LOW-RANK TUCKER APPROXIMATION OF A TENSOR FROM STREAMING DATA *

YIMING SUN†, YANG GUO‡, CHARLENE LUO§, JOEL TROPP¶, AND MADELEINE UDELL†

Abstract. This paper describes a new algorithm for computing a low-Tucker-rank approximation of a tensor. The method applies a randomized linear map to the tensor to obtain a *sketch* that captures the important directions within each mode, as well as the interactions among the modes. The sketch can be extracted from streaming or distributed data or with a single pass over the tensor, and it uses storage proportional to the degrees of freedom in the output Tucker approximation. The algorithm does not require a second pass over the tensor, although it can exploit another view to compute a superior approximation. The paper provides a rigorous theoretical guarantee on the approximation error. Extensive numerical experiments show that that the algorithm produces useful results that improve on the state of the art for streaming Tucker decomposition.

Key words. Tucker decomposition, tensor compression, dimension reduction, sketching method, randomized algorithm, streaming algorithm

AMS subject classifications. 68Q25, 68R10, 68U05

1. Introduction.

Large-scale datasets with natural tensor (multidimensional array) structure arise in a wide variety of applications including computer vision [38], neuroscience [10], scientific simulation [4], sensor networks [29], and data mining [21]. In many cases, these tensors are too large to manipulate, to transmit, or even to store in a single machine. Luckily, tensors often exhibit a low-rank structure, and can be approximated by a low-rank tensor factorization, such as CANDECOMP/PARAFAC (CP), tensor train, or Tucker factorization [20]. These factorizations reduce the storage costs by exposing the latent structure. Sufficiently low rank tensors can be compressed by several orders of magnitude with negligible loss. However, computing these factorizations can require substantial computational resources. Indeed, one particular challenge is that these large tensors may not fit in main memory on our computer.

In this paper, we develop a new algorithm to compute a low-rank Tucker approximation for a tensor from streaming data, using storage proportional to the degrees of freedom in the output Tucker approximation. The algorithm forms a linear sketch of the tensor, and operates on the sketch to compute a low-rank Tucker approximation. Importantly, the main computational work is all performed on a small tensor, of size proportional to the core tensor of the Tucker factorization. We derive detailed probabilistic error bounds on the quality of the approximation in terms of the tail energy of any matricization of the target tensor.

This algorithm is useful in at least three concrete problem settings:

1. **Streaming:** Data from the tensor is generated sequentially. At each time stamp, we may observe a low dimensional slice, an individual entry, or an additive update to the tensor (the so-called "turnstile" model [26]). For example, each slice of the tensor may represent a subsequent time step in a simulation, or sensor measurements at a particular time. In the streaming

*Submitted to the editors DATE.

†Cornell University, Ithaca (ys784@cornell.edu, udell@cornell.edu).
‡University of Wisconsin-Madison, Madison, WI (yguo@cs.wisc.edu).
§Columbia University, New York, NY (cl3788@columbia.edu).
¶California Institute of Technology, Pasadena, CA (jtropp@cms.caltech.edu)

setting, the complete tensor is not stored; indeed, it may be much larger than available computing resources.
Our algorithm can approximate tensors revealed via streaming updates by sketching the updates and storing the sketch. Linearity of the sketch guarantees that sketching commutes with slice, entrywise, or additive updates. Our method forms an approximation of the tensor only after all the data has been observed, rather than approximating the tensor-observed-so-far at any time. This protocol allows for offline data analysis, including many scientific applications. Conversely, this protocol is not suitable for real-time monitoring.

2. **Limited memory:** Data describing the tensor is stored on a hard disk of a computer with much smaller RAM. This setting reduces to the streaming setting by streaming the data from disk.
3. **Distributed:** Data describing the tensor may be stored on many different machines. Communicating data between these machines may be costly due to low network bandwidth or high latency. Our algorithm can approximate tensors stored in a distributed computing enviroment by sketching the data on each slave machine and transmitting the sketch to a master, which computes the sum of the sketches. Linearity of the sketch guarantees that the sum of the sketches is the sketch of the full tensor.

In the streaming setting, the tensor is not stored, so we require an algorithm that can compute an approximation from a single pass over the data. In constrast, multiple passes over the data are possible in the memory-limited or distributed settings.

This paper presents algorithms for all these settings, among other contributions:

- We present a new method to form a linear sketch of an unknown tensor. This sketch captures both the principal subspaces of the tensor along each mode, and the action of the tensor that links these subspaces. This tensor sketch can be formed from any dimension reduction map. Those sketches themselves are useful in many application even without forming Tucker decomposition like in video clustering.
- We develop a practical one-pass algorithm to compute a low rank Tucker approximation from streaming data. The algorithm sketches the tensor and then recovers a low rank Tucker approximation from this sketch.
- We propose a two-pass algorithm that improves on the one-pass method. Both the one-pass and two-pass methods are appropriate in a limited memory or distributed data setting.
- We develop provable probabilistic guarantees on the performance of both the one-pass and two-pass algorithms when the tensor sketch is composed of Gaussian dimension reduction maps.
- We exhibit several random maps that can be used to sketch the tensor. Compared to the Gaussian map, others are cheaper to store, easier to apply, and deliver similar performance experimentally in tensor approximation error. In particular, we demonstrate the effective performance of a row-product random matrix, which we call the Tensor Random Projection (TRP), which uses exceedingly low storage.
- We perform a comprehensive simulation study with synthetic data, and consider applications to several real datasets, to demonstrate the practical performance of our method. Our methods reduce approximation error compared to the only existing one-pass Tucker approximation algorithm [25] by more than an order of magnitude given the same storage budget.
- We have developed and released an open-source package in python that

implements our algorithms.

2. Background and Related Work.

2.1. Notation. Our paper follows the notation of [20]. We denote *scalar*, *vector*, *matrix*, and *tensor* variables respectively by lowercase letters (x), boldface lowercase letters ($\mathbf{x}$), boldface capital letters ($\mathbf{X}$), and boldface Euler script letters ($\boldsymbol{\mathcal{X}}$). For two vectors $\mathbf{x}$ and $\mathbf{y}$, we write $\mathbf{x} > \mathbf{y}$ if $\mathbf{x}$ is greater than $\mathbf{y}$ elementwise and $\mathbf{x} \prec \mathbf{y}$ as the opposite.

Define $[N] := \{1, \ldots, N\}$. For a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, we denote its i^{th} row, j^{th} column, and $(i,j)^{th}$ element as $\mathbf{X}(i,.)$, $\mathbf{X}(.,j)$, and $\mathbf{X}(i,j)$, respectively, for $i \in [m]$, $j \in [n]$. We use $\mathbf{X}^\dagger \in \mathbb{R}^{n \times m}$ to denote the *Moore–Penrose pseudoinverse* of the matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$. In particular, $\mathbf{X}^\dagger = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^T$ if $m \geqslant n$ 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.

2.1.1. Tail energy. To state our results, we will need a tensor equivalent for the decay in the spectrum of a matrix. For each unfolding $\mathbf{X}^{(n)}$, define the *ρth tail energy*

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

where $\sigma_k(\mathbf{X}^{(n)})$ is the kth largest singular value of $\mathbf{X}^{(n)}$.

2.1.2. Kronecker and Khatri-Rao product. For two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$, we define the *Kronecker product* $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{IK \times JL}$ as

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} \mathbf{A}(1,1)\mathbf{B} & \cdots & \mathbf{A}(1,J)\mathbf{B} \\ \vdots & \ddots & \vdots \\ \mathbf{A}(I,1)\mathbf{B} & \cdots & \mathbf{A}(I,J)\mathbf{B} \end{bmatrix}. \tag{2.1}$$

For $J = L$, we define the *Khatri-Rao product* as $\mathbf{A} \odot \mathbf{B}$, i.e. the "matching column-wise" Kronecker product. The resulting matrix of size $(IJ) \times K$ is defined as

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{A}(\cdot,1) \otimes \mathbf{B}(\cdot,1) \cdots \mathbf{A}(\cdot,K) \otimes \mathbf{B}(\cdot,K)]$$

2.1.3. Tensor basics. For a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, its *mode* or *order* is the number N of dimensions. 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 $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}$ is defined as $\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} \boldsymbol{\mathcal{X}}_{i_1 \ldots i_N} \boldsymbol{\mathcal{Y}}_{i_1 \ldots i_N}$. The *Frobenius norm* of $\boldsymbol{\mathcal{X}}$ is $\|\boldsymbol{\mathcal{X}}\|_F = \sqrt{\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle}$.

2.1.4. Tensor unfoldings. Let $\bar{I} = \Pi_{j=1}^N I_j$ and $I_{(-n)} = \Pi_{j \neq n} I_j$, and let $\mathbf{vec}(\boldsymbol{\mathcal{X}})$ denote the vectorization of $\boldsymbol{\mathcal{X}}$. The *mode-n unfolding* of $\boldsymbol{\mathcal{X}}$ is the matrix $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times I_{(-n)}}$. The inner product for tensors matches that of any mode-n unfolding:

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \langle \mathbf{X}^{(n)}, \mathbf{Y}^{(n)} \rangle = \mathrm{Tr}((\mathbf{X}^{(n)})^\top \mathbf{Y}^{(n)}). \tag{2.2}$$

2.1.5. Tensor rank. The *mode-n rank* is the rank of the mode-n unfolding. We say the *rank* of $\boldsymbol{\mathcal{X}}$ is $\mathbf{r}(\boldsymbol{\mathcal{X}}) = (r_1, \ldots, r_N)$ if its *mode-n rank* is r_n for each $n \in [N]$. This notion of rank corresponds to the size of the core tensor in a Tucker factorization of $\boldsymbol{\mathcal{X}}$. A *superdiagonal* tensor generalizes a diagonal matrix: all entries are zero except for the entries whose indices in each dimension are equal.

2.1.6. Tensor contractions. Write $\mathcal{G} = \mathcal{X} \times_n \mathbf{U}$ for the *mode-n (matrix) product* of $\mathcal{X}$ with $\mathbf{U} \in \mathbb{R}^{J \times I_n}$. That is, $\mathcal{G} = \mathcal{X} \times_n \mathbf{U} \iff \mathbf{G}^{(n)} = \mathbf{U}\mathbf{X}^{(n)}$. The tensor $\mathcal{G}$ has dimension $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$. Mode products with respect to different modes commute: for $\mathbf{U} \in \mathbb{R}^{J_1 \times I_n}$, $\mathbf{V} \in \mathbb{R}^{J_2 \times I_m}$

$$\mathcal{X} \times_n \mathbf{U} \times_m \mathbf{V} = \mathcal{X} \times_m \mathbf{V} \times_n \mathbf{U} \quad \text{if} \quad n \neq m.$$

Mode products along the same mode simplify: for $\mathcal{A} \in \mathbb{R}^{J_1 \times I_n}$, $\mathcal{B} \in \mathbb{R}^{J_2 \times J_1}$,

$$\mathcal{X} \times_n \mathbf{A} \times_n \mathbf{B} = \mathcal{X} \times_n (\mathbf{B}\mathbf{A}).$$

2.2. Tucker Approximation. 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 idea of Tucker approximation is finding a *core tensor* $\mathcal{G} \in \mathbb{R}^{r_1 \times \cdots \times r_N}$ and matrices with orthogonal columns $\mathbf{U}_n \in \mathbb{R}^{I_n \times r_n}$ for $n \in [N]$, called *factor matrices*, so that

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

For brevity, we define $[\![\mathcal{G}; \mathbf{U}_1, \cdots, \mathbf{U}_N]\!] = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \cdots \times_N \mathbf{U}_N$. Any best rank-$\mathbf{r}$ Tucker approximation is of the form $[\![\mathcal{G}^\star; \mathbf{U}_1^\star, \cdots, \mathbf{U}_N^\star]\!]$, where $\mathcal{G}^\star, \mathbf{U}_n^\star$ solve the problem

$$\begin{array}{ll} \text{minimize} & \|\mathcal{X} - \mathcal{G} \times_1 \times \ldots \mathbf{U}_{n+1} \times_N \mathbf{U}_N\|_F^2 \\ \text{subject to} & \mathbf{U}_n^\top \mathbf{U}_n = \mathbf{I}. \end{array} \tag{2.3}$$

The problem 2.3 is a challenging nonconvex optimization problem. Moreover, the solution is not unique [20]. We use the notation $[\![\mathcal{X}]\!]_\mathbf{r}$ to represent a best rank-$\mathbf{r}$ Tucker approximation of the tensor $\mathcal{X}$, which in general we cannot compute.

2.2.1. HOSVD. The standard approach to computing a rank $\mathbf{r} = (r_1, \ldots, r_N)$ Tucker approximation for a tensor $\mathcal{X}$ begins with the higher order singular value decomposition (HOSVD) [13, 36], (2.1):

Algorithm 2.1 Higher order singular value decomposition (HOSVD) [13, 36]

Given: tensor $\mathcal{X}$, target rank $\mathbf{r} = (r_1, \ldots, r_N)$

1. *Factors.* For $n \in [N]$, compute the top r_n left singular vectors $\mathbf{??}_n$ of $\mathbf{??}^{(n)}$.
2. *Core.* Contract these with $\mathcal{X}$ to form the core

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{??}_1^T \cdots \times_N \mathbf{??}_N^T.$$

Return: Tucker approximation $\mathcal{X}_{\text{HOSVD}} = [\![\mathcal{G}; \mathbf{??}_1, \ldots, \mathbf{??}_N]\!]$

The HOSVD can be computed in two passes over the tensor [42, 8]. We describe this method briefly here, and in more detail in the next section. In the first pass, sketch each matricization $\mathbf{??}^{(n)}$, $n \in [N]$, and use randomized linear algebra (e.g., the randomized range finder of [16]) to (approximately) recover its range $\mathbf{U}_n$. To form the core $\mathcal{X} \times_1 \mathbf{??}_1^T \cdots \times_N \mathbf{??}_N^T$ requires a second pass over $\mathcal{X}$, since the factor matrices $\mathbf{??}_n$ depend on $\mathcal{X}$. The main algorithmic contribution of this paper is to develop a method to approximate both the factor matrices and the core in just one pass over $\mathcal{X}$.

2.2.2. ST-HOSVD. [37] proposes a sequentially truncated higher-order singular value decomposition which enjoys the same approximation compared to HOSVD, but requires less operators. The idea, is to update the low rank approximation of the target tensor $\mathcal{X}$ when we get factor matrix in Algorithm 2.1 for the following factor extraction. Readers can go to [37] for more details.

2.2.3. HOOI. The higher order orthogonal iteration (HOOI) [13] (2.2) improves on the resulting Tucker factorization by repeatedly minimizing the objective of 2.3 over the the core and the factor matrices. Notice the core update (2.5) admits the

Algorithm 2.2 Higher order orthogonal iteration (HOOI) [13]

Given: tensor $\mathcal{X}$, target rank $\mathbf{r} = (r_1, \dots, r_N)$
Initialize: compute $\mathcal{X} \approx [\![\mathcal{G}; \mathbf{U}_1, \dots, \mathbf{U}_N]\!]$ using HOSVD
Repeat:

1. *Factors.* For each $n \in [N]$,

$$\mathbf{U}_n \leftarrow \arg\min_{\mathbf{U_n}} \|[\![\mathcal{G}; \mathbf{U}_1, \dots, \mathbf{U}_N]\!] - \mathcal{X}\|_F^2 , \tag{2.4}$$

2. *Core.*

$$\mathcal{G} \leftarrow \arg\min_{\mathcal{G}} \|[\![\mathcal{G}; \mathbf{U}_1, \dots, \mathbf{U}_N]\!] - \mathcal{X}\|_F^2 . \tag{2.5}$$

Return: Tucker approximation $\mathcal{X}_{\text{HOOI}} = [\![\mathcal{G}; \mathbf{U}_1, \dots, \mathbf{U}_N]\!]$

closed form solution $\mathcal{G} \leftarrow \mathcal{X} \times_1 \mathbf{U}_1^\top \cdots \times_N \mathbf{U}_N^\top$, which motivates the second step of HOSVD.

2.3. Previous Work. The only previous work on streaming Tucker approximation is [25], which develops a streaming method called Tucker TensorSketch (T.-TS) [25, Algorithm 2]. T.-TS improves on HOOI by sketching the data matrix in the least squares problems. However, the success of the approach depends on the quality of the initial core and factor matrices, and the alternating least squares algorithm takes several iterations to converge.

In contrast, our work is motivated by HOSVD (not HOOI), and requires no initialization or iteration. We treat the tensor as a *multilinear* operator. The sketch identifies a low-dimensional subspace *for each input argument* that captures the action of the operator. The reconstruction produces a low-Tucker-rank multilinear operator with the same action on this low-dimensional tensor product space. This linear algebraic view allows us to develop the first guarantees on approximation error for this class of problems[1]. Moreover, we show in numerical experiments that our algorithm achieves a better approximation of the original tensor given the same memory resources.

More generally, there is a large literature on randomized algorithms for matrix factorizations and for solving optimization problems; see e.g. the review articles [16, 40]. In particular, our method is strongly motivated by the recent papers [33, 34], which provide methods for one-pass matrix approximation. The novelty of this paper is in our design of a core sketch (and reconstruction) for the Tucker decomposition, together with provable performance guarantees. The proof requires a careful accounting of the errors resulting from the factor sketches and from the core sketch. The structure of the Tucker sketch guarantees that these errors are independent.

Many researchers have used randomized algorithms to compute tensor decompositions. For example, [39, 7] apply sketching techniques to the CP decomposition, while

[1] The guarantees in [25] hold only when a new sketch is applied for each subsequent least squares solve; the resulting algorithm cannot be used in a streaming setting. In contrast, the practical streaming method T.-TS fixes the sketch for each mode, and so has no known guarantees. Interestingly, experiments in [25] show that the method achieves lower error using a fixed sketch (with no guarantees) than using fresh sketches at each iteration.

[35] suggests sparsifying the tensor. Several papers aim to make Tucker decomposition efficient in the limited-memory or distributed settings [6, 42, 4, 19, 23, 8].

3. Dimension Reduction Maps. In this section, we first introduce some commonly used randomized dimension reduction maps together with some mathematical background, and explain how to calculate and update sketches.

3.1. Dimension Reduction Map. Dimension reduction maps (DRMs) take a collection of high dimensional objects to a lower dimensional space while preserving certain geometric properties [27]. For example, we may wish to preserve the pairwise distances between vectors, or to preserve the column space of matrices. We call the output of a DRM on an object x a *sketch* of x.

Some common DRMs include matrices with i.i.d. Gaussian entries or i.i.d. ± 1 entries. The Scrambled Subsampled Randomized Fourier Transform (SSRFT) [41] and sparse random projections [1, 24] can achieve similar performance with fewer computational and storage requirements; see F for details.

Our theoretical bounds rely on properties of the Gaussian DRM. However, our numerical experiments indicate that many other DRMs yield qualitatively similar results; see, e.g., Figure 1, Figure 10 and Figure 9) in Figure I.

3.2. Tensor Random Projection. Here we present a strategy for reducing the storage of the random map that makes use of the tensor random projection (TRP), and extremely low storage structured dimension reduction map proposed in [30]. The *tensor random projection (TRP)* $\mathbf{\Omega} : \prod_{n=1}^{N} I_n \to \mathbb{R}^k$ is defined as the iterated Khatri-Rao product of DRMS $\mathbf{A}_n \in \mathbb{R}^{I_n \times k}$, $n \in [N]$:

$$\mathbf{\Omega} = \mathbf{A}_1 \odot \cdots \odot \mathbf{A}_N. \tag{3.1}$$

Each $\mathbf{A}_n \in \mathbb{R}^{I_n \times k}$ can be a Gaussian map, sign random projection, SSRFT, etc. The number of constituent maps N and their dimensions I_n for $n \in [N]$ are parameters of the TRP, and control the quality of the map; see [30] for details. The TRP map is a row-product random matrix, which behaves like a Gaussian map in many respects [28]. Our experimental results confirm this behavior.

Supposing each I_n is the same for $n \in [N]$, the TRP can be formed (and stored) using only kNI random variables, while standard dimension reduction maps use randomness (and storage) that grows as I^N when applied to a generic (dense) tensor. 1 compares the computational and storage costs for different DRMs.

	Storage Cost	Computation Cost
Gaussian	kI^N	kI^N
Sparse	μkI^N	μkI^N
SSRFT	I^N	$I^N \log(k)$
TRP	kNI	kI^N

Table 1: Performance of Different Dimension Reduction Maps: We compare the storage cost and the computational cost of applying a DRM mapping $\mathbb{R}^{I^N}$ to $\mathbb{R}^k$ to a dense tensor in $\mathbb{R}^{I^N}$. Here μ is the sparse factor for sparse random projection. The TRP considered here is composed of Gaussian DRMs.

We do not need to explicitly form or store the TRP map $\mathbf{\Omega}$. Instead, we can store its constituent DRMs $\mathbf{A}_1, \ldots, \mathbf{A}_N$ and compute the action of the map on the

matricized tensor using the definition of the TRP. The additional computation required is minimal and empirically incurs almost no performance loss.

4. Algorithms for Tucker approximation. In this section, we present our proposed tensor sketch and algorithms for one- and two-pass Tucker approximation, and discuss the computational complexity and storage cost of these methods for both sparse and dense input tensors. We present guarantees for these methods in 5.

4.1. Tensor compression via sketching.

The Tucker sketch. Our Tucker sketch generalizes the matrix sketch of [33] to higher order tensors. To compute a Tucker sketch for tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ with sketch size parameters $\mathbf{k}$ and $\mathbf{s}$, draw independent, random DRMs

$$\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2, \ldots, \boldsymbol{\Omega}_N \quad \text{and} \quad \boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \ldots, \boldsymbol{\Phi}_N, \tag{4.1}$$

with $\boldsymbol{\Omega}_n \in \mathbb{R}^{I_{(-n)} \times k_n}$ and $\boldsymbol{\Phi}_n \in \mathbb{R}^{I_n \times s_n}$ for $n \in [N]$. Use these DRMs to compute

$$\begin{aligned} \mathbf{V}_n &= \mathbf{X}^{(n)} \boldsymbol{\Omega}_n && \in \mathbb{R}^{I_n \times k_n}, \quad n \in [N], \\ \mathcal{H} &= \mathcal{X} \times_1 \boldsymbol{\Phi}_1^\top \cdots \times_N \boldsymbol{\Phi}_N^\top && \in \mathbb{R}^{s_1 \times \cdots \times s_N}. \end{aligned}$$

The *factor sketch* $\mathbf{V}_n$ captures the span of the mode-n fibers of $\mathcal{X}$ for each $n \in [N]$, while the *core sketch* $\mathcal{H}$ contains information about the interaction between different modes. See 4.1 for pseudocode.

To produce a rank $\mathbf{r} = \{r_1, \ldots, r_N\}$ Tucker approximation of $\mathcal{X}$, choose sketch size parameters $\mathbf{k} = (k_1, \ldots, k_N) \succeq \mathbf{r}$ and $\mathbf{s} = (s_1, \ldots, s_N) \preceq \mathbf{k}$. (Vector inequalities hold elementwise.) Our approximation guarantees depend closely on the parameters $\mathbf{k}$ and $\mathbf{s}$. As a rule of thumb, we suggest selecting $\mathbf{s} = 2\mathbf{k} + 1$, as the theory requires $\mathbf{s} > 2\mathbf{k}$, and choosing $\mathbf{k}$ as large as possible given storage limitations.

The sketches $\mathbf{V}_n$ and $\mathcal{H}$ are linear functions of the original tensor, so we can compute the sketches in a single pass over the tensor $\mathcal{X}$. Linearity enables easy computation of the sketch even in the streaming model (E.1) or distributed model (E.2). Storing the sketches requires memory $\sum_{n=1}^{N} I_n \cdot k_n + \Pi_{i=1}^{N} s_n$: much less than the full tensor.

Algorithm 4.1 Tucker Sketch

Given: RDRM (a function that generates a random DRM)

1: **function** TuckerSketch($\mathcal{X}; \mathbf{k}, \mathbf{s}$)
2: Form DRMs $\boldsymbol{\Omega}_n = \text{RDRM}(I_{(-n)}, k_n)$ and $\boldsymbol{\Phi}_n = \text{RDRM}(I_n, s_n)$, $n \in [N]$
3: Compute factor sketches $\mathbf{V}_n \leftarrow \mathbf{X}^{(n)} \boldsymbol{\Omega}_n$, $n \in [N]$
4: Compute core sketch $\mathcal{H} \leftarrow \mathcal{X} \times_1 \boldsymbol{\Phi}_1^\top \times \cdots \times_N \boldsymbol{\Phi}_N^\top$
5: **return** $(\mathcal{H}, \mathbf{V}_1, \ldots, \mathbf{V}_N, \{\boldsymbol{\Phi}_n, \boldsymbol{\Omega}_n\}_{n \in [N]})$
6: **end function**

Remark 4.1. The DRMs $\boldsymbol{\Omega}_n \in \mathbb{R}^{I_{(-n)} \times k_n}$ are large—much larger than the size of the Tucker factorization we seek! Even using a low memory mapping such as the SSRFT and sparse random map, the storage cost required grows as $\mathcal{O}(I_{(-n)})$. However, we do not need to store these matrices. Instead, we can generate (and regenerate) them as needed using a (stored) random seed.[2]

[2] Our theory assumes the DRMs are random, whereas our experiments use pseudorandom numbers. In fact, for many pseudorandom number generators it is NP hard to determine whether the output is random or pseudorandom [3]. In particular, we expect both to perform similarly for tensor approximation.

Remark 4.2. Alternatively, the TRP (3.2) can be used to limit the storage of $\mathbf{\Omega_n}$ required. The Khatri-Rao structure in the sketch need not match the structure in the matricized tensor. However, we can take advantage of the structure of our problem to reduce storage even further. We generate DRMs $\mathbf{A}_n \in \mathbb{R}^{I_n \times k}$ for $n \in [N]$ and define $\mathbf{\Omega}_n = \mathbf{A}_1 \odot \cdots \mathbf{A}_{n-1} \odot \mathbf{A}_{n+1} \odot \cdots \odot \mathbf{A}_N$ for each $n \in [N]$. Hence we need not store the maps $\mathbf{\Omega}_n$, but only the small matrices $\mathbf{A}_n$. The storage required is thereby reduced from $\mathcal{O}(N(\prod_{n=1}^N I_n)k)$ to $\mathcal{O}((\sum_{n=1}^N I_n)k)$, while the approximation error is essentially unchanged. We use this method in our experiments.

4.2. Low-Rank Approximation. Now we explain how to construct a Tucker decomposition of $\mathcal{X}$ with target Tucker rank $\mathbf{k}$ from the factor and core sketches.

We first present a simple two-pass algorithm, 4.2, that uses only the factor sketches by projecting the unfolded matrix of original tensor $\mathcal{X}$ to the column space of each factor sketch. To project to the column space of each factor matrix, we calculate the QR decomposition of each factor sketch:

$$\mathbf{V}_n = \mathbf{Q}_n \mathbf{R}_n \quad \text{for } n \in [N], \tag{4.2}$$

where $\mathbf{Q}_n \in \mathbb{R}^{I_n \times k_n}$ has orthonormal columns and $\mathbf{R}_n \in \mathbb{R}^{k_n \times k_n}$ is upper triangular. Consider the tensor approximation

$$\tilde{\mathcal{X}} = \mathcal{X} \times_1 \mathbf{Q}_1 \mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_N \mathbf{Q}_N^\top. \tag{4.3}$$

This approximation admits the guarantees stated in 5.1. Using the commutativity of the mode product between different modes, we can rewrite $\tilde{\mathcal{X}}$ as

$$\tilde{\mathcal{X}} = \underbrace{\left[\mathcal{X} \times \mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_N^\top\right]}_{\mathcal{W}_2} \times_1 \mathbf{Q}_1 \times_2 \cdots \times_N \mathbf{Q}_N = [\![\mathcal{W}_2; \mathbf{Q}_1, \ldots, \mathbf{Q}_N]\!], \tag{4.4}$$

which gives an explicit Tucker approximation $\tilde{\mathcal{X}}$ of our original tensor. The core approximation $\mathcal{W}_2 \in \mathbb{R}^{k_1 \times \cdots \times k_N}$ is much smaller than the original tensor $\mathcal{X}$. To compute this approximation, we need access to $\mathcal{X}$ twice: once to compute $\mathbf{Q}_1, \ldots, \mathbf{Q}_N$, and again to apply them to $\mathcal{X}$ in order to form $\mathcal{W}_2$.

Algorithm 4.2 Two Pass Sketch and Low Rank Recovery

Given: tensor $\mathcal{X}$, sketch parameters $\mathbf{k}$ and $\mathbf{s} \geqslant \mathbf{k}$

1. *Sketch.* $\left(\mathcal{H}, \mathbf{V}_1, \ldots, \mathbf{V}_N, \{\mathbf{\Phi}_n, \mathbf{\Omega}_n\}_{n \in [N]}\right) = \textsc{TuckerSketch}\,(\mathcal{X}; \mathbf{k}, \mathbf{s})$
2. *Recover factor matrices.* For $n \in [N]$, $(\mathbf{Q}_n, \sim) \leftarrow \mathrm{QR}(\mathbf{V}_n)$
3. *Recover core.* $\mathcal{W}_2 \leftarrow \mathcal{X} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N$

Return: Tucker approximation $\hat{\mathcal{X}}_2 = [\![\mathcal{W}_2; \mathbf{Q}_1, \ldots, \mathbf{Q}_N]\!]$ with rank $\preceq \mathbf{k}$

One-Pass Approximation. To develop a one-pass method, we must use the core sketch $\mathcal{H}$ — the compression of $\mathcal{X}$ using the random projections $\mathbf{\Phi}_n$ – to approximate $\mathcal{W}_2$ — the compression of $\mathcal{X}$ using random projections $\mathbf{Q}_n$. To develop intuition, consider the following calculation: if the factor matrix approximations $\mathbf{Q}_n$ capture the range of $\mathcal{X}$ well, then projection onto their ranges in each mode approximately preserves the action of $\mathcal{X}$:

$$\mathcal{X} \approx \mathcal{X} \times_1 \mathbf{Q}_1 \mathbf{Q}_1^\top \times \cdots \times_N \mathbf{Q}_N \mathbf{Q}_N^\top$$

Recall that for tensor $\mathcal{A}$, and matrix $\mathbf{B}$ and $\mathbf{C}$ with compatible sizes, $\mathcal{A} \times_n (\mathbf{BC}) = (\mathcal{A} \times_n \mathbf{C}) \times_n \mathbf{B}$. Use this rule to collect terms to recognize the two pass core approximation $\mathcal{W}_2$:

$$\mathcal{X} \approx \left(\mathcal{X} \times_1 \mathbf{Q}_1^\top \times \cdots \times_N \mathbf{Q}_N^\top\right) \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N = \mathcal{W}_2 \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N$$

Now contract both sides of this approximate equality with the DRMs $\mathbf{\Phi}_n$ and recognize the core sketch $\mathcal{H}$:

$$\mathcal{H} := \mathcal{X} \times_1 \mathbf{\Phi}_1^\top \cdots \times_N \mathbf{\Phi}_N^\top \approx \mathcal{W}_2 \times_1 \mathbf{\Phi}_1^\top \mathbf{Q}_1 \times \cdots \times_N \mathbf{\Phi}_N^\top \mathbf{Q}_N.$$

We have chosen $\mathbf{s} > \mathbf{k}$ so each $\mathbf{\Phi}_n^\top \mathbf{Q}_n$ has a left inverse with high probability. Hence we can solve the approximate equality for $\mathcal{W}_2$:

$$\mathcal{W}_2 \approx \mathcal{H} \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \times \cdots \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger =: \mathcal{W}_1.$$

The right hand side of the approximation defines the one pass core approximation $\mathcal{W}_1$. B.2 controls the error in this approximation.

4.3 summarizes the resulting one-pass algorithm. One (streaming) pass over the tensor can be used to sketch the tensor; to recover the tensor, we only access the sketches. 5.2 (below) bounds the overall quality of the approximation.

Algorithm 4.3 One Pass Sketch and Low Rank Recovery

Given: tensor $\mathcal{X}$, sketch parameters $\mathbf{k}$ and $\mathbf{s} \geqslant \mathbf{k}$

1. *Sketch.* $\left(\mathcal{H}, \mathbf{V}_1, \ldots, \mathbf{V}_N, \{\mathbf{\Phi}_n, \mathbf{\Omega}_n\}_{n\in[N]}\right) = \textsc{TuckerSketch}\,(\mathcal{X}; \mathbf{k}, \mathbf{s})$
2. *Recover factor matrices.* For $n \in [N]$, $(\mathbf{Q}_n, \sim) \leftarrow \mathrm{QR}(\mathbf{V}_n)$
3. *Recover core.* $\mathcal{W}_1 \leftarrow \mathcal{H} \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \times \cdots \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger$

Return: Tucker approximation $\hat{\mathcal{X}}_1 = [\![\mathcal{W}_1; \mathbf{Q}_1, \ldots, \mathbf{Q}_N]\!]$ with rank $\leqslant \mathbf{k}$

The time and storage cost of 4.3 is given by 2. The time and storage complexity of these methods compare favorably to the only previous method for streaming Tucker approximation [25]; see H for details.

	Stage	Time Cost	Storage Cost
4.3 (One Pass)	Sketching	$\mathcal{O}(((1-(s/I)^N)/(1-(s/I)) + Nk)I^N)$	$kNI + s^N$
	Recovery	$\mathcal{O}((k^2 s^N (1-(k/s)^N))/(1-k/s) + k^2 NI)$	
	Total	$\mathcal{O}(((s(1-(s/I)^N))/(1-s/I) + Nk)I^N)$	

Table 2: Computational Complexity of 4.3 on tensor $\mathcal{X} \in \mathbb{R}^{I\times\cdots\times I}$ with parameters (k, s), using a TRP composed of Gaussian DRMs inside the Tucker sketch. By far the majority of the time is spent sketching the tensor $\mathcal{X}$.

4.3. Fixed-Rank Approximation. Algorithm 4.2 and algorithm 4.3 produce a two-pass and one-pass rank-$\mathbf{k}$ tensor approximation respectively. It is often valuable to truncate this approximation to a user-specified target rank $\mathbf{r} \leqslant \mathbf{k}$ [34, Figure 4].

Our fixed rank approximation method is motivated by the following lemma:

LEMMA 4.1. *Let $\mathcal{W} \in \mathbb{R}^{k_1\times\cdots\times k_N}$ be a tensor, and let $\mathbf{Q}_n \in \mathbb{R}^{I_n \times k_n}$ be orthogonal matrices with $k_n \geqslant r_n$ for $n \in [N]$. Then*

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

(This lemma does not necessarily hold if the best rank-r Tucker approximation $[\![\cdot]\!]$ is replaced by the output of any concrete algorithm such as HOSVD or HOOI.) The proof of 4.1 appears in C.

Motivated by this lemma, to produce a fixed rank $\mathbf{r}$ approximation of $\mathcal{X}$, we compress the core tensor approximation from 4.2 or 4.3 to rank $\mathbf{r}$. This compression is cheap because the core approximation $\mathcal{W} \in \mathbb{R}^{k_1 \times \cdots \times k_N}$ is small. We present this method (using HOOI as the the compression algorithm) as 4.4. Other compression algorithms can be used to trade off the quality of approximation with the difficulty of running the algorithm. Reasonable choices include the sequentially-truncated HOSVD (ST-HOSVD) [37] or TTHRESH [5]. Both HOSVD and ST-HOSVD are psedual

Algorithm 4.4 Fixed rank approximation

Given: Tucker approximation $[\![\mathcal{W}; \mathbf{Q}_1, \cdots, \mathbf{Q}_N]\!]$ of tensor $\mathcal{X}$, rank target $\mathbf{r}$

1. *Approximate core with fixed rank.* $\mathcal{G}, \mathbf{U}_1, \cdots, \mathbf{U}_N \leftarrow \mathrm{HOOI}(\mathcal{W}, \mathbf{r})$
2. *Compute factor matrices.* For $n \in [N]$, $\mathbf{P}_n \leftarrow \mathbf{Q}_n \mathbf{U}_n$

Return: Tucker approximation $\hat{\mathcal{X}}_{\mathbf{r}} = [\![\mathcal{G}; \mathbf{P}_1, \ldots, \mathbf{P}_N]\!]$ with rank $\leqslant \mathbf{r}$

5. Guarantees. In this section, we present probabilistic guarantees on the preceding algorithms. We show that approximation error for the one-pass algorithm is the sum of the error from the two-pass algorithm and the error resulting from the core approximation. Proofs for the three theorems in this section can be found in the corresponding subsections of A.

5.1. Low rank approximation. 5.1 guarantees the performance of the two pass method 4.2.

THEOREM 5.1. *Sketch the tensor $\mathcal{X}$ using a Tucker sketch with parameters $\mathbf{k}$ using DRMs with i.i.d. Gaussian $\mathcal{N}(0,1)$ entries. Then the approximation $\hat{\mathcal{X}}_2$ computed with the two pass method 4.2 satisfies*

$$\mathbb{E}\|\mathcal{X} - \hat{\mathcal{X}}_2\|_F^2 \leqslant \min_{1 \leqslant \rho_n < k_n - 1} \sum_{n=1}^{N} \left(1 + \frac{\rho_n}{k_n - \rho_n - 1}\right) (\tau_{\rho_n}^{(n)})^2.$$

The two pass method does not use the core sketch, so this result does not depend on $\mathbf{s}$.

Theorem 5.2 guarantees the performance of one pass method 4.3.

THEOREM 5.2. *Sketch the tensor $\mathcal{X}$ using a Tucker sketch with parameters $\mathbf{k}$ and $\mathbf{s} > 2\mathbf{k}$ using DRMs with i.i.d. Gaussian $\mathcal{N}(0,1)$ entries. Then the approximation $\hat{\mathcal{X}}_1$ computed with the one pass method 4.3 satisfies the bound*

$$\mathbb{E}\|\mathcal{X} - \hat{\mathcal{X}}_1\|_F^2 \leqslant (1 + \Delta) \min_{1 \leqslant \rho_n < k_n - 1} \sum_{n=1}^{N} \left(1 + \frac{\rho_n}{k_n - \rho_n - 1}\right) (\tau_{\rho_n}^{(n)})^2,$$

where $\Delta := \max_{n=1}^{N} k_n/(s_n - k_n - 1)$.

The theorem shows that the method works best for tensors whose unfoldings exhibit spectral decay. As a simple consequence of this result, we see that the two pass method with $\mathbf{k} > \mathbf{r} + 1$ perfectly recovers a tensor with exact Tucker rank $\mathbf{r}$, since in that case $\tau_{r_n}^{(n)} = 0$ for each $n \in [N]$. However, this theorem states a stronger bound: the method exploits decay in the spectrum, wherever (in the first k_n singular values of each mode n unfolding) it occurs.

We see that the additional error due to sketching the core is a multiplicative factor Δ more than the error due to sketching the factor matrices. This factor Δ decreases as the size of the core sketch $\mathbf{s}$ increases.

Remark 5.1. Both HOSVD, ST-HOSVD achieves so called pseudo optimal with parameter $\sqrt{N}$. For example, for ST-HOSVD,

$$\|[\mathcal{X} - [\![\mathcal{X}]\!]_{\mathbf{ST-k}}\|_F \leqslant \sqrt{N}\|[\mathcal{X} - [\![\mathcal{X}]\!]_{\mathbf{k}}\|_F \leqslant \sqrt{\sum_{n=1}^{N} (\tau_{k_n}^{(n)})^2}, \tag{5.1}$$

where $[\![\mathcal{X}]\!]_{\mathbf{k}}$ is the optimal Tucker rank $\mathbf{k}$ approximation. Thus, the low rank approximation for two pass algorithm is pseduo optimal with factor $\sqrt{2N}$ while for one pass algorithm, if we choose $\mathbf{s} > 2\mathbf{k} + 1 (\Delta \leqslant 2)$, it is also pseduo optimal with factor $2\sqrt{N}$.

5.2 also offers guidance on how to select the sketch size parameters $\mathbf{s}$ and $\mathbf{k}$. In particular, suppose that the mode-n unfolding has a good rank r_n approximation for each mode n. Then the choices $k_n = 2r_n + 1$ and $s_n = 2k_n + 1$ ensure that

$$\mathbb{E}\|\mathcal{X} - \hat{\mathcal{X}}\|_F^2 \leqslant 4 \sum_{n=1}^{N} (\tau_{r_n}^{(n)})^2.$$

More generally, as k_n/r_n and s_n/k_n increase, the leading constant in the approximation error tends to one.

5.2. Fixed rank approximation. We now present a conditional analysis of the fixed rank approximation method given a low rank approximation. Recall that $[\![\cdot]\!]_{\mathbf{r}}$ returns a best rank-$\mathbf{r}$ Tucker approximation.

THEOREM 5.3. *Suppose* $\hat{\mathcal{X}} = [\![\mathcal{W}; \mathbf{Q}_1, \cdots, \mathbf{Q}_N]\!]$ *approximates the target tensor* $\mathcal{X}$*, and let* $\hat{\mathcal{X}}_{\mathbf{r}}$ *denote some rank* $\mathbf{r}$ *approximation to* $\hat{\mathcal{X}}$ *from some procedure like ST-HOSVD, Then for output any fix rank* $\mathbf{r}$

$$\mathbb{E}\|\mathcal{X} - \hat{\mathcal{X}}_{\mathbf{r}}\|_F \leqslant \|\mathcal{X} - [\![\mathcal{X}]\!]_{\mathbf{r}}\|_F + 2\sqrt{\mathbb{E}\|\mathcal{X} - \hat{\mathcal{X}}\|_F^2}.$$

The second term on the right-hand side of 5.3 is controlled by 5.1 and 5.2. Hence we can combine these results to provide guarantees for fixed rank approximation with either the two pass or one pass algorithms.

The resulting bound shows that the best rank-$\mathbf{r}$ approximation of the output from the one or two pass algorithms is comparable in quality to a true best rank-$\mathbf{r}$ approximation of the input tensor. An important insight is that the sketch size parameters $\mathbf{s}$ and $\mathbf{k}$ that guarantee a good low rank approximation also guarantee a good fixed rank approximation: the error due to sketching depends only on the sketch size parameters $\mathbf{k}$ and $\mathbf{s}$, and not on the target rank $\mathbf{r}$.

In practice, one would truncate the rank of the approximation using HOOI (4.4), rather than the best rank $\mathbf{r}$ approximation (**??**). Guarantees for resulting algorithm are beyond the scope of this paper, since there are no strong guarantees on the performance of HOOI; however, it is widely believed to produce an approximation that is usually quite close to the best rank $\mathbf{r}$ approximation.

5.3. Proof sketch. To bound the approximation error of the algorithms presented in the main body of this paper, we first develop several structural results showing an additive decomposition of the error. First, the total error is the sum of the error due to sketching and the error due to fixed rank approximation. Second, the sketching error is the sum of the error due to the factor matrix approximations and to the core approximation. Third, the error due to the factor matrix approximations is the sum of the error in each the modes, as the errors due to each mode are mutually orthogonal.

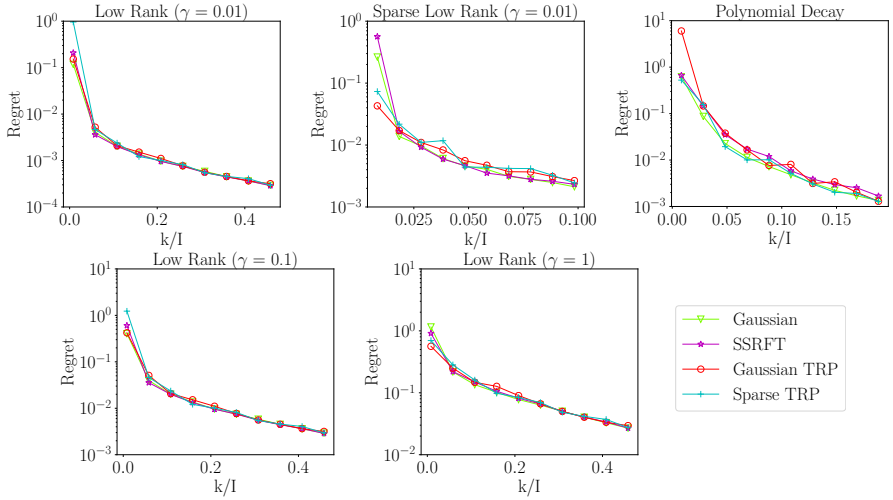


Fig. 1: *Different DRMs perform similarly.* We approximate 3D synthetic tensors (see 6.1) with $I = 600$, using our one-pass algorithm with $r = 5$ and varying k ($s = 2k + 1$), using a variety of DRMs in the Tucker sketch: Gaussian, SSRFT, Gaussian TRP, or Sparse TRP.

This finishes the approximation error bound for the two pass algorithm, 5.1. As for the error due to the core approximation, we rewrite the approximation error in the core tensor as a sum over each mode of errors that are mutually orthogonal. Indeed, these errors have the same form as the errors due to the factor matrix approximations, scaled down by a factor $\Delta(k, s)$ that depends on the sketch sizes $\mathbf{k}$ and $\mathbf{s}$. This argument shows the error due to the core approximation is at most a factor $\Delta(k, s)$ times the error due to the factor matrix approximation.

6. Numerical Experiments. In this section, we study the performance of our method. We compare the performance of the method using various different DRMs, including TRP. We also compare our method with the algorithm proposed by [25] to show that for the same storage budget, our method produces better approximations. Our two-pass algorithm outperforms the one-pass version, as expected. (Contrast this to [25], where the multi-pass method performs less well than the one-pass version.)

We evaluate the experimental results using two metrics:

$$\begin{aligned} &\text{normalized error:} && \|\mathcal{X} - \hat{\mathcal{X}}\|_F / \|\mathcal{X}\|_F \\ &\text{regret:} && \left(\|\mathcal{X} - \hat{\mathcal{X}}\|_F - \|\mathcal{X} - \mathcal{X}_{\text{HOOI}}\|_F\right) / \|\mathcal{X}\|_F. \end{aligned}$$

The normalized error measures the fraction of the energy in $\mathcal{X}$ captured by the approximation. The regret measures the increase in normalized error due to using the approximation $\hat{\mathcal{X}}$ rather than using $\mathcal{X}_{\text{HOOI}}$. The relative error measures the decrease in performance relative to HOOI. The normalized error of a rank $\mathbf{r}$ Tucker approximation $\hat{X}$ is always positive when $\mathcal{X}$ has a larger rank. In general, we find our proposed methods approaches the performance of HOOI for large enough storage budgets.

We ran all experiments on a server with 128 Intel® Xeon® E7-4850 v4 2.10GHz CPU cores and 1056GB memory. The code for our method is available at an anonymous Github repository https://github.com/tensorsketch/tensorsketch.

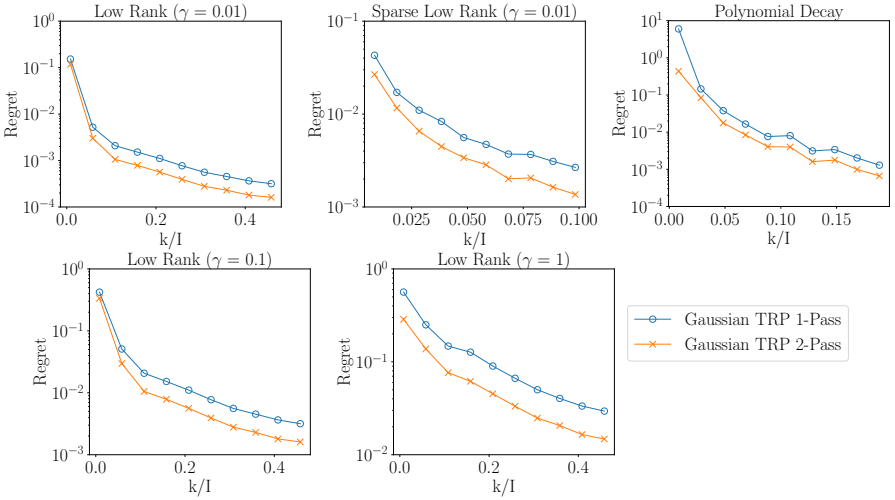


Fig. 2: *Two-pass improves on one-pass.* We approximate 3D synthetic tensors (see 6.1) with $I = 600$, using our one-pass and two-pass algorithms with $r = 5$ and varying k ($s = 2k + 1$), using the Gaussian TRP in the Tucker sketch.

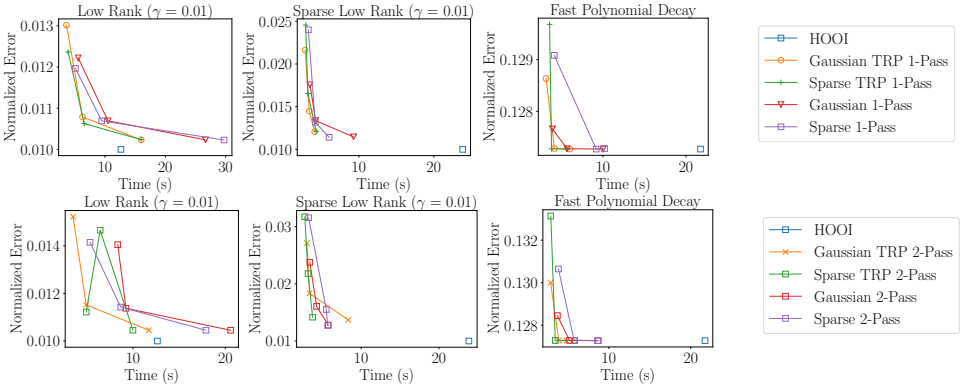


Fig. 3: *Faster approximations.* We approximate 3D synthetic tensors with $I = 600$ generated as described in 6.1, using HOOI and our one-pass and two-pass algorithms with $r = 5$ for a few different k ($s = 2k + 1$).

6.1. Synthetic experiments. All synthetic experiments use an input tensor with equal side lengths I. We consider three different data generation schemes:

- *Low rank + noise.* Generate a core tensor $\mathcal{G} \in \mathbb{R}^{r^N}$ with entries drawn from Unif($[0, 1]$). Independently generate N orthogonal factor matrices $\mathbf{A}_1, \ldots, \mathbf{A}_N \in \mathbb{R}^{r \times I}$. Define $\mathcal{X}^\natural = \mathcal{G} \times_1 \mathbf{A}_1 \cdots \times_N \mathbf{A}_N$ and the noise parameter $\gamma > 0$. Generate an input tensor as $\mathcal{X} = \mathcal{X}^\natural + (\gamma \|\mathcal{X}^\natural\|_F / I^{N/2})\boldsymbol{\epsilon}$ where the noise $\boldsymbol{\epsilon}$ has i.i.d. $\mathcal{N}(0, 1)$ entries.
- *Sparse low rank + noise.* We construct the input tensor $\mathcal{X}$ as above (Low Rank + Noise), but with sparse factor matrices $\mathbf{A}_n$: If δ_n is the sparsity (proportion of non-zero elements) of $\mathbf{A}_n$, then the sparsity of the true signal

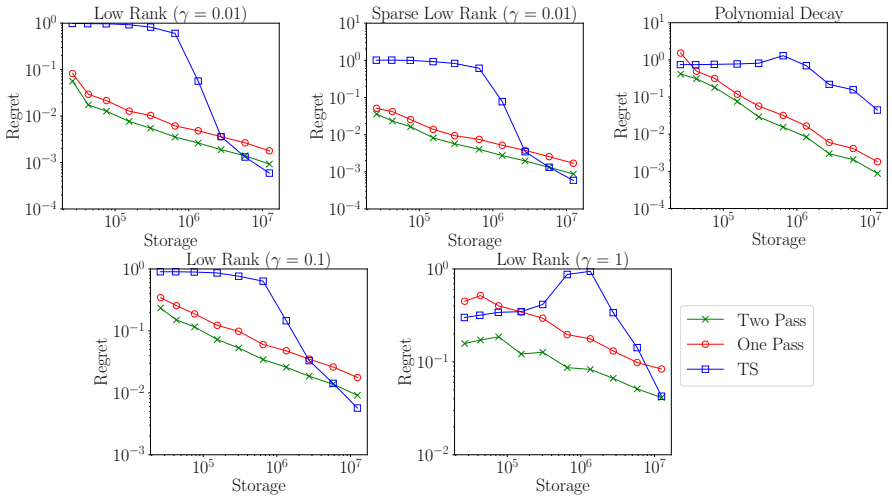


Fig. 4: *Approximations improve with more memory: synthetic data.* We approximate 3D synthetic tensors (see 6.1) with $I = 300$, using T.-TS and our one-pass and two-pass algorithms with the Gaussian TRP to produce approximations with equal ranks $r = 10$. Notice every marker on the plot corresponds to a 2700× compression!

$\mathbf{X}^\natural$ is $\prod_{n=1}^{N} \delta_n$. We use $\delta_n = 0.2$ unless otherwise specified.

- *Polynomial decay.* We construct the input tensor $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{superdiag}(1, \ldots, 1, 2^{-t}, 3^{-t}, \ldots, (I-r)^{-t}).$$

The first r entries are 1. Recall **superdiag** converts a vector to N dimensional superdiagonal tensor. Our experiments use $t = 1$ (geometric decay).

6.1.1. Different dimension reduction maps perform similarly. Our first experiment investigates the performance of our one-pass fixed-rank algorithm as the sketch size (and hence, required storage) varies, for several types of dimension reductions maps, including Gaussian, SSRFT, Gaussian TRP, and Sparse TRP. We generate synthetic data as described above with $\mathbf{r} = (5, 5, 5), I = 600$. 1 shows the rank-$\mathbf{r}$ approximation error as a function of the compression factor k/I. (Results for other input tensors are presented as 9 and 10 in I.) We see that the log relative error for our one-pass algorithm converges to that of HOOI as k increases for all input tensors. In the low rank case, the convergence rate is lower for higher noise levels. In general, the performance for different maps are approximately the same, although our theory only pertains to the Gaussian map.

We evaluate the run time for HOOI and our two algorithms with several different DRMs in 3. We can see that the one-pass algorithm is always slightly faster than the two-pass algorithm. The TRP generally provides a modest speedup in addition to the memory advantage. Both our one-pass and two-pass algorithms achieve nearly the accuracy of HOOI, and are usually much faster.

6.1.2. A second pass reduces error. The second experiment compares our two-pass and one-pass algorithm. The design is similar to the first experiment. 2 shows that the two-pass algorithm typically outperforms the one-pass algorithm, especially

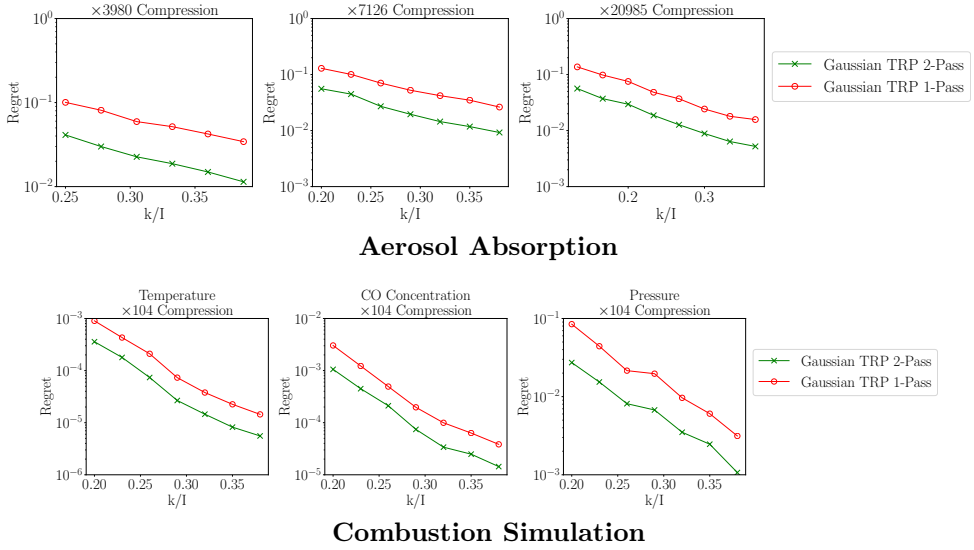


Fig. 5: *Approximations improves with more memory: real data.* We approximate aerosol absorption and combustion data using our one-pass and two-pass algorithms with the Gaussian TRP. We compare three target ranks ($r/I = 0.125, 0.1, 0.067$) for the former, and use the same target rank ($r/I = 0.1$) for each measured quantity in the combustion dataset. Notice $r/I = 0.1$ gives a hundred-fold compression!

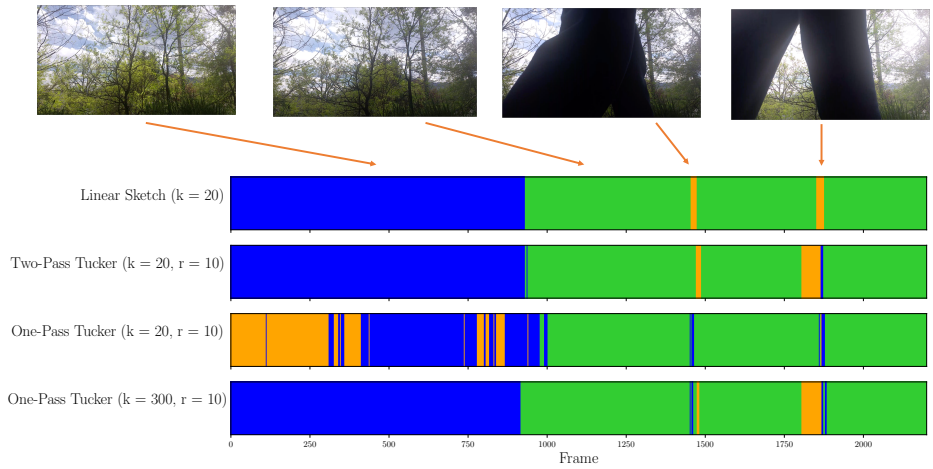
in the high-noise, sparse, or rank-decay case. Both converge at the same asymptotic rate. (Results for other input tensors are available in I.)

6.1.3. Improvement on state-of-the-art. The third experiment compares the performance of our two-pass and one-pass algorithms and Tucker TensorSketch (T.–TS), as described in [25], the only extant one-pass algorithm. For a fair comparison, we allocate the same storage budget to each algorithm and compare the relative error of the resulting fixed-rank approximations. We approximate synthetic 3D tensors with side length $I = 300$ with Tucker rank $r = 10$. We use the suggested parameter settings for each algorithm: $k = 2r$ and $s = 2k + 1$ for our methods; $K = 10$ for T.–TS. Our one-pass algorithm (with the Gaussian TRP) uses $((2k + 1)^N + kIN)$ storage, whereas T.-TS uses $(Kr^{2N} + Kr^{2N-2})$ storage (see 3 in H).

Figure 4 shows that our algorithms generally perform as well as T.–TS, and dramatically outperforms for small storage budgets. For example, our method achieves 1/50, 1/50, 1/7, and 1/4 the relative error of T.–TS for low rank and sparse low rank ($\gamma = 0.01$), low rank ($\gamma = 0.1$), and polynomial-decay input tensors, respectively. For the low rank ($\gamma = 1$) tensor, the performance of T.–TS is not even monotone as the storage budget increases! The performance of T.–TS is comparable with that of the algorithms presented in this paper only when the storage budget is large.

Remark 6.1. The paper [25] proposes a multi-pass method, Tucker Tensor-Times-Matrix-TensorSketch (TTMTS) that is dominated by the one-pass method Tucker TensorSketch(TS) in all numerical experiments; hence we compare only with T.-TS.

6.2. Applications. We also apply our method to datasets drawn from three application domains: climate, combustion, and video.



Video Scene Classification

Fig. 6: *Video Scene Classification* ($2200 \times 1080 \times 1980$): We classify frames from the video data from [25] (collected as a third order tensor with size $2200 \times 1080 \times 1980$) using K-means with K=3 on vectors computed using four different methods. $s = 2k+1$ throughout. 1) The linear sketch along the time dimension (Row 1). 2-3) the Tucker factor along the time dimension, computed via our two-pass (Row 2) and one-pass (Row 3) algorithms. 4) The Tucker factor along the time dimension, computed via our one-pass (Row 4) algorithm

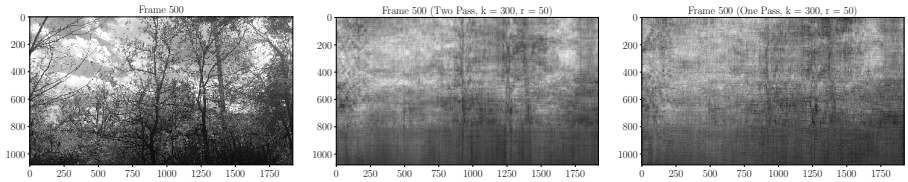


Fig. 7: *Visualizing Video Recovery:* Original frame (left); approximation by two-pass sketch (middle); approximation by one-pass sketch (right).

- *Climate data.* We consider global climate simulation datasets from the Community Earth System Model (CESM) Community Atmosphere Model (CAM) 5.0 [17, 18]. The dataset on aerosol absorption has four dimensions: times, altitudes, longitudes, and latitudes ($240 \times 30 \times 192 \times 288$). The data on net radiative flux at surface and dust aerosol burden have three dimensions: times, longitudes, and latitudes ($1200 \times 192 \times 288$). Each of these quantitives has a strong impact on the absorption of solar radiation and on cloud formation.
- *Combustion data.* We consider combustion simulation data from [22]. The data consists of three measured quantities — pressure, CO concentration, and temperature — each observed on a $1408 \times 128 \times 128$ spatial grid.
- *Video data.* We use our streaming method to cluster frames of a video, as in

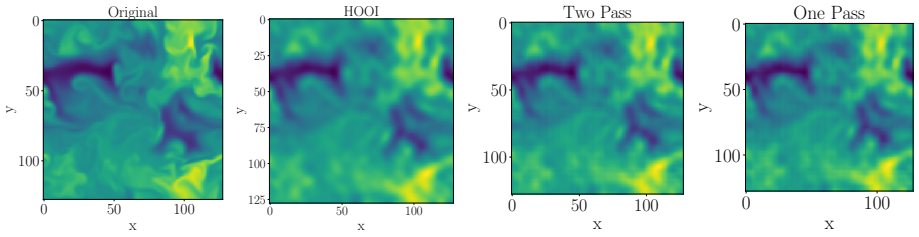


Fig. 8: *Visualizing Combustion Simulation:* All four figures show a slice of the temperature data along the first dimension. The approximation uses $\mathbf{r} = (281, 25, 25)$, $\mathbf{k} = (562, 50, 50)$, $\mathbf{s} = (1125, 101, 101)$, with the Gaussian TRP in the Tucker sketch.

[25]. Here, a low frame rate camera is mounted in a fixed position as people walk by. A 3D tensor is constructed with each video frames as a slice. The video consists of 2493 frames, each of size 1080 by 1980. As a tensor, stored as a `numpy.array`, the video data is 41.4 GB in total.

6.2.1. Data compression. We show that our proposed algorithms are able to successfully compress climate and combustion data even when the full data does not fit in memory. Since the Tucker rank of the original tensor is unknown, we perform experiments for three different target ranks. In this experiment, we hope to understand the effect of different choices of storage budget k to achieve the same compression ratio. We define the compression ratio as the ratio in size between the original input tensor and the output Tucker factors, i.e. $\frac{\prod_{i=1}^{N} I_i}{\sum_{i=1}^{N} r_i I_i + \prod_{i=1}^{N} r_i}$. As in our experiments on simulated data, 5 shows that the two-pass algorithm outperforms the one-pass algorithm as expected. However, as the storage budget k increases, both methods converge to the performance of HOOI. The rate of convergence is faster for smaller target ranks. Performance of our algorithms on the combustion simulation is qualitatively similar, but converges faster to the performance of HOOI. 8 visualizes the recovery of the temperature data in combustion simulation for a slice along the first dimension. We could observe that the recovery for both two-pass and one-pass algorithm approximate the recovery from HOOI. 13 in I shows similar results on another dataset.

6.2.2. Video scene classification. We show how to use our single pass method to classify scenes in the video data described above. The goal is to identify frames in which people appear. We remove the first 100 frames and last 193 frames where the camera setup happened, as in [25]. We stream over the tensor and sketch it using parameters $k = 300, s = 601$. Finally, we compute a fixed-rank approximation with $\mathbf{r} = (10, 10, 10)$ and $(20, 20, 20)$. We apply K-means clustering to the resulting 10 or 20 dimensional vectors corresponding to each of the remaining 2200 frames.

We experimented with clustering vectors found in three ways: from the two-pass or one-pass Tucker approximation, or directly from the factor sketch.

When matching the video frames with the classification result, we can see that the background light is relatively dark at the beginning, thus classified into `Class` 0. After a change in the backgroun light, most other frames of the video are classified into `Class` 1. When a person passes by the camera, the frames are classified into `Class` 2. Right after the person passed by, the frames are classified into `Class` 0, the brighter background scene, due to the light adjustment.

Our classification results (using the linear sketch or approximation) are similar to those in [25] while using only 1/500 as much storage; the one pass approximation requires more storage (but still less than [25]) to achieve similar performance. In particular, using the sketch itself, rather than the Tucker approximation, to summarize the data enables very efficient video scene classification.

On the other hand, to reconstruct the original video frames we require much larger $\mathbf{k}$ and $\mathbf{r}$: the video is not very low rank along the spatial dimensions. 7 shows that even with $\mathbf{s} = 601, 601, 601, \mathbf{k} = (300, 300, 300), \mathbf{r} = (50, 50, 50)$, the recovered frame is very noisy.

Acknowledgments. MU, YS, and YG were supported in part by DARPA Award FA8750-17-2-0101. JAT gratefully acknowledges support from ONR Awards N00014-11-10025, N00014-17-12146, and N00014-18-12363. The authors wish to thank Osman Asif Malik and Stephen Becker for their help in understanding and implementing Tucker TensorSketch, and Tamara Kolda for insightful comments on an early draft.

REFERENCES

[1] D. Achlioptas, *Database-friendly random projections: Johnson-Lindenstrauss with binary coins*, Journal of computer and System Sciences, 66 (2003), pp. 671–687.

[2] N. Ailon and B. Chazelle, *The fast Johnson–Lindenstrauss transform and approximate nearest neighbors*, SIAM Journal on computing, 39 (2009), pp. 302–322.

[3] S. Arora and B. Barak, *Computational complexity: a modern approach*, Cambridge University Press, 2009.

[4] W. Austin, G. Ballard, and T. G. Kolda, *Parallel tensor compression for large-scale scientific data*, in Parallel and Distributed Processing Symposium, 2016 IEEE International, IEEE, 2016, pp. 912–922.

[5] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, *Tthresh: Tensor compression for multidimensional visual data*, IEEE transactions on visualization and computer graphics, (2019).

[6] M. Baskaran, B. Meister, N. Vasilache, and R. Lethin, *Efficient and scalable computations with sparse tensors*, in High Performance Extreme Computing (HPEC), 2012 IEEE Conference on, IEEE, 2012, pp. 1–6.

[7] C. Battaglino, G. Ballard, and T. G. Kolda, *A practical randomized cp tensor decomposition*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 876–901.

[8] C. Battaglino, G. Ballard, and T. G. Kolda, *Faster parallel tucker tensor decomposition using randomization*, (2019).

[9] C. Boutsidis and A. Gittens, *Improved matrix algorithms via the subsampled randomized hadamard transform*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 1301–1340.

[10] A. Cichocki, *Tensor decompositions: a new concept in brain data analysis?*, arXiv preprint arXiv:1305.0395, (2013).

[11] K. L. Clarkson and D. P. Woodruff, *Low-rank approximation and regression in input sparsity time*, Journal of the ACM (JACM), 63 (2017), p. 54.

[12] G. Cormode and M. Hadjieleftheriou, *Finding frequent items in data streams*, Proceedings of the VLDB Endowment, 1 (2008), pp. 1530–1541.

[13] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM journal on Matrix Analysis and Applications, 21 (2000), pp. 1253–1278.

[14] V. De Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 1084–1127.

[15] H. Diao, Z. Song, W. Sun, and D. P. Woodruff, *Sketching for Kronecker Product Regression and P-splines*, arXiv e-prints, (2017), arXiv:1712.09473, p. arXiv:1712.09473, https://arxiv.org/abs/1712.09473.

[16] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM review, 53 (2011), pp. 217–288.

[17] J. W. Hurrell, M. M. Holland, P. R. Gent, S. Ghan, J. E. Kay, P. J. Kushner, J.-F. Lamarque, W. G. Large, D. Lawrence, K. Lindsay, et al., *The community earth system model: a framework for collaborative research*, Bulletin of the American Meteorological Society, 94 (2013), pp. 1339–1360.

[18] J. Kay, C. Deser, A. Phillips, A. Mai, C. Hannay, G. Strand, J. Arblaster, S. 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 (2015), pp. 1333–1349.

[19] O. Kaya and B. Uçar, *High performance parallel algorithms for the tucker decomposition of sparse tensors*, in Parallel Processing (ICPP), 2016 45th International Conference on, IEEE, 2016, pp. 103–112.

[20] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM review, 51 (2009), pp. 455–500.

[21] T. G. Kolda and J. Sun, *Scalable tensor decompositions for multi-aspect data mining*, in 2008 Eighth IEEE International Conference on Data Mining, IEEE, 2008, pp. 363–372.

[22] S. Lapointe, B. Savard, and G. Blanquart, *Differential diffusion effects, distributed burning, and local extinctions in high karlovitz premixed flames*, Combustion and flame, 162 (2015), pp. 3341–3355.

[23] J. Li, C. Battaglino, I. Perros, J. Sun, and R. Vuduc, *An input-adaptive and in-place approach to dense tensor-times-matrix multiply*, in High Performance Computing, Networking, Storage and Analysis, 2015 SC-International Conference for, IEEE, 2015,

pp. 1–12.
[24] P. Li, T. J. Hastie, and K. W. Church, *Very sparse random projections*, in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2006, pp. 287–296.
[25] O. A. Malik and S. Becker, *Low-rank tucker decomposition of large tensors using tensorsketch*, in Advances in Neural Information Processing Systems, 2018, pp. 10116–10126.
[26] S. Muthukrishnan et al., *Data streams: Algorithms and applications*, Foundations and Trends® in Theoretical Computer Science, 1 (2005), pp. 117–236.
[27] S. Oymak and J. A. Tropp, *Universality laws for randomized dimension reduction, with applications*, Information and Inference: A Journal of the IMA, (2015).
[28] M. Rudelson, *Row products of random matrices*, Advances in Mathematics, 231 (2012), pp. 3199–3231.
[29] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, *Incremental tensor analysis: Theory and applications*, ACM Transactions on Knowledge Discovery from Data (TKDD), 2 (2008), p. 11.
[30] Y. Sun, Y. Guo, J. A. Tropp, and M. Udell, *Tensor random projection for low memory dimension reduction*, in NeurIPS Workshop on Relational Representation Learning, 2018, https://r2learning.github.io/assets/papers/CameraReadySubmission%2041.pdf.
[31] J. A. Tropp, *Improved analysis of the subsampled randomized hadamard transform*, Advances in Adaptive Data Analysis, 3 (2011), pp. 115–126.
[32] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Practical sketching algorithms for low-rank matrix approximation*, SIAM Journal on Matrix Analysis and Applications, 38 (2017), pp. 1454–1485.
[33] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *More practical sketching algorithms for low-rank matrix approximation*, Tech. Report 2018-01, California Institute of Technology, Pasadena, California, 2018.
[34] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Streaming low-rank matrix approximation with an application to scientific simulation*, Submitted to SISC, (2019), https://arxiv.org/abs/1902.08651.
[35] C. E. Tsourakakis, *Mach: Fast randomized tensor decompositions*, in Proceedings of the 2010 SIAM International Conference on Data Mining, SIAM, 2010, pp. 689–700.
[36] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.
[37] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, *A new truncation strategy for the higher-order singular value decomposition*, SIAM Journal on Scientific Computing, 34 (2012), pp. A1027–A1052.
[38] M. A. O. Vasilescu and D. Terzopoulos, *Multilinear analysis of image ensembles: Tensorfaces*, in European Conference on Computer Vision, Springer, 2002, pp. 447–460.
[39] Y. Wang, H.-Y. Tung, A. J. Smola, and A. Anandkumar, *Fast and guaranteed tensor decomposition via sketching*, in Advances in Neural Information Processing Systems, 2015, pp. 991–999.
[40] D. P. Woodruff et al., *Sketching as a tool for numerical linear algebra*, Foundations and Trends® in Theoretical Computer Science, 10 (2014), pp. 1–157.
[41] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for the approximation of matrices*, Applied and Computational Harmonic Analysis, 25 (2008), pp. 335–366.
[42] G. Zhou, A. Cichocki, and S. Xie, *Decomposition of big tensors with low multilinear rank*, arXiv preprint arXiv:1412.1885, (2014).

Appendix A. Proof of Main Results.

A.1. Error bound for the two pass approximation Algorithm 4.2.

Proof of Theorem 5.1. Suppose $\hat{\mathcal{X}}_2$ is the low-rank approximation from 4.2. Use the definition of the mode-n product to see

$$\begin{aligned}\hat{\mathcal{X}}_2 &= \left[\mathcal{X} \times_1 \mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_N^\top\right] \times_1 \mathbf{Q}_1 \times_1 \cdots \times_N \mathbf{Q}_N \\ &= \mathcal{X} \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_N\mathbf{Q}_N^\top.\end{aligned}$$

Although it seems that we sequentially project tensor $\mathcal{X}$ to column space spanned with $\mathbf{Q}_n$, but since mode product is exchangeable, in fact $\hat{\mathcal{X}}_2$ is the projection to space $\{\mathcal{X} : \mathcal{X}^{(n)} \in \mathbf{col}(\mathbf{Q}_n)\}$. This is a generalization of projection matrix where [14] has a very detailed explanation and it is referred as multi-linear orthogonal projection. Following exact techniques in Theorem 5.1 in [37] by sequentially applying Pythagorean theory sequentially we can show that

$$\|\hat{\mathcal{X}}_2 - \mathcal{X}\|_F^2 \leqslant \sum_{n=1}^{N} \left\|(\mathbf{I} - \mathbf{Q}_n\mathbf{Q}_n^\top)\mathbf{X}^{(n)}\right\|_F^2. \tag{A.1}$$

∎

Then taking expectation on $\mathbf{Q}_n$, and applying Lemma D.2 we complete the proof.

A.2. Error bound for the one pass approximation Algorithm 4.3.

Proof of Theorem 5.2. We show the approximation error can be decomposed as the error due to the factor matrix approximations and the error due to the core approximation. Let $\hat{\mathcal{X}}_1$ be the one pass approximation from 4.3, and let

$$\hat{\mathcal{X}}_2 = \mathcal{X} \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_N\mathbf{Q}_N^\top, \tag{A.2}$$

be the two pass approximation from 4.2. The difference in one-pass and two-pass approximation is in the core:

$$\hat{\mathcal{X}}_1 - \hat{\mathcal{X}}_2 = (\mathcal{W} - \mathcal{X} \times_1 \mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_n^\top) \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N.$$

Thus $\hat{\mathcal{X}}_1 - \hat{\mathcal{X}}_2$ is in the space defined above: $\{\mathcal{X} : \mathcal{X}^{(n)} \in \mathbf{col}(\mathbf{Q}_n)\}$ while as pointed before $\hat{\mathcal{X}}_2 - \mathcal{X}$ is perpendicular to that space. Therefore,

$$\langle \hat{\mathcal{X}}_1 - \hat{\mathcal{X}}_2, \hat{\mathcal{X}}_2 - \mathcal{X} \rangle = 0. \tag{A.3}$$

Now we use the (expectation of) the Pythagorean theorem to bound the expected error of the one pass approximation:

$$\mathbb{E}\|\hat{\mathcal{X}}_1 - \mathcal{X}\|_F^2 = \mathbb{E}\|\hat{\mathcal{X}}_1 - \hat{\mathcal{X}}_2\|_F^2 + \mathbb{E}\|\hat{\mathcal{X}}_2 - \mathcal{X}\|_F^2. \tag{A.4}$$

Consider the first term which is due to core approximation. Based in the definition of $\hat{\mathcal{X}}_1$ and $\tilde{\mathcal{X}}_2$ we can see that

$$\begin{aligned}\|\hat{\mathcal{X}}_1 - \hat{\mathcal{X}}_2\|_F^2 &= \|(\mathcal{W}_1 - \mathcal{X} \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top) \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N\|_F^2 \\ &= \|(\mathcal{W}_1 - \mathcal{X} \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top)\|_F^2,\end{aligned}$$

where we use the invariance of the Frobenius norm under orthonormal transformations to get the second line. Now using B.2 to bound for the error due to the core approximation as

$$\mathbb{E}\|\hat{\mathcal{X}}_1 - \hat{\mathcal{X}}_2\|_F^2 \leqslant \Delta \left[\sum_{n=1}^{N} \left(1 + \frac{\rho_n}{k_n - \rho_n - 1}\right) (\tau_{\rho_n}^{(n)})^2\right].$$

Finally, as shown in proof for 5.1 to bound the error due to the factor matrix approximations (the second term in (A.4)) as

$$\mathbb{E}\|\hat{\mathcal{X}}_2 - \mathcal{X}\|_F^2 \leqslant \left[\sum_{n=1}^{N} \left(1 + \frac{\rho_n}{k_n - \rho_n - 1}\right) (\tau_{\rho_n}^{(n)})^2\right].$$

Summing these two bounds finishes the proof. ∎

A.3. Error bound for the fixed rank approximation Algorithm 4.4.

Proof of Theorem 5.3. Our argument follows the proof of [32, Proposition 6.1]:

$$\begin{aligned}
\|\mathcal{X} - [\![\hat{\mathcal{X}}]\!]_{\mathbf{r}}\|_F &\leqslant \|\mathcal{X} - \hat{\mathcal{X}}\|_F + \|\hat{\mathcal{X}} - [\![\hat{\mathcal{X}}]\!]_{\mathbf{r}}\|_F \\
&\leqslant \|\mathcal{X} - \hat{\mathcal{X}}\|_F + \|\hat{\mathcal{X}} - [\![\mathcal{X}]\!]_{\mathbf{r}}\|_F \\
&\leqslant \|\mathcal{X} - \hat{\mathcal{X}}\|_F + \|\hat{\mathcal{X}} - \mathcal{X} + \mathcal{X} - [\![\mathcal{X}]\!]_{\mathbf{r}}\|_F \\
&\leqslant 2\|\mathcal{X} - \hat{\mathcal{X}}\|_F + \|\mathcal{X} - [\![\mathcal{X}]\!]_{\mathbf{r}}\|_F.
\end{aligned}$$

The first and the third line are the triangle inequality, and the second line follows from the definition of the best rank-r approximation. Take the expectation of $\|\mathcal{X} - \hat{\mathcal{X}}\|_F$ and use Jensen's inequality $\mathbb{E}\|\mathcal{X} - \hat{\mathcal{X}}\|_F \leqslant \sqrt{\mathbb{E}\|\mathcal{X} - \hat{\mathcal{X}}\|_F^2}$ to finish the proof. ∎

Appendix B. Probabilistic Analysis of Core Sketch Error. This section contains the most technical part of our proof. We provide a probabilistic error bound for the difference between the two pass core approximation $\mathcal{W}_2$ from 4.2 and the one pass core approximation $\mathcal{W}_1$ from 4.3.

Introduce for each $n \in [N]$ the orthonormal matrix $\mathbf{Q}_n^{\perp}$ that forms a basis for the subspace orthogonal to $\mathbf{Q}_n$, so that $\mathbf{Q}_n^{\perp}(\mathbf{Q}_n^{\perp})^{\top} = \mathