, I_{(-n)})} \sigma_k^2(\mathbf{X}^{(n)}), \tag{2.2}$$

where $\sigma_k(\mathbf{X}^{(n)})$ is the kth largest singular values for $\mathbf{X}^{(n)}$. We will review more properties of tensor operators in Appendix J.

2.2 Sketching and Linear Update

The Sketch First, we draw a series of independent random test matrices:

$$\mathbf{\Omega}_1, \mathbf{\Omega}_2, \ldots, \mathbf{\Omega}_N \quad \text{and} \quad \mathbf{\Phi}_1, \mathbf{\Phi}_2, \ldots, \mathbf{\Phi}_N, \tag{2.3}$$

with $\mathbf{\Omega}_n \in \mathbb{R}^{I_{(-n)}\times k}$ and $\mathbf{\Phi}_n \in \mathbb{R}^{s\times I_n}$, $n \in [N]$. For theoretical development, we focus on the case where they are independent standard normal matrices. In Algorithm 1, we define the sketches of the tensor $\mathcal{X}$ as $\mathcal{Z} \in \mathbb{R}^{s^N}$, $\mathbf{G}_1, \ldots, \mathbf{G}_N$, $\mathbf{G}_n \in \mathbb{R}^{I_n\times k}$, given by:

$$\mathbf{G}_n = \mathbf{X}^{(n)}\mathbf{\Omega}_n = \mathbf{Q}_n\mathbf{R}_n \quad \text{and} \quad \mathcal{Z} = \mathcal{X} \times_1 \mathbf{\Phi}_1 \times \cdots \times_N \mathbf{\Phi}_N, \tag{2.4}$$

where $\mathbf{Q}_n \in \mathbb{R}^{I_n\times k}, \mathbf{R}_n \in \mathbb{R}^{k\times k}$ is the QR decomposition of $\mathbf{G}_n, n \in [N]$. Notice that these sketches are linear with respect to the entries of the original tensor, so they could be computed in the streaming manner with a single pass, even though we write them in the compact form. $\mathbf{G}_1, \ldots, \mathbf{G}_N$ capture the information $\mathcal{X}$ along each dimension, and $\mathcal{Z}$ records the core information of $\mathcal{X}$. Storing the sketches, $\mathcal{Z}$ and $\mathbf{G}_n, n \in [N]$, costs $\sum_{n=1}^N I_n \cdot k + s^N$. For linear update or recovery, at the first glance, we need information of all test matrices which is quite expensive. As shown next, we can take advantage of the deterministic nature of pseudo-random number generators by only storing the generating schemes. By doing this, we can reduce the memory use to $\sum_{n=1}^N (kI_{(-n)} + sI_N)$.

Role of Pseudo-randomness Interestingly enough, the pseudo-random number generator is a deterministic process which only requires recording a few initial parameters to generate the whole process, such as the famous Mersenne Twister [16]. Thanks to this property (luckily they are not true random variables), we assume all test matrices are given under same random environment and thus we will not explicitly mention their generation in further sections. In practice, there is no truly random number generator. Thus, it would be interesting to explore theoretical properties under a weak assumption like pairwise independence but this is not the concern for this paper.

Linear Update The sketching procedure can be directly applied to the streaming case with linear updates as shown in Algorithm 2. As mentioned above, to achieve memory/storage efficiency, all test matrices are generated under the same environment during the sketching stage.

Algorithm 1 Computing Tensor Sketches

function COMPUTESKETCH($\mathcal{X}, k, s$)
  $\mathcal{Z} \leftarrow \mathcal{X} \times_1 \mathbf{\Phi}_1 \times \cdots \times_N \mathbf{\Phi}_N$
  for $n = 1 \dots N$ **do**
    $\mathbf{G}_n \leftarrow \mathbf{X}^{(n)} \mathbf{\Omega}_n$
  end for
  return $(\mathbf{G}_1, \dots, \mathbf{G}_N, \mathcal{Z})$
end function

Algorithm 2 Linear Update

function LINEARUPDATES($\mathcal{H}, \mathbf{G}_1, \dots, \mathbf{G}_N, \mathcal{Z}; \theta_1, \theta_2$)
  for $n = 1, \dots, N$ **do**
    $\mathbf{G}_n \leftarrow \theta_1 \mathbf{G}_n + \theta_2 \mathbf{H}^{(n)} \mathbf{\Omega}_n$
  end for
  $\mathcal{Z} \leftarrow \theta_1 \mathcal{Z} + \theta_2 \mathcal{H} \times_1 \mathbf{\Phi}_1 \times \cdots \times_N \mathbf{\Phi}_N$
  return $(\mathbf{G}_1, \dots, \mathbf{G}_N, \mathcal{Z})$
end function

2.3 Low-Rank Approximation

We develop a one-pass streaming algorithm for low rank approximation. "One-pass" means that we only access $\mathcal{X}$ once during the approximation procedure. Like [19], we construct a core tensor, $\mathcal{W} \in \mathbb{R}^{k^N}$.

$$\mathcal{W} = \mathcal{Z} \times_1 (\mathbf{\Phi}_1 \mathbf{Q}_1)^\dagger \cdots \times_N (\mathbf{\Phi}_N \mathbf{Q}_N)^\dagger, \tag{2.5}$$

where $(\mathbf{\Phi}_n \mathbf{Q}_n)^\dagger \in \mathbb{R}^{k \times s}$. Then we can obtain the approximated tensor $\hat{\mathcal{X}}$ as

$$\hat{\mathcal{X}} = \mathcal{W} \times_1 \mathbf{Q}_1 \times \cdots \times_N \mathbf{Q}_N. \tag{2.6}$$

One key element behind this approach is $\mathcal{X} \approx \mathcal{X} \times_1 \mathbf{Q}_1 \mathbf{Q}_1^T \times \cdots \times_N \mathbf{Q}_N \mathbf{Q}_N^T$ for certain types of $\mathcal{X}$. As in Lemma B.1, we extend [10]'s result in randomized linear algebra, $\mathbf{X} \approx \mathbf{Q}\mathbf{Q}^T\mathbf{X}$, where $\mathbf{Q}$ is the orthonormal basis in the QR factorization of $\mathbf{X}\mathbf{\Omega}$ and $\mathbf{X}\mathbf{\Omega}$ is the sketch of $\mathbf{X}$ along its column space. A direct application of this finding leads to the two-pass low rank approximation as in Algorithm 7. However, in two-pass sketching, we need to access $\mathcal{X}$ again to compute the core tensor when constructing the core tensor. This is impractical in a decentralized setting. Therefore, we developed the one-pass sketching algorithm, which does not use $\mathcal{X}$ during the tensor recovery, but its sketch $\mathcal{Z}$ instead.

The idea behind the reformulation of the core sketch in one-pass sketching is to replace $\mathcal{X}$ with its low-dimensional projection with the dimensional reduction maps:

$$\begin{aligned} \mathcal{Z} &= \mathcal{X} \times_1 \mathbf{\Phi}_1 \times \cdots \times_N \mathbf{\Phi}_N \\ &\approx (\mathcal{X} \times_1 \mathbf{\Phi}_1 \times \cdots \times_N \mathbf{\Phi}_N) \times_1 \mathbf{Q}_1 \mathbf{Q}_1^T \times \cdots \times_N \mathbf{Q}_N \mathbf{Q}_N^T \\ &\approx \mathcal{X} \times_1 \mathbf{\Phi}_1 \times_1 \mathbf{Q}_1 \mathbf{Q}_1^T \times \cdots \times_N \mathbf{\Phi}_N \times_N \mathbf{Q}_N \mathbf{Q}_N^T. \end{aligned} \tag{2.7}$$

After recursively multiplying the pseudo-inverse of $\mathbf{\Phi}_n \mathbf{Q}_n$, we can see the RHS equal to $\mathcal{X} \times_1 \mathbf{Q}_1^T \times \cdots \times_N \mathbf{Q}_N^T$ which is exactly the core tensor for two-pass algorithm shown in Algorithm 7. Therefore, we can still control the approximation error like applying methods proposed in [6, 20].

Lemma C.2 will provide a formal analysis of the error bound for the core tensor approximation and Theorem 3.1 will use it to construct the bound for the low rank approximation. For all theoretical analysis above and below, we assume the Gaussian dimension reduction map.

2.4 Fixed-Rank Approximation

Suppose we want to use a streaming algorithm to get a Tucker decomposition with a user-specified rank, $\mathbf{r} = (r_1, \dots, r_N)$. Let $[\![\mathcal{W}]\!]_\mathbf{r}$ be the fixed rank approximation from the solution in (1.3). Then, Lemma 2.1 enables us to compute the smaller fixed-rank approximation (Algorithm 4) from the low-rank approximation (Algorithm 3) by performing a fixed-rank approximation on the core tensor $\mathcal{W}$.

Algorithm 3 One-Pass Low-Rank Approximation

function ONEPASSLOWRANKRECOVERY($\mathbf{G}_1, \ldots, \mathbf{G}_N, \mathcal{Z}$)
  for $n = 1 \ldots N$ **do**
    $(\mathbf{Q}_n, \sim) \leftarrow \mathrm{QR}(\mathbf{G}_n)$
  end for
  $\mathcal{W} \leftarrow \mathcal{Z} \times_1 (\mathbf{\Phi}_1 \mathbf{Q}_1)^\dagger \cdots \times_N (\mathbf{\Phi}_N \mathbf{Q}_N)^\dagger$
  $\hat{\mathcal{X}} \leftarrow \mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N$
  return $\hat{\mathcal{X}}$
end function

Lemma 2.1. *Suppose $[\![\cdot]\!]_{\mathbf{r}}$ is the best rank $\mathbf{r}$ tucker approximation operator (theoretically). Then for a tensor $\mathcal{W} \in \mathbb{R}^{k^N}$ and orthogonal matrices: $\mathbf{Q}_n \in \mathbb{R}^{k \times r_n}, n \in [N]$, with $\max_{n=1}^N r_n \leq k$,*

$$[\![\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N]\!]_{\mathbf{r}} = [\![\mathcal{W}]\!]_{\mathbf{r}} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N. \tag{2.8}$$

In particular, if the best rank $\mathbf{r}$ tucker approximation $[\![\mathcal{W}]\!]_{\mathbf{r}} = \mathcal{C} \times_1 \mathbf{H}_1 \times \cdots \times_N \mathbf{H}_N$, then $[\![\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N]\!]_{\mathbf{r}} = \mathcal{C} \times_1 \mathbf{Q}_1\mathbf{H}_1 \times \cdots \times_N \mathbf{Q}_N\mathbf{H}_N$.

Algorithm 4 One-Pass Fixed-Rank Approximation

Input: r is $N \times 1$ array of the target rank with $\max_{n=1}^N r_n \leq k < .5s$.
function ONEPASSFIXRANKRECOVERY($\mathbf{G}_1, \ldots, \mathbf{G}_N, \mathcal{Z}, \mathbf{r}, k, s$)
  for $n = 1 \ldots N$ **do**
    $(\mathbf{Q}_n, \sim) \leftarrow \mathrm{QR}(\mathbf{G}_n)$
  end for
  $\mathcal{W} \leftarrow \mathcal{Z} \times_1 (\mathbf{\Phi}_1 \mathbf{Q}_1)^\dagger \cdots \times_N (\mathbf{\Phi}_N \mathbf{Q}_N)^\dagger$
  $\hat{\mathcal{X}} \leftarrow [\![\mathcal{W}]\!]_{\mathbf{r}} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N$
  return $\hat{\mathcal{X}}$
end function

2.5 Implementation and Costs

In practice, the main bottleneck for deploying algorithms on large-scale tensor datasets is the cost of communication between the disk and the memory. Our method only requires tensor $\mathcal{X}$ to go through the memory once. The two-pass algorithm (Algorithm 7) requires tensor $\mathcal{X}$ to go through the memory twice, while HOOI requires passing the whole tensor $N + 1$ times during the higher-order SVD and once per iteration during the ALS . However, we will still discuss the storage cost and computational complexity (flops) for a complete analysis.

Storage Cost In higher order SVD, we need to store the orthonormal singular vectors and core tensor for $k^N + k\sum_{n=1}^N I_n$. Two-pass algorithms (Algorithm 7, 8) require the exactly same space to store the sketches in (2.3) and linkage tensor $\mathcal{W}$ but with much higher pass-efficiency. The one-pass algorithms (Algorithm 3, 4) only need to store the core sketch $\mathcal{Z}$ in addition with a total storage cost $s^N + k\sum_{n=1}^N I_n$.

Computational Complexity Let the compression factor $\delta_1 := \frac{k}{\min\{I_1,\ldots,I_n\}}$, and $\delta_2 := \frac{s}{\min\{I_1,\ldots,I_n\}}$, the sparse factor μ be the proportion of non-empty entries in $\mathcal{X}$. We list the computational complexity for sketching and recovery stage separately in Table 1. Even though our main contribution is the pass-efficiency, Algorithm 7 attains the same level of computational complexity, and Algorithm 3 only requires a slightly more in terms of a different compression factor.

2.6 Distributed Setting

In practice, people usually store the data in a distributed system. We can further extend our algorithm to adapt to distributed systems during the data passing stage, that is the sketching stage (Algorithm 1). We adopt the MPI of [2] to compute the tensor-matrix product. During the recovery stage in

	Stage	Flops (Batch)	Flops (Streaming)
Two-Pass Sketching	Sketch	$\mathcal{O}(kN\bar{I})$	$\mathcal{O}(\mu kN\bar{I})$
	Recovery	$\mathcal{O}(\frac{k(1-\delta_1^N)\bar{I}}{1-\delta_1} + k^2(\sum_{n=1}^N I_n))$	$\mathcal{O}(\mu k^N N(\sum_{n=1}^N I_n) + k^2(\sum_{n=1}^N I_n))$
	Total	$\mathcal{O}((\frac{k(1-\delta_1^N)}{1-\delta_1} + Nk)\bar{I})$	$\mathcal{O}(\mu kN\bar{I} + \mu k^N N(\sum_{n=1}^N I_n))$
One-Pass Sketching	Sketch	$\mathcal{O}((\frac{s(1-\delta_2^N)}{1-\delta_2} + Nk)\bar{I})$	$\mathcal{O}(\mu kN\bar{I} + \mu s^N N(\sum_{n=1}^N I_n))$
	Recovery	$\mathcal{O}(k^2(\sum_{n=1}^N I_n) + \frac{k^2 s^N(1-(k/s)^N)}{1-k/s})$	$\mathcal{O}(k^2(\sum_{n=1}^N I_n) + \frac{k^2 s^N(1-(k/s)^N)}{1-k/s})$
	Total	$\mathcal{O}((\frac{s(1-\delta_2^N)}{1-\delta_2} + Nk)\bar{I})$	$\mathcal{O}(\mu kN\bar{I} + \mu s^N N(\sum_{n=1}^N I_n))$
Higher-Order SVD		$\mathcal{O}((\frac{k(1-\delta_1^N)}{1-\delta_1} + Nk)\bar{I})$	–

Table 1: Computational Complexity for Low-Rank Approximation: *For tensor X of size $I_1 \times \cdots \times I_N$, its sketches $\mathbf{\Omega}_1, \ldots, \mathbf{\Omega}_N, \mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_N$ has size $I_1 \times k,\ \ldots I_N \times k,\ s \times I_1,\ \ldots, s \times I_N$. μ is the proportion of non-zero entries in $\mathcal{X}$. $\delta_1 := \frac{k}{\min\{I_1,\ldots I_n\}}$, and $\delta_2 := \frac{s}{\min\{I_1,\ldots I_n\}}$ are the compression factor. $\bar{I} = \prod_{i=1}^N I_i$. Note: when calculating the computational complexity, we omit the last step of the recovery and only consider the compression stage. See Appendix I for detailed derivations.*

Algorithm 3, 4, we can compute $\mathcal{W}, \mathbf{Q}_1, \ldots, \mathbf{Q}_n$ from the sketches and run HOOI on $\mathcal{W}$ in different nodes, since the intermediate memory cost is quite small. Then we can distribute the recovered tensor $\hat{\mathcal{X}}$ back to the origin nodes using the method from [2].

3 Main Theorem

In this section, we present the main theory on approximation error for both one-pass and two pass algorithms under low rank approximation (Algorithm 3, Algorithm 7) and fix rank approximation using Tucker decomposition (Algorithm 4, Algorithm 8).

Theorem 3.1. *let $\hat{\mathcal{X}}_1$, $\hat{\mathcal{X}}_2$ be the output from function OnePassLowRankRecovery and TwoPass-LowRankRecovery in Algorithm 3 and Algorithm 7 respectively. Assuming all random test matrices $\mathbf{\Omega}_n, \mathbf{\Phi}_n, 1 \le n \le N$ are standard Gaussian random matrix,*

$$\mathbb{E}\|\mathcal{X} - \hat{\mathcal{X}}_2\|_F^2 \le \min_{1\le\rho<k-1} \sum_{n=1}^N \left(1 + \frac{\rho}{k-\rho-1}\right)(\tau_\rho^{(n)})^2, \tag{3.1}$$

and assuming $s > 2k$,

$$\mathbb{E}\|\mathcal{X} - \hat{\mathcal{X}}_1\|_F^2 \le \min_{1\le\rho<k-1} \left(1 + \frac{k}{s-k-1}\right) \sum_{n=1}^N \left(1 + \frac{\rho}{k-\rho-1}\right)(\tau_\rho^{(n)})^2. \tag{3.2}$$

Now we present the probabilistic error bound for fix rank $\mathbf{r}$ approximation. Lemma 2.1 claims that one pass fix rank approximation returns $[\![\hat{\mathcal{X}}_1]\!]_\mathbf{r}$ where $\hat{\mathcal{X}}_1$ is got from one pass low rank approximation from Algorithm 3 and two pass fix rank approximation returns $[\![\hat{\mathcal{X}}_2]\!]_\mathbf{r}$ where $\hat{\mathcal{X}}_2$ is got from two pass low rank approximation from Algorithm 3.

Corollary 3.2. *let $[\![\hat{\mathcal{X}}_1]\!]_\mathbf{r}$, $[\![\hat{\mathcal{X}}_2]\!]_\mathbf{r}$ be the theoretical output from function OnePassFixRankRecovery and TwoPassFixRankRecovery in Algorithm 4 and Algorithm 8 respectively. Assuming all random test matrices $\mathbf{\Omega}_n, \mathbf{\Phi}_n, 1 \le n \le N$ are standard Gaussian random matrix, then*

$$\mathbb{E}\|\mathcal{X} - [\![\hat{\mathcal{X}}_2]\!]_\mathbf{r}\|_F \le 2 \min_{1\le\rho<k-1} \sqrt{\sum_{n=1}^N \left(1 + \frac{\rho}{k-\rho-1}\right)(\tau_\rho^{(n)})^2} + \|\mathcal{X} - [\![\mathcal{X}]\!]_\mathbf{r}\|_F, \tag{3.3}$$

and if $s > 2k$,

$$\mathbb{E}\|\mathcal{X} - [\![\hat{\mathcal{X}}_1]\!]_\mathbf{r}\|_F \le 2 \min_{1\le\rho<k-1} \sqrt{\left(1 + \frac{k}{s-k-1}\right) \sum_{n=1}^N \left(1 + \frac{\rho}{k-\rho-1}\right)(\tau_\rho^{(n)})^2} + \|\mathcal{X} - [\![\mathcal{X}]\!]_\mathbf{r}\|_F, \tag{3.4}$$

where $[\![\mathcal{X}]\!]_{\mathbf{r}}$ stands for best rank $\mathbf{r}$ tucker approximation. Here $[\![\cdot]\!]_{\mathbf{r}}$ is the operator returns the theoretical best rank $\mathbf{r}$ tucker approximation. We assume fix rank approximation stage $[\![\mathcal{W}]\!]_{\mathbf{r}}$ in both Algorithm 4 and 8 could attain the theoretical optimum. Then by lemma 2.1, the algorithms 4 and 8 will return $[\![\hat{\mathcal{X}}_1]\!]_{\mathbf{r}}$ and $[\![\hat{\mathcal{X}}_2]\!]_{\mathbf{r}}$ respectively.

4 Numerical Experiments

In this section compare the accuracy of two-pass and one-pass fix rank approximation algorithms compared to the classical Tucker decomposition for both synthetic examples and real data.

4.1 Synthetic Examples

4.1.1 Experimental Setup. For simplicity, all numerical experiments with synthetic examples assume that $\mathcal{X}$ has equal side length, I. We consider two scenarios from the the view of signal recovery and approximation error respectively. In case of signal recovery, we set $\mathcal{X} = \mathcal{X}^\natural + \mathcal{E}$, where $\mathcal{X}^\natural$ is the low rank signal tensor and $\mathcal{E}$ is the Gaussian noise tensor. Then, we evaluate the relative error as $\left| [\![\hat{\mathcal{X}}]\!]_{\mathbf{r}} - \mathcal{X}^\natural \|_F \right| / \|\mathcal{X}^\natural\|_F$, where $[\![\hat{\mathcal{X}}]\!]_{\mathbf{r}}$ is rank $\mathbf{r}$ approximation by the one-pass or two-pass fixed rank algorithm.

From the view of approximation error, we set $\mathcal{X}$ to be a super-diagonal tensor with decaying values along super-diagonal after rth entry. Under this scenario, we evaluate the algorithm accuracy by the relative error: $\left| [\![\hat{\mathcal{X}}]\!]_{\mathbf{r}} - \mathcal{X} \|_F \right| / \|\mathcal{X}\|_F$, where $[\![\hat{\mathcal{X}}]\!]_{\mathbf{r}}$ is the rank $\mathbf{r}$ approximation by our two algorithms.

Now we present the details of our data-generating procedure. In the low rank plus noise case, we let γ be the *noise level*. Then, given a low rank tensor $\mathcal{X}^\natural \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ as our true signal, we add a random noise tensor with each element from $\mathcal{N}(0, \sigma^2)$ where $\sigma^2 = \gamma^2 \|\mathcal{X}^\natural\|_F^2 / \bar{I}$ and $\bar{I} = \prod_{n=1}^N I_n$. Since $\|\mathcal{X}^\natural\|_F$ represents the true signal strength, the square of noise to signal ratio $\gamma^2 \approx \frac{\mathbb{E}\|E\|^2}{\|\mathcal{X}^\natural\|_F^2} = \frac{\sigma^2 \bar{I}}{\|\mathcal{X}^\natural\|_F^2}$. Additionally, given Corollary 3.2, we set $s = 2k + 1$ in the one-pass algorithms.

1. Superdiagonal + Noise: Superdiagonal tensor with rank r + $\sqrt{\frac{\gamma^2 \cdot r}{I^N}}\mathcal{N}(0,1)$. we consider three settings with the noise level $\gamma = 0.01, 0.1, 1$ to represent the low noise, medium noise and high noise cases,
2. Polynomial Decay:

$$\mathcal{X} = \mathrm{superdiag}(1, \ldots, 1, 2^{-\mathrm{t}}, 3^{-\mathrm{t}}, \ldots, (\mathrm{N} - \mathrm{r})^{-\mathrm{t}}). \tag{4.1}$$

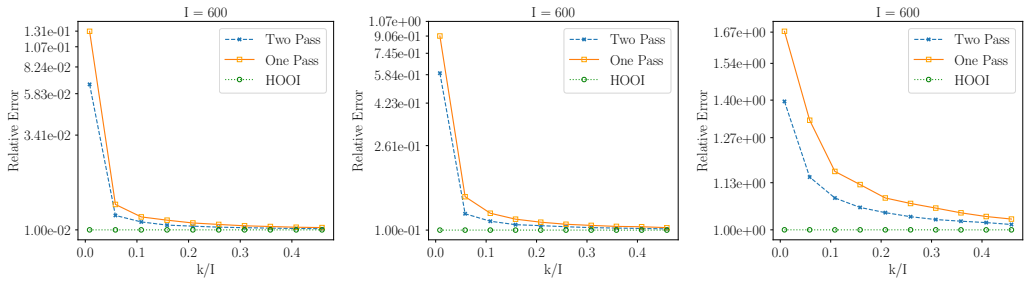
   We consider two cases where $t = 1, 2$ representing slow and fast polynomial decay cases.
3. Exponential Decay:

$$\mathcal{X} = \mathrm{superdiag}(1, \ldots, 1, 10^{-\mathrm{t}}, 10^{-2\mathrm{t}}, \ldots, (10)^{-(\mathrm{N}-\mathrm{r})\mathrm{t}}) \tag{4.2}$$

   We consider two cases where $t = 0.25, 1$ representing slow and fast exponential decay.
4. Low Rank: Generate a core tensor $\mathcal{C} \in \mathbb{R}^{k \times \cdots \times k}$, with each entries $\mathrm{Unif}([0,1])$. Independently generate N orthogonal arm matrices by first creating $\mathbf{A}_1, \ldots, \mathbf{A}_N$, with each entry $\mathcal{N}(0,1)$, and then computing the arm matrices by $(\mathbf{Q}_n, \sim) = \mathrm{QR}(\mathbf{A}_\mathrm{n})$, for $1 \le n \le N$.

$$\mathcal{X} = \mathcal{C} \times_1 \mathbf{Q}_1 \times \cdots \times_N \mathbf{Q}_N + \sqrt{\frac{\gamma^2 \cdot \|\mathcal{X}^\sharp\|_F^2}{I^N}}\mathcal{N}(0,1). \tag{4.3}$$

4.1.2 Numerical Results. Figure 1 and 3 display the performance for Tucker Decomposition, one pass and two pass fixed rank algorithm with different input tensors. In all cases with noisy input tensor with the noise level γ, Tucker decomposition performs very well with a relative error approximate to γ. The relative error for one-pass and one-pass algorithms converges to that of Tucker decomposition as k increases. In particular, the convergence rate is higher when the noise is lower or n is larger. In noiseless cases, we observe a similar pattern: the relative error for two-pass and one-pass algorithms converges to that of Tucker decomposition. In particular, the algorithm achieves convergence at relatively small k. The rate is higher when the decaying rate is higher.

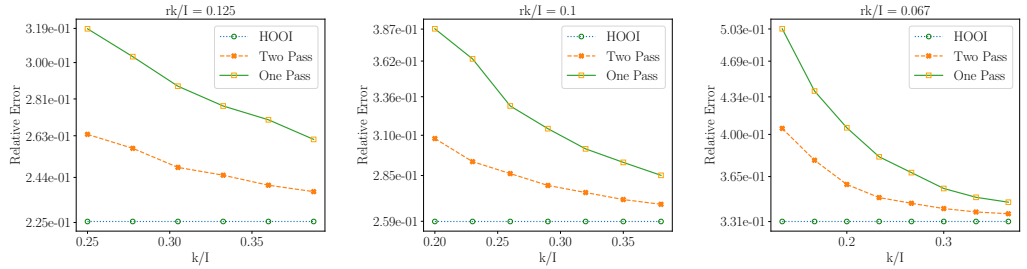


Superdiagonal +Low/Medium/High Noise ($\gamma = 0.01, 0.1, 1$)

Figure 1: Relative error for fixed-rank tensor approximation as a function of the compression factor k/n: *We compare the relative errors presented in log scale for two-pass sketching, one-pass sketching and Tucker decomposition for the input tensor with superdiagonal + noise design ($\gamma = 0.01, 0.1, 1$). Rank r = 5.*

4.2 Application Examples

For real-world application of our methods, we experimented on the state-of-the-art global climate simulation datasets based on the Community Earth System Model (CESM) Community Atmosphere Model (CAM) 5.0 [11, 12]. In particular, we used the dataset on aerosol absorption, which can affect climate through absorbing the solar radiation and changing cloud formation. This dataset recorded absorption at different times, altitudes, longitudes, and latitudes ($240 \times 30 \times 192 \times 288$). Since the Tucker rank of the original tensor remains unknown, we compared different models for different choices of rank, with the result given in Figure 2. Also, we performed additional experiments on the net surface radiative flux and dust aerosol burden data with the results given in Figure 4, 5 in Appendix H.



Aerosol Absorption

Figure 2: Relative error for fixed-rank tensor approximation on climate simulation data ($240 \times 30 \times 192 \times 288$): *We compared the relative errors presented in log scale for two-pass sketching, one-pass sketching and Tucker decomposition with different ranks ($rk/I = 0.125, 0.2, 0.067$).*

5 Conclusion

This paper proposes a new one-pass sketching algorithm for finding the Tucker Decomposition for large tensors in a streaming model. Theorem 3.1 presents an error bound for the tensor approximation which grows linearly with dimension N. In addition, our experiments and application experiments compare the error generated from our algorithm with that of the widely used Tucker Decomposition.

References

[1] Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

[2] Woody Austin, Grey Ballard, and Tamara G Kolda. Parallel tensor compression for large-scale scientific data. In *Parallel and Distributed Processing Symposium, 2016 IEEE International*, pages 912–922. IEEE, 2016.

[3] Venkatesan T Chakaravarthy, Jee W Choi, Douglas J Joseph, Xing Liu, Prakash Murali, Yogish Sabharwal, and Dheeraj Sreedhar. On optimizing distributed tucker decomposition for dense tensors. In *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*, pages 1038–1047. IEEE, 2017.

[4] Joon Hee Choi and S Vishwanathan. Dfacto: Distributed factorization of tensors. In *Advances in Neural Information Processing Systems*, pages 1296–1304, 2014.

[5] Andrzej Cichocki. Tensor decompositions: a new concept in brain data analysis? *arXiv preprint arXiv:1305.0395*, 2013.

[6] Kenneth L Clarkson and David P Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 205–214. ACM, 2009.

[7] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[8] N Benjamin Erichson, Krithika Manohar, Steven L Brunton, and J Nathan Kutz. Randomized cp tensor decomposition. *arXiv preprint arXiv:1703.09074*, 2017.

[9] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[10] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[11] James W Hurrell, Marika M Holland, Peter R Gent, Steven Ghan, Jennifer E Kay, Paul J Kushner, J-F Lamarque, William G Large, D Lawrence, Keith Lindsay, et al. The community earth system model: a framework for collaborative research. *Bulletin of the American Meteorological Society*, 94(9):1339–1360, 2013.

[12] JE Kay, C Deser, A Phillips, A Mai, C Hannay, G Strand, JM Arblaster, SC Bates, G Danabasoglu, J Edwards, et al. The community earth system model (cesm) large ensemble project: A community resource for studying climate change in the presence of internal climate variability. *Bulletin of the American Meteorological Society*, 96(8):1333–1349, 2015.

[13] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[14] Tamara G Kolda and Jimeng Sun. Scalable tensor decompositions for multi-aspect data mining. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 363–372. IEEE, 2008.

[15] Pieter M Kroonenberg and Jan De Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97, 1980.

[16] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[17] Christos H Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, 61(2):217–235, 2000.

[18] Anh-Huy Phan, Petr Tichavskỳ, and Andrzej Cichocki. Fast alternating ls algorithms for high order candecomp/parafac tensor factorizations. *IEEE Transactions on Signal Processing*, 61(19):4834–4846, 2013.

[19] Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Randomized single-view algorithms for low-rank matrix approximation. *arXiv preprint arXiv:1609.00048*, 2016.

[20] Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Practical sketching algorithms for low-rank matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1454–1485, 2017.

[21] Charalampos E Tsourakakis. Mach: Fast randomized tensor decompositions. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 689–700. SIAM, 2010.

[22] M Alex O Vasilescu and Demetri Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *European Conference on Computer Vision*, pages 447–460. Springer, 2002.

[23] Yining Wang and Anima Anandkumar. Online and differentially-private tensor decomposition. In *Advances in Neural Information Processing Systems*, pages 3531–3539, 2016.

[24] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335–366, 2008.

Appendix A Proof for Main Results

A.1 Proof for Theorem 3.1

Let $\tilde{\mathcal{X}}$ denote the compressed tensor,

$$\tilde{\mathcal{X}} = \mathcal{X} \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_N\mathbf{Q}_N^\top. \tag{A.1}$$

The error bound for $\hat{\mathcal{X}}_2$ from two pass algorithm follows directly from lemma B.1 as the compression tensor is exactly the output of TwoPassLowRank in algorithm 7: $\hat{\mathcal{X}}_2 = \tilde{\mathcal{X}}$. Now we turn to the proof for the one-pass algorithm.

We claim that

$$\langle \hat{\mathcal{X}}_1 - \tilde{\mathcal{X}}, \tilde{\mathcal{X}} - \mathcal{X} \rangle = 0. \tag{A.2}$$

To see why, for $n \in [N]$, let

$$\mathcal{Y}_n = \mathcal{X} \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \times_2 \cdots \times_n \mathbf{Q}_n\mathbf{Q}_n^\top, \tag{A.3}$$

and $\mathcal{Y}_0 = \mathcal{X}$. Then

$$\mathcal{X} - \tilde{\mathcal{X}} = \mathcal{Y}_0 - \mathcal{Y}_N = \sum_{n=0}^{N-1} (\mathcal{Y}_n - \mathcal{Y}_{n+1}). \tag{A.4}$$

Besides, given the formula of $\hat{\mathcal{X}}_1$ in algorithm 3 and the definition of $\tilde{\mathcal{X}}_1$, it is not hard to show that

$$\hat{\mathcal{X}}_1 - \tilde{\mathcal{X}} = (\mathcal{W} - \mathcal{X} \times_1 \mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_n^\top) \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N. \tag{A.5}$$

For any $0 \le n < N$,

$$\mathcal{Y}_n - \mathcal{Y}_{n+1} = \mathcal{Y}_n \times_{n+1} (\mathbf{I} - \mathbf{Q}_{n+1}\mathbf{Q}_{n+1}^\top). \tag{A.6}$$

Then with ,

$$\begin{aligned} &\langle \mathcal{Y}_n - \mathcal{Y}_{n+1}, \hat{\mathcal{X}}_1 - \tilde{\mathcal{X}} \rangle = \langle (\mathbf{I} - \mathbf{Q}_{n+1}\mathbf{Q}_{n+1}^\top)\mathbf{Y}_n^{n+1}, \mathbf{Q}_{n+1}\mathbf{B}_n^{(n+1)} \rangle \\ &= \mathrm{Tr}(\mathbf{Y}_{\mathrm{n}}^{(\mathrm{n}+1)}(\mathbf{I} - \mathbf{Q}_{\mathrm{n}+1}\mathbf{Q}_{\mathrm{n}+1}^\top)\mathbf{Q}_{\mathrm{n}+1}\mathbf{B}_{\mathrm{n}}^{(\mathrm{n}+1)}) = 0, \end{aligned} \tag{A.7}$$

where $\mathbf{B}_n^{(n+1)}$ is the mode $(n+1)$th unfolding of tensor $\mathcal{B}_n$, $n \in [N]$, defined as

$$\mathcal{B}_n = (\mathcal{W} - \mathcal{X} \times_1 \mathbf{Q}_1^\top \times \times_N \mathbf{Q}_n^\top) \times_1 \mathbf{Q}_1 \cdots \times_n \mathbf{Q}_n \times_{n+2} \mathbf{Q}_{n+2} \cdots \times_N \mathbf{Q}_N. \tag{A.8}$$

Then (A.2) indicates

$$\mathbb{E}\|\hat{\mathcal{X}}_1 - \mathcal{X}\|_F^2 = \mathbb{E}\|\hat{\mathcal{X}}_1 - \tilde{\mathcal{X}}\|_F^2 + \mathbb{E}\|\tilde{\mathcal{X}} - \mathcal{X}\|_F^2. \tag{A.9}$$

Based the construction of $\hat{\mathcal{X}}_1$ and definition of $\mathcal{W}$ in algorithm 3,

$$\begin{aligned} \mathbb{E}\|\hat{\mathcal{X}}_1 - \tilde{\mathcal{X}}\|_F^2 &= \mathbb{E}\|(\mathcal{W} - \mathcal{X} \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top) \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N\|_F^2 \\ &= \mathbb{E}\|(\mathcal{W} - \mathcal{X} \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top)\|_F^2, \end{aligned} \tag{A.10}$$

where we use the fact that the mode product with an orthogonal matrix will not change the value of Frobenius norm.

Applying the probabilistic error bound in lemma B.1 for the second term in (A.9) and applying probabilistic error bound in lemma C.2 into above equation, taking minimum among $1 \le \rho < k-1$ we could finish proof.

A.2 Proof for Corollary 3.2

Follow the argument in the proof for Proposition 7.1 in [20], we could get

$$\begin{aligned} \|\mathcal{X} - [\![\hat{\mathcal{X}}_2]\!]_{\mathbf{r}}\|_F &\le \|\mathcal{X} - \hat{\mathcal{X}}_2\|_F + \|\hat{\mathcal{X}}_2 - [\![\hat{\mathcal{X}}_2]\!]_{\mathbf{r}}\|_F \\ &\le \|\mathcal{X} - \hat{\mathcal{X}}_2\|_F + \|\hat{\mathcal{X}}_2 - [\![\mathcal{X}]\!]_{\mathbf{r}}\|_F \\ &\le 2\|\mathcal{X} - \hat{\mathcal{X}}_2\|_F + \|\mathcal{X} - [\![\mathcal{X}]\!]_{\mathbf{r}}\|_F. \end{aligned} \tag{A.11}$$

The first and the third line follow from the trangle inequality and the second line follow from the definition of best rank r approximation. Then taking the expectation, applying the result of theorem 3.1 and using the fact $\mathbb{E}y^2 \ge (\mathbb{E}|y|)^2$ for any random variable y, we could finish the proof. And with the same argument, we could show the probabilistic error bound for $[\![\hat{\mathcal{X}}_1]\!]_{\mathbf{r}}$.

Appendix B Probabilistic Analysis of the Compression Error

First, we establish a bound for the expected error at the compression stage. Let $\tilde{\mathcal{X}}$ follow the definition in (A.1).

Lemma B.1. *For any natural number $\rho < k-1$,*

$$\mathbb{E}\|\tilde{\mathcal{X}} - \mathcal{X}\|_F \leq \sum_{n=1}^{N} \left(1 + \frac{\rho}{k-\rho-1}\right) (\tau_\rho^{(n)})^2. \tag{B.1}$$

Proof. The main idea of the proof is to extend the result of [10] to the tensor case. To construct similar bound, we first decompose the compression error into the sum of differences between $\mathcal{Y}_m$ as in (B.2), and then bound each term individually as in (B.7).

Follow the definition of $\mathcal{Y}_n$ in A.3,

$$\|\mathcal{X} - \tilde{\mathcal{X}}\|_F^2 = \|\mathcal{Y}_0 - \mathcal{Y}_N\|_F^2 = \|\sum_{n=0}^{N-1} (\mathcal{Y}_n - \mathcal{Y}_{n+1})\|_F^2. \tag{B.2}$$

For any $0 \leq m < n < N$,

$$\mathcal{Y}_m - \mathcal{Y}_{m+1} = \mathcal{Y}_m \times_{(m+1)} (\mathbf{I} - \mathbf{Q}_{m+1}\mathbf{Q}_{m+1}^\top). \tag{B.3}$$

and

$$\mathcal{Y}_n - \mathcal{Y}_{n+1} = \underbrace{\left[\mathcal{X} \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \times_2 \cdots \times_m \mathbf{Q}_m\mathbf{Q}_m^\top \times_{(m+2)} \cdots \times_n \mathbf{Q}_n\mathbf{Q}_n^\top \times_{(n+1)} (\mathbf{I} - \mathbf{Q}_{(n+1)}\mathbf{Q}_{(n+1)}^\top)\right]}_{\mathcal{A}}$$
$$\times_{(m+1)} \mathbf{Q}_{m+1}\mathbf{Q}_{m+1}^\top. \tag{B.4}$$

Then second part of lemma E.3 claims that

$$\langle \mathcal{Y}_m - \mathcal{Y}_{m+1}, \mathcal{Y}_n - \mathcal{Y}_{n+1} \rangle = \langle \mathcal{Y}_m \times_{(m+1)} (\mathbf{I} - \mathbf{Q}_{m+1}\mathbf{Q}_{m+1}^\top), \mathcal{A} \times_{m+1} \mathbf{Q}_{m+1}\mathbf{Q}_{m+1}^\top \rangle = 0. \tag{B.5}$$

It indicates that we could decompose error with the Pythagorean Theorem as:

$$\|\mathcal{X} - \tilde{\mathcal{X}}\|_F^2 = \sum_{n=0}^{N-1} \|(\mathcal{Y}_n - \mathcal{Y}_{n+1})\|_F^2. \tag{B.6}$$

Now we shall bound $\|\mathcal{Y}_n - \mathcal{Y}_{n+1}\|_F^2$ for each n.

$$\begin{aligned} \|\mathcal{Y}_n - \mathcal{Y}_{n+1}\|_F^2 &= \|\mathcal{X} \times_{(n+1)} (\mathbf{I} - \mathbf{Q}_{n+1}\mathbf{Q}_{n+1}^\top) \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \cdots \times_n \mathbf{Q}_n\mathbf{Q}_n^\top\|_F^2 \\ &\leq \|\mathcal{X} \times_{(n+1)} (\mathbf{I} - \mathbf{Q}_{n+1}\mathbf{Q}_{n+1}^\top)\|_F^2 \\ &= \|(\mathbf{I} - \mathbf{Q}_{n+1}\mathbf{Q}_{n+1}^\top)\mathbf{X}^{(n)}\|_F^2, \end{aligned} \tag{B.7}$$

where the second line follows from the fact that the projection is contractive with second part in lemma E.3. Applying lemma E.2 to last line of above inequality, we could show that

$$\mathbb{E}\|\mathcal{Y}_n - \mathcal{Y}_{n+1}\|_F^2 \leq \left(1 + \frac{\rho}{k-\rho-1}\right) (\tau_\rho^{(n+1)})^2. \tag{B.8}$$

Sum all the upper bounds up for each term in (B.6), we finish the proof. □

Appendix C Probabilistic Analysis of Core Sketch Error

Let us introduce the orthonormal matrix $\mathbf{Q}_n^\perp$ whose range is complementary to that of $\mathbf{Q}_n$ for each n, i.e., $\mathbf{Q}_n^\perp(\mathbf{Q}_n^\perp)^\top = \mathbf{I} - \mathbf{Q}_n\mathbf{Q}_n^\top$. Next, we denote

$$\mathbf{\Phi}_n^Q = \mathbf{\Phi}_n\mathbf{Q}_n \quad \mathbf{\Phi}_n^{Q^\perp} = \mathbf{\Phi}_n\mathbf{Q}_n^\perp. \tag{C.1}$$

Apparently, conditional on $\mathbf{Q}_n$, $\mathbf{\Phi}_n^Q$ and $\mathbf{\Phi}_n^{Q^\perp}$ are independent of each other.

C.1 Decomposition of Core Sketch Error

This session, we show a decomposition formula for core sketch error.

Lemma C.1. *Assume $\mathbf{\Phi}_n$ has full column rank (this holds with probability 1 in fact). Then*

$$\mathcal{W} - \mathcal{X} \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top = \sum_{(i_1,\ldots,i_N)\in\{0,1\}^N, \prod_{j=1}^N i_j \neq 0} \mathcal{Y}_{i_1\ldots i_N}, \tag{C.2}$$

where

$$\begin{aligned}\mathcal{Y}_{i_1\ldots i_N} = \mathcal{X} \times_1 &\left(\mathbb{1}_{i_1=0}\mathbf{Q}_1^\top + \mathbb{1}_{i_1=1}(\mathbf{\Phi}_1^Q)^\dagger \mathbf{\Phi}_1^{Q^\perp}(\mathbf{Q}_1^\perp)^\top)\right) \\ &\times_2 \cdots \times_N \left(\mathbb{1}_{i_N=0}\mathbf{Q}_N^\top + \mathbb{1}_{i_1=1}(\mathbf{\Phi}_N^Q)^\dagger \mathbf{\Phi}_N^{Q^\perp}(\mathbf{Q}_N^\perp)^\top)\right).\end{aligned} \tag{C.3}$$

Proof. We could write $\mathcal{W}$ as

$$\begin{aligned}&\mathcal{W} = \mathcal{Z} \times_1 (\mathbf{\Phi}_1\mathbf{Q}_1)^\dagger \times_2 \cdots \times_N (\mathbf{\Phi}_N\mathbf{Q}_N)^\dagger \\ &= (\mathcal{X} - \tilde{\mathcal{X}}) \times_1 \mathbf{\Phi}_1 \times_2 \cdots \times_N \mathbf{\Phi}_N \times_1 (\mathbf{\Phi}_1\mathbf{Q}_1)^\dagger \times_2 \cdots \times_N (\mathbf{\Phi}_N\mathbf{Q}_N)^\dagger \\ &+ \tilde{\mathcal{X}} \times_1 \mathbf{\Phi}_1 \times_2 \cdots \times_N \mathbf{\Phi}_N \times_1 (\mathbf{\Phi}_1\mathbf{Q}_1)^\dagger \times_2 \cdots \times_N (\mathbf{\Phi}_N\mathbf{Q}_N)^\dagger.\end{aligned} \tag{C.4}$$

We could simplify the second term as

$$\begin{aligned}&\tilde{\mathcal{X}} \times_1 \mathbf{\Phi}_1 \times_2 \cdots \times_N \mathbf{\Phi}_N \times_1 (\mathbf{\Phi}_1\mathbf{Q}_1)^\dagger \times_2 \cdots \times_N (\mathbf{\Phi}_N\mathbf{Q}_N)^\dagger \\ &= \mathcal{X} \times_1 (\mathbf{\Phi}_1\mathbf{Q}_1)^\dagger \mathbf{\Phi}_1\mathbf{Q}_1\mathbf{Q}_1^\top \times_2 \cdots \times_N (\mathbf{\Phi}_N\mathbf{Q}_N)^\dagger \mathbf{\Phi}_N\mathbf{Q}_N\mathbf{Q}_N^\top \\ &= \mathcal{X} \times_1 \mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_N^\top.\end{aligned} \tag{C.5}$$

The last equation comes from the fact that for a matrix $\mathbf{A}$ with shape $s \times k$, if $s > k$, $A^\dagger A = I_k$. Therefore

$$\begin{aligned}&\mathcal{W} = (\mathcal{X} - \tilde{\mathcal{X}}) \times_1 \mathbf{\Phi}_1 \times_2 \cdots \times_N \mathbf{\Phi}_N \times_1 (\mathbf{\Phi}_1\mathbf{Q}_1)^\dagger \times_2 \cdots \times_N (\mathbf{\Phi}_N\mathbf{Q}_N)^\dagger \\ &+ \mathcal{X} \times_1 \mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_N^\top.\end{aligned} \tag{C.6}$$

Then the error could be decomposed as

$$\begin{aligned}&(\mathcal{X} - \tilde{\mathcal{X}}) \times_1 \mathbf{\Phi}_1 \times_2 \cdots \times_N \mathbf{\Phi}_N \times_1 (\mathbf{\Phi}_1\mathbf{Q}_1)^\dagger \times_2 \cdots \times_N (\mathbf{\Phi}_N\mathbf{Q}_N)^\dagger \\ &= (\mathcal{X} - \tilde{\mathcal{X}}) \times_1 (\mathbf{\Phi}_1\mathbf{Q}_1)^\dagger \mathbf{\Phi}_1 \times_2 \cdots \times_N (\mathbf{\Phi}_N\mathbf{Q}_N)^\dagger \mathbf{\Phi}_N \\ &= (\mathcal{X} - \tilde{\mathcal{X}}) \times_1 (\mathbf{\Phi}_1\mathbf{Q}_1)^\dagger \mathbf{\Phi}_1(\mathbf{Q}_1\mathbf{Q}_1^\top + \mathbf{Q}_1^\perp(\mathbf{Q}_1^\perp)^\top) \cdots \times_N (\mathbf{\Phi}_N\mathbf{Q}_N)^\dagger \mathbf{\Phi}_N(\mathbf{Q}_N\mathbf{Q}_N^\top + \mathbf{Q}_N^\perp(\mathbf{Q}_N^\perp)^\top) \\ &= (\mathcal{X} - \tilde{\mathcal{X}}) \times_1 (\mathbf{Q}_1^\top + (\mathbf{\Phi}_1^Q)^\dagger \mathbf{\Phi}_1^{Q^\perp}(\mathbf{Q}_1^\perp)^\top) \times_2 \cdots \times_N (\mathbf{Q}_N^\top + (\mathbf{\Phi}_N^Q)^\dagger \mathbf{\Phi}_N^{Q^\perp}(\mathbf{Q}_N^\perp)^\top).\end{aligned} \tag{C.7}$$

We need to sum up these $2^N - 1$ terms. To simply the summation, we claim followings:

1. $(\mathcal{X} - \tilde{\mathcal{X}}) \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top = 0$.
2. for any $1 \leq n \leq N$, $\tilde{\mathcal{X}} \times_n (\mathbf{\Phi}_n^Q)^\dagger \mathbf{\Phi}_n^{Q^\perp}(\mathbf{Q}_n^\perp)^\top = 0$.

Here 0 means a tensor with all zero elements. There two claims could be got straightly from exchange rule of mode product given by J.1. Then follow the definition of $\mathcal{Y}_{i_1\ldots i_N}$ in (C.3), we could write (C.7) as

$$\sum_{(i_1,\ldots,i_N)\in\{0,1\}^N, \prod_{j=1}^N i_j \neq 0} \mathcal{Y}_{i_1\ldots i_N} \tag{C.8}$$

to complete the proof. □

C.2 Probabilistic Core Error Bound

In this section, we derive the probabilistic error bound based on the core error decomposition shown in lemma C.1.

Lemma C.2. *If $s > 2k$, each element of $\mathbf{\Phi}_n, \mathbf{\Omega}_n$ is from standard Gaussian distribution for all $n, n \in [N]$, then for any natural number $1 \leq \rho < k$,*

$$\mathbb{E}\|\mathcal{W} - \mathcal{X} \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top\|_F^2 \leq \frac{k}{s-k-1}\left[\sum_{n=1}^{N}\left(1 + \frac{\rho}{k-\rho-1}\right)(\tau_\rho^{(n)})^2\right]. \tag{C.9}$$

Proof. It suffices to show that

$$\mathbb{E}\left[\|\mathcal{W} - \mathcal{X} \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top\|_F^2 \mid \mathbf{\Omega}_1, \cdots, \mathbf{\Omega}_N\right] \leq \frac{k}{s-k-1}\|\mathcal{X} - \tilde{\mathcal{X}}\|_F^2, \tag{C.10}$$

then we could take an expectation with respect to $\mathbf{\Omega}_1, \cdots, \mathbf{\Omega}_N$ and apply lemma B.1 to finish the proof. Here we use the fact that $\{\mathbf{\Omega_n}, 1 \leq n \leq N\}$ are independent with $\{\mathbf{\Phi_n}, 1 \leq n \leq N\}$ and randomness of $\mathbf{Q}_n$ solely comes from $\mathbf{\Omega}_n$.

Lemma C.1 decompose the core error as summantion of $\mathcal{Y}_{i_1 \cdots i_n}$ on the set $\{i_j \in \{0,1\}, \prod_{j=1}^N i_j \neq 0\}$. We will first show the upper bound for each $\mathcal{Y}_{i_1 \cdots i_n}$. Applying lemma E.2, and noticing on the index set, $\sum_{j=1}^N i_j \geq 1$,

$$\mathbb{E}\left[\|\mathcal{Y}_{i_1 \ldots i_N}\|_F^2 \mid \mathbf{\Omega}_1 \cdots \mathbf{\Omega}_N\right] = \left(\frac{k}{s-k-1}\right)^{\sum_{j=1}^N i_j}\|\mathcal{B}_{i_1 \ldots i_N}\|_F^2 \leq \left(\frac{k}{s-k-1}\right)\|\mathcal{B}_{i_1 \ldots i_N}\|_F^2, \tag{C.11}$$

where we define $\mathcal{B}_{i_1 \ldots i_N}$ as

$$\begin{aligned}&\mathcal{B}_{i_1 \ldots i_N} = \\ &\mathcal{X} \times_1 (\mathbb{1}_{i_1=0}\mathbf{Q}_1\mathbf{Q}_1^\top + \mathbb{1}_{i_1=1}\mathbf{Q}_1^\perp(\mathbf{Q}_1^\perp)^\top) \cdots \times_N (\mathbb{1}_{i_N=0}\mathbf{Q}_N\mathbf{Q}_N^\top + \mathbb{1}_{i_N=1}\mathbf{Q}_N^\perp(\mathbf{Q}_N^\perp)^\top).\end{aligned} \tag{C.12}$$

Here we use the fact that mode product with orthogonal matrix does not change the F norm:

$$\begin{aligned}&\left\|\mathcal{X} \times_1 (\mathbb{1}_{i_1=0}\mathbf{Q}_1\mathbf{Q}_1^\top + \mathbb{1}_{i_1=1}\mathbf{Q}_1^\perp(\mathbf{Q}_1^\perp)^\top) \cdots \times_N (\mathbb{1}_{i_N=0}\mathbf{Q}_N\mathbf{Q}_N^\top + \mathbb{1}_{i_N=1}\mathbf{Q}_N^\perp(\mathbf{Q}_N^\perp)^\top)\right\|_F \\ &= \left\|\mathcal{X} \times_1 (\mathbb{1}_{i_1=0}\mathbf{Q}_1^\top + \mathbb{1}_{i_1=1}(\mathbf{Q}_1^\perp)^\top) \cdots \times_N (\mathbb{1}_{i_N=0}\mathbf{Q}_N^\top + \mathbb{1}_{i_N=1}(\mathbf{Q}_N^\perp)^\top)\right\|_F.\end{aligned} \tag{C.13}$$

Now we show that inner product of two different $\mathcal{B}_{i_1 \cdots i_N}$ is zero. Consider $q_1, q_2 \in \{0,1\}^N$ be the index(binary) vector of length N. For different index q_1, q_2, at least there exists $1 \leq r \leq N$, such that their r-th element is different. Without loss of generality, $q_1(r) = 0$ and $q_2(r) = 1$,

$$\langle \mathcal{B}_{q_1}, \mathcal{B}_{q_2} \rangle = \langle \mathbf{B}_{q_1}^{(r)}, \mathbf{B}_{q_2}^{(r)} \rangle = \langle \ldots \mathbf{Q}_r^\top \mathbf{Q}_r^\perp \ldots \rangle = 0. \tag{C.14}$$

Noticing

$$\sum_{(i_1, \ldots, i_N) \in \{0,1\}^N} \mathcal{B}_{i_1 \ldots i_N} = \mathcal{X}, \tag{C.15}$$

and $\mathcal{B}_{1, \cdots, 1} = \tilde{\mathcal{X}}$ (with all $i_n = 1$), then by Pythagorean theorem, we have

$$\sum_{(i_1, \ldots, i_N) \in \{0,1\}^N, \prod_{j=1}^N i_j \neq 0} \|\mathcal{B}_{i_1 \ldots i_N}\|_F^2 = \|\mathcal{X} - \tilde{\mathcal{X}}\|_F^2. \tag{C.16}$$

Putting all these together,

$$\begin{aligned}&\mathbb{E}\left[\|\mathcal{W} - \mathcal{X} \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top\|_F^2 \mid \mathbf{\Omega}_1, \cdots, \mathbf{\Omega}_N\right] \\ &= \sum_{(i_1, \ldots, i_N) \in \{0,1\}^N, \prod_{j=1}^N i_j \neq 0} \mathbb{E}\left[\|\mathcal{Y}_{i_1 \ldots i_N}\|_F^2 \mid \mathbf{\Omega_1}, \ldots, \mathbf{\Omega}_N\right] \\ &\leq \frac{k}{s-k-1}\left(\sum_{(i_1, \ldots, i_N) \in \{0,1\}^N, \prod_{j=1}^N i_j \neq 0} \|\mathcal{B}_{i_1 \ldots i_N}\|_F^2\right) \\ &= \frac{k}{s-k-1}\|\mathcal{X} - \tilde{\mathcal{X}}\|_F^2.\end{aligned} \tag{C.17}$$

Noticing no $\mathbf{\Omega}_n$ involved in above bound, taking expectation on $\mathbf{\Omega}_n$ completes the proof. □

Appendix D Proof for Lemma 2.1

Proof. First we claim there exists one best rank $\mathbf{r}$ tucker decomposition(there may be more than one optimal solutions),

$$[\![\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N]\!]_{\mathbf{r}} = \mathcal{S} \times_1 \mathbf{U}_1 \cdots \times_N \mathbf{U}_N, \tag{D.1}$$

where $\mathbf{Q}_n \in \mathbb{R}^{k \times I_n}, \mathbf{U}_n \in \mathbb{R}^{r \times I_n}, n \in [N]$, $\mathcal{W} \in \mathbb{R}^{k^N}$ and $\mathcal{S} \in \mathbb{R}^{s^N}$, such that $\mathrm{Col}(\mathbf{U}_\mathrm{n}) \subseteq \mathrm{Col}(\mathbf{Q}_\mathrm{n})$, for all $n \in [N]$. Here $\mathrm{Col}(\mathbf{U}_\mathrm{n})$ denotes the column of space of $\mathbf{U}_n$: the space spanned from columns of $\mathbf{U}_n$. Our argument is that given a solution $[\![\mathcal{S}; \mathbf{U}_n, n \in [N]]\!]$, $[\![\mathcal{S}; \mathbf{U}_n^{\mathbf{Q}_n}\mathbf{Q}_n^\top, n \in [N]]\!]$ is also a solution to the optimization problem (1.3) i.e.,

$$\mathcal{S} \times_1 \mathbf{U}_1^{\mathbf{Q}_1}\mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{U}_N^{\mathbf{Q}_N}\mathbf{Q}_N^\top = \mathcal{S} \times_1 \mathbf{U}_1 \times \cdots \times_N \mathbf{U}_N, \tag{D.2}$$

where $\mathbf{U}_n^{\mathbf{Q}_n}$ and $\mathbf{U}_n^{\mathbf{Q}_n^\perp}$ follow the notations in proof of lemma C.2. It suffices to show that we could replace $\mathbf{U}_1$ with $\mathbf{U}_1^{\mathbf{Q}_1}\mathbf{Q}_1^\top$ still attaining the solution to optimization problem (1.3) and rest follows with the same argument. Rewriting identity matrix,

$$\begin{aligned}
&\|\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N - \mathcal{S} \times_1 \mathbf{U}_1 \cdots \times_N \mathbf{U}_N\|_F^2 \\
&= \|\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N - \mathcal{S} \times_1 [\mathbf{U}_1^{\mathbf{Q}_1}\mathbf{Q}_1^\top + \mathbf{U}_1^{\mathbf{Q}_1^\perp}(\mathbf{Q}_1^\perp)^\top] \times_2 \cdots \times_N \mathbf{U}_N\|_F^2.
\end{aligned} \tag{D.3}$$

Similar to the trick in (B.5) and (C.14), $\forall n_1, n_2 \in [N]$,

$$\begin{aligned}
&\langle \mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N, \mathcal{S} \times_1 \cdots \times_{n_1} (\mathbf{U}_{n_1}^{\mathbf{Q}_{n_1}^\perp}(\mathbf{Q}_{n_1}^\perp)^\top) \cdots \times_N \mathbf{U}_N \rangle = 0 \\
&\langle \mathcal{S} \times_1 \cdots \times_{n_1} (\mathbf{U}_{n_1}^{\mathbf{Q}_{n_1}}\mathbf{Q}_{n_1}^\top) \times_2 \cdots \times_N \mathbf{U}_N, \mathcal{S} \times_1 \cdots \times_{n_2} (\mathbf{U}_{n_2}^{\mathbf{Q}_{n_2}^\perp}(\mathbf{Q}_{n_2}^\perp)^\top) \cdots \times_N \mathbf{U}_N \rangle = 0,
\end{aligned} \tag{D.4}$$

which indicates that

$$\begin{aligned}
&\|\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N - \mathcal{S} \times_1 \mathbf{U}_1 \cdots \times_N \mathbf{U}_N\|_F^2 \\
&= \|\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N - \mathcal{S} \times_1 (\mathbf{U}_1^{\mathbf{Q}_1}\mathbf{Q}_1^\top) \cdots \times_N \mathbf{U}_N\|_F^2 \\
&+ \|\mathcal{S} \times_1 (\mathbf{U}_1^{\mathbf{Q}_1^\perp}(\mathbf{Q}_1^\perp)^\top) \cdots \times_N \mathbf{U}_N\|_F^2.
\end{aligned} \tag{D.5}$$

By definition of minimizer, we could claim

$$\begin{aligned}
&\|\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N - \mathcal{S} \times_1 \mathbf{U}_1 \cdots \times_N \mathbf{U}_N\|_F^2 \\
&= \|\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N - \mathcal{S} \times_1 (\mathbf{U}_1^{\mathbf{Q}_1}\mathbf{Q}_1^\top) \cdots \times_N \mathbf{U}_N\|_F^2
\end{aligned} \tag{D.6}$$

which completes the argument.

Then we could assume $\mathbf{U}_n = \mathbf{G}_n\mathbf{Q}_n$, where both $\mathbf{G}_n \in \mathbb{R}^{r \times k}$and $\mathbf{Q}_n \in \mathbb{R}^{k \times I_n}$ are orthogonal matrices. The optimization problem becomes:

$$\begin{aligned}
&\min_{\mathcal{G}, \mathbf{G}_n} \|\mathcal{W} \times_1 \mathbf{Q}_1 \times_2 \cdots \times_N \mathbf{Q}_N - \mathcal{G} \times_1 \mathbf{G}_1\mathbf{Q}_1 \times_2 \cdots \times_N \mathbf{G}_N\mathbf{Q}_N\|_F^2 \\
&s.t. \quad \mathbf{G}_n^\top\mathbf{G}_n = \mathbf{I}_{r \times r}, \quad n \in [N].
\end{aligned} \tag{D.7}$$

Noticing that the mode product with an orthogonal matrix does not change the Frobenius norm and exchanging rule of mode product (J.1), we could further simply the optimization problem to

$$\begin{aligned}
&\min_{\mathcal{G}, \mathbf{G}_n} \|\mathcal{W} - \mathcal{G} \times_1 \mathbf{G}_1 \times_2 \cdots \times_N \mathbf{G}_N\|_F^2 \\
&s.t. \quad \mathbf{G}_n^\top\mathbf{G}_n = \mathbf{I}_{r \times r}, \quad n \in [N],
\end{aligned} \tag{D.8}$$

which completes the proof. □

Appendix E Technical Lemmas

E.1 Technical Lemmas for sketching Matrix

All the proof for lemmas in this section could be found in chapter 9 and 10 in [10].

Lemma E.1. *Assume that $t > q$. Let $\mathbf{G}_1 \in \mathbb{R}^{t\times q}$ and $\mathbf{G}_2 \in \mathbb{R}^{t\times p}$ be independent standard normal matrices. For any matrix $\mathbf{B}$ with conforming dimensions,*

$$\mathbb{E}\|\mathbf{G}_1^\dagger \mathbf{G}_2 \mathbf{B}\|_F^2 = \frac{q}{t-q}\|\mathbf{B}\|_F^2. \tag{E.1}$$

Lemma E.2. *Suppose that $\mathbf{A}$ is a real $m \times n$ matrtix with singular value $\sigma_1 \geq \sigma_2 \geq \cdots$, choose a target rank $k \geq 2$ and an oversampling parameter $p \geq 2$, where $k + p \leq \min\{m, n\}$. Draw an $n \times (k + p)$ standard Guassian matrix $\mathbf{\Omega}$, and construct the sample matrix $\mathbf{Y} = \mathbf{A\Omega}$, then the expectation of approximation error*

$$\mathbb{E}\|(\mathbf{I} - \mathbf{P}_\mathbf{Y})\mathbf{A}\|_F^2 \leq \left(1 + \frac{k}{p-1}\right)\left(\sum_{j>k}\sigma_j^2\right). \tag{E.2}$$

E.2 Some Facts for Projection of Mode Unfolding of a Tensor

This session generalizes some results of matrix projection to the case where we do projection on the mode-n unfolding matrix of A tensor. It turns out that the similar property holds as expected. The first part of the lemma claims that projection is contractive, and the second one in fact states a version of Pythagorean Theorem.

Lemma E.3. *Given tensors: $\mathcal{X}$, $\mathcal{Y}$ with size $I_1 \times I_2 \times \cdots \times I_N$, and an orthogonal matrix $\mathbf{Q}(\mathbf{Q}^\top\mathbf{Q} = \mathbf{I})$ with size $I_n \times k$. We claim followings:*

1. *Projection contracts the Frobenius norm of*

$$\|\mathcal{X} \times_n \mathbf{Q}\mathbf{Q}^\top\|_F = \|\mathcal{X} \times_n \mathbf{Q}^\top\|_F \leq \|\mathcal{X}\|_F. \tag{E.3}$$

2.

$$\|\mathcal{X} \times_n \mathbf{Q}\mathbf{Q}^\top + \mathcal{Y} \times_n (\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\|_F^2 = \|\mathcal{X} \times_n \mathbf{Q}\mathbf{Q}^\top\|_F^2 + \|\mathcal{Y} \times_n (\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\|_F^2. \tag{E.4}$$

Proof. For first part,

$$\begin{aligned}&\|\mathcal{X} \times_n \mathbf{Q}\mathbf{Q}^\top\|_F^2 = \|\mathbf{Q}\mathbf{Q}^\top\mathbf{X}^{(n)}\|_F^2 = \mathrm{Tr}(\mathbf{X}^{(\mathrm{n})\top}\mathbf{Q}\mathbf{Q}^\top\mathbf{Q}\mathbf{Q}^\top\mathbf{X}^{(\mathrm{n})}) \\ &= \mathrm{Tr}(\mathbf{X}^{(\mathrm{n})\top}\mathbf{Q}\mathbf{Q}^\top\mathbf{X}^{(\mathrm{n})}) = \|\mathbf{Q}^\top\mathbf{X}^{(\mathrm{n})}\|_\mathrm{F}^2 = \|\mathcal{X} \times_\mathrm{n} \mathbf{Q}^\top\|_\mathrm{F}^2 \leq \|\mathbf{X}^{(\mathrm{n})}\|_\mathrm{F}^2 = \|\mathcal{X}\|_\mathrm{F}^2.\end{aligned} \tag{E.5}$$

where we use the factor that projection is contractive for matrix. Tensor operators could be referred in section J.

For second part, it suffices to show that

$$\begin{aligned}&\langle \mathcal{X} \times_n \mathbf{Q}\mathbf{Q}^\top, \mathcal{Y} \times_n (\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\rangle = \langle \mathbf{Q}\mathbf{Q}^\top\mathbf{X}^{(n)}, (\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\mathcal{Y}^{(n)}\rangle \\ &= \mathrm{Tr}(\mathbf{X}^{(\mathrm{n})\top}\mathbf{Q}\mathbf{Q}^\top(\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\mathcal{Y}^{(\mathrm{n})}) = 0.\end{aligned} \tag{E.6}$$

□

Appendix F More Algorithms

Higher-order singular value decomposition (HOSVD) is a classical method in finding low-rank structure in tensors [7]. Usually it serves as the initial step for the alternating least square (ALS) in computing the tucker decomposition. If we assume the tensor to have rank $\mathbf{r} = (r_1, \ldots, r_N)$, then the corresponding HOSVD algorithm is given by Algorithm 5. The HOOI Tucker decomposition (HOSVD and ALS) is given by Algorithm 6.

As discussed in the previous sections, our one pass algorithms (Algorithm 3 and 4) build on the two-pass low-rank and fixed-rank approximation algorithms, which are given by Algorithm 7 and 8.

Algorithm 5 Higher Order SVD

function HOSVD($\mathcal{X}, \mathbf{r}$)
  for $n = 1, \ldots N$ **do**
    $(\mathbf{U}_n, \cdot, \cdot) \leftarrow \mathrm{SVD}_{\mathrm{r_n}}(\mathbf{X}^{(\mathrm{n})})$
  end for
  $\mathcal{S} \leftarrow \mathcal{X} \times_1 \mathbf{U}_1^\top \times \cdots \times_N \mathbf{U}_N^\top$
  $\hat{\mathcal{X}} \leftarrow \mathcal{S} \times_1 \mathbf{U}_1 \times \cdots \times \mathbf{U}_N$
  return $(\mathcal{S}, \mathbf{U}_1, \ldots, \mathbf{U}_N, \hat{\mathcal{X}})$
end function

Algorithm 6 HOOI (Higher Order SVD + ALS)

function TUCKER($\mathcal{X}, \mathbf{r}$)
  $(\mathcal{S}, \mathbf{U}_1, \ldots, \mathbf{U}_N, \_)$ =HOSVD($\mathcal{X}, \mathbf{r}$)
  repeat
    for $n = 1, \ldots, N$ **do**
      $\mathcal{A} \leftarrow \mathcal{X} \times \mathbf{U}_1^T \times \cdots \times \mathbf{U}_N^T$
      $(\mathbf{U}_n, \cdot, \cdot) \leftarrow \mathrm{SVD}_{r_n}(\mathbf{A}^{(n)})$
    end for
  until Maximum iteration is reached or convergence
  $\mathcal{S} \leftarrow \mathcal{X} \times_1 \mathbf{U}_1^\top \times \cdots \times_N \mathbf{U}_N^\top$
  $\hat{\mathcal{X}} \leftarrow \mathcal{S} \times_1 \mathbf{U}_1 \times \cdots \times \mathbf{U}_N$
  return $(\mathcal{S}, \mathbf{U}_1, \ldots, \mathbf{U}_N, \hat{\mathcal{X}})$
end function

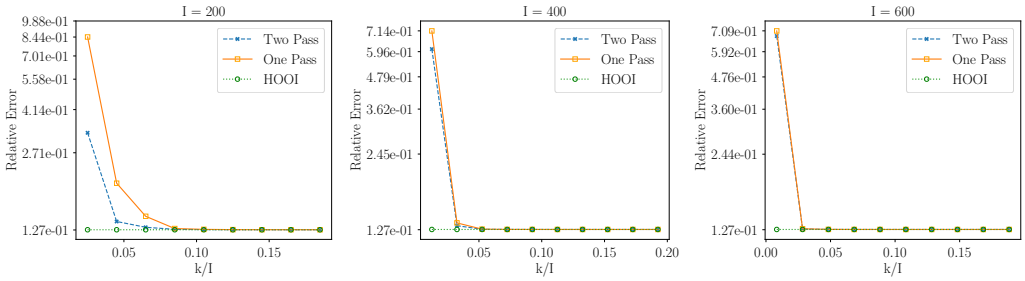
Algorithm 7 Two Pass Low-Rank Approximation

function TWOPASSLOWRANKRECOVERY($\mathbf{G}_1, \ldots \mathbf{G}_N, \mathcal{X}$)
  for $n = 1 \ldots N$ **do**
    $(\mathbf{Q}_n, \sim) \leftarrow \mathrm{QR}(\mathbf{G}_\mathrm{n})$
  end for
  $\hat{\mathcal{X}} \leftarrow \mathcal{X} \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N\mathbf{Q}_N^\top$
  return $\hat{\mathcal{X}}$
end function

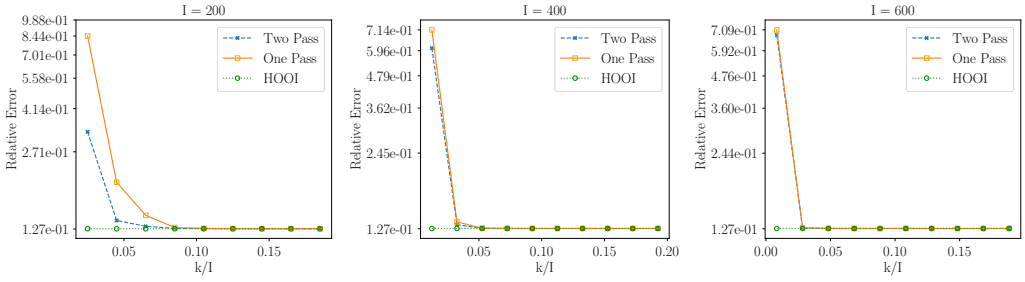
Algorithm 8 Two Pass Fixed-Rank Approximation

function TWOPASSFIXRANKRECOVERY($\mathbf{G}_1, \ldots \mathbf{G}_N, \mathcal{X}, \mathbf{r}$)
  for $n = 1 \ldots N$ **do**
    $(\mathbf{Q}_n, \sim) \leftarrow \mathrm{QR}(\mathbf{G}_\mathrm{n})$
  end for
  $\mathcal{W} \leftarrow \mathcal{X} \times_1 \mathbf{Q}^\top{}_1 \times \cdots \times_N \mathbf{Q}_N^\top$
  $\hat{\mathcal{X}} \leftarrow [\![\mathcal{W}]\!]_\mathbf{r} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N$
  return $\hat{\mathcal{X}}$
end function

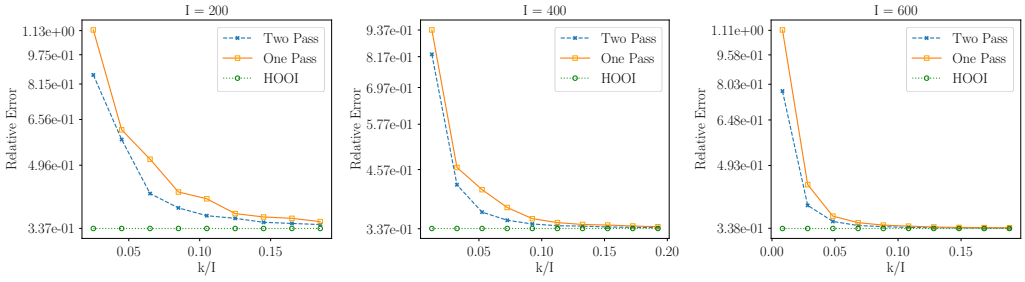
Appendix G More Simulation Results



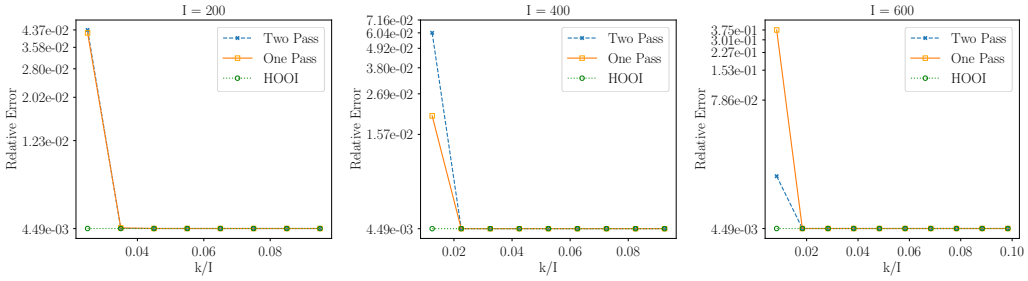
Superdiagonal + Low Noise ($\gamma = 0.01$, $r = 1$)



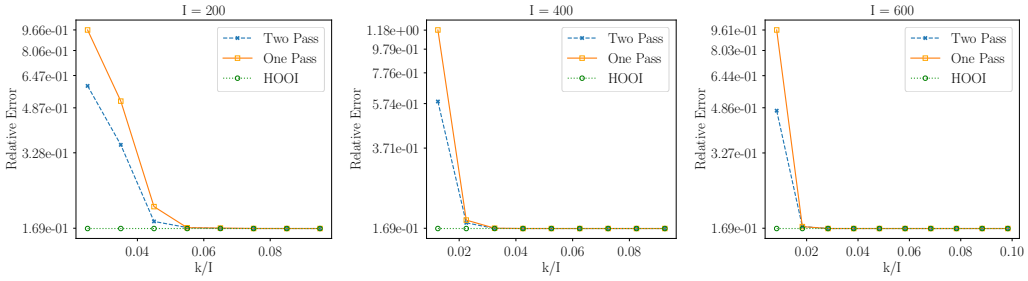
Fast Polynomial Decay ($t = 2$)



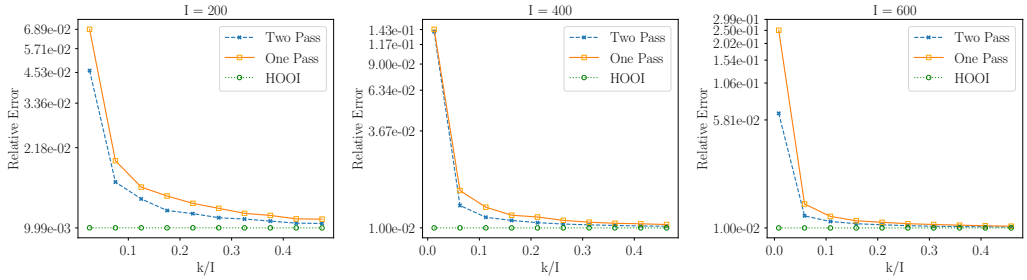
Slow Polynomial Decay ($t = 1$)



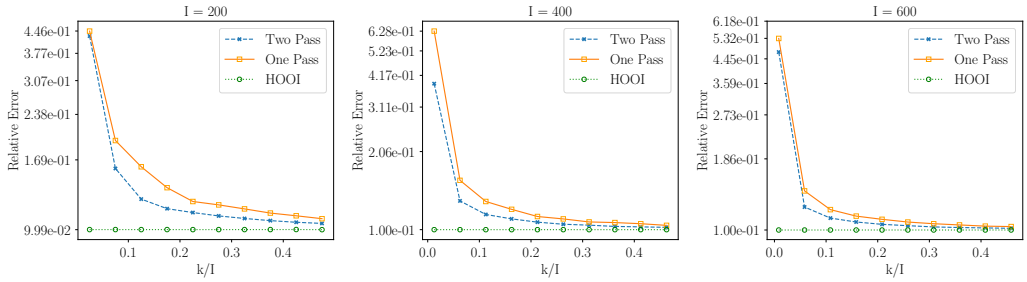
Fast Exponential Decay ($t = 1$)



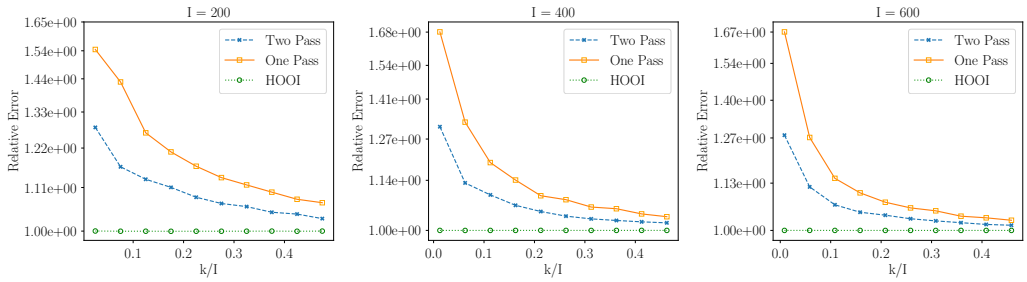
Slow Exponential Decay ($t = 0.25$)



Low Rank + Low Noise ($\gamma = 0.01$)



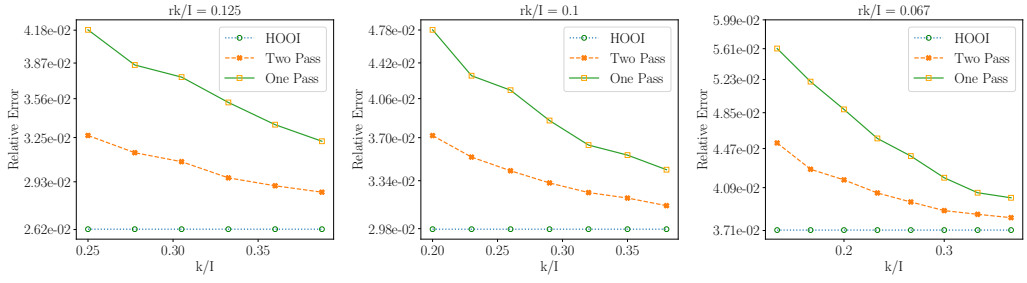
Low Rank + Medium Noise ($\gamma = 0.1$)



Low Rank + High Noise ($\gamma = 1$)

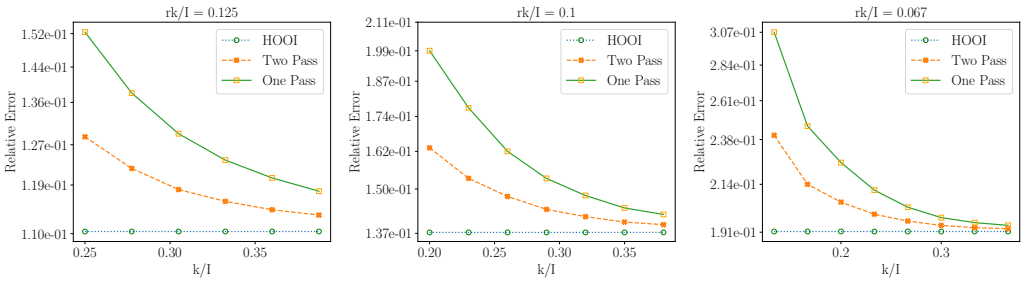
Figure 3: Relative error for fixed-rank tensor approximation as a function of the compression factor k/n: *We compare the relative error presented in log scale for two-pass sketching, one-pass sketching and Tucker decomposition for different design tensors when n = 200, 400, 600.* $r = 5$ *except for the first row.*

Appendix H More Real Data Results



Net Radiative Flux at Surface

Figure 4: Relative error for fixed-rank tensor approximation on the net radiative flux data ($1200 \times 192 \times 288$): *We compare the relative errors presented in log scale for two-pass sketching, one-pass sketching and Tucker decomposition with different ranks* ($rk/I = 0.125, 0.2, 0.067$)*. The dataset comes from the CESM CAM and the net radiative flux determines the energy received by the earth surface through radiation*



Dust Aerosol Burden

Figure 5: Relative error for fixed-rank tensor approximation on the dust aerosol burden data ($1200 \times 192 \times 288$): *We compare the relative errors presented in log scale for two-pass sketching, one-pass sketching and Tucker decomposition with different ranks ($rk/I = 0.125, 0.2, 0.067$). The dataset comes from the CESM CAM and the dust aerosol burden measures the amount of aerosol contributed by the dust.*

Appendix I Computational Complexity

Flops for Batch In the sketching stage of the two-pass algorithms, we need to first compute the arm sketches, $\mathbf{G}_n = \mathbf{X}\mathbf{\Omega}_n, n \in [N]$ with $kN\hat{I}$ flops in total. Then, in the recovery stage, we first perform "economy size" QR factorizations on $\mathbf{G}_1, \ldots, \mathbf{G}_N$ with $\mathcal{O}(k^2(\sum_{n=1}^{N} I_n))$ to find the orthonormal bases $\mathbf{Q}_1, \ldots, \mathbf{Q}_N$. Then, we compute the linkage tensor $\mathcal{W}$ by recursively multiplying $\mathcal{X}$ by $\mathbf{Q}_n^\top, n \in [N]$, with $(k \cdot I_1 \cdot I_{(-1)} + k \cdot I_2 \cdot \frac{I_{(-2)} \cdot k}{I_1} + \cdots + k^N \cdot I_N)$ flops, which is bounded by $\mathcal{O}(\frac{k(1-\delta_1^N)\bar{I}}{1-\delta_1})$. The multiplication step together with the arm sketch computation dominates the total cost as in Table 1.

In the sketching case, the one-pass algorithm needs to additionally compute the core tensor sketch $\mathcal{Z}$ by recursively multiplying $\mathcal{X}$ by $\mathbf{\Phi}_n, n \in [N]$. We can find the upper bound for the number of flops to be $\frac{s(1-\delta_1^N)}{1-\delta_1}\bar{I}$. Finding the orthonormal basis of the arm sketch takes $\mathcal{O}(k^2(\sum_{n=1}^{N} I_n))$ similar to the two-pass algorithm. To find the linkage tensor $\mathcal{W}$, we need to recursively solve linear square problems, with $\frac{k^2 s^N (1-(k/s)^N)}{1-k/s}$ flops. The sketch computation dominates the total time complexity, which is higher than the total time cost of the two-pass algorithm with a different compression factor.

The higher order SVD directly acts on $\mathcal{X}$ by first computing the SVD for each unfolding in $\mathcal{O}(kN\bar{I})$, and then multiplying $\mathcal{X}$ by $\mathbf{U}_1^\top, \ldots, \mathbf{U}_N^\top$ in $\mathcal{O}(\frac{k(1-\delta_1^N)\bar{I}}{1-\delta_1})$. The total time cost is approximately the same as the two-pass algorithm. Note: we can use the randomized SVD in the first step to improve the computational cost to $\bar{I}N \log k + \sum_{n=1}^{N}(I_n + I_{(-n)})k^2$ [10].

Flops for Streaming Using the elementwise representation of the mode product, we can significantly simplify the computation for sparse input tensor in the streaming model. In computing the arm sketches for both one-pass and two-pass algorithm, the complexity goes from $kN\bar{I}$ to $\mu kN\bar{I}$. Furthermore, we can reduce the computational complexity of multiplying $\mathcal{X}$ by $\mathbf{Q}_1^T, \ldots, \mathbf{Q}_N^T$ from $\mathcal{O}(\frac{k(1-\delta_1^N)\bar{I}}{1-\delta_1})$ to $\mathcal{O}(\mu k^N N(\sum_{n=1}^{N} I_n))$ in the recovery stage of the two-pass algorithm. Similarly, we can reduce the complexity of multiplying $\mathcal{X}$ by $\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_N$ from $\frac{s(1-\delta_2^N)\bar{I}}{1-\delta_2}$ to $\mu s^N N(\sum_{n=1}^{N} I_n)$ in the sketching phase of one-pass algorithm. The higher order SVD, in comparison, does not support the streaming model.

Appendix J Review of Tensor Algorithmic Operators

We review some basic tensor algorithmic operators first. Let $\mathcal{X}$ be the tensor of shape $I_1 \times I_2 \times \cdots \times I_N$. For tensor product with distinct modes ($m \neq n$), the order of mode products does not matter:

$$\mathcal{X} \times_m \mathbf{A} \times_n \mathbf{B} = \mathcal{X} \times_n \mathbf{B} \times_m \mathbf{A}. \tag{J.1}$$

If the modes are the same, then

$$\mathcal{X} \times_n \mathbf{A} \times_n \mathbf{B} = \mathcal{X} \times_n (\mathbf{BA}). \tag{J.2}$$

It is worth noting that the inner product for tensors is the same with respect to any mode-n matricization, with its definition given by

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \langle \mathbf{X}^n, \mathbf{Y}^{(n)} \rangle = \mathrm{Tr}((\mathbf{X}^{(\mathrm{n})})^\top \mathbf{Y}^{(\mathrm{n})}). \tag{J.3}$$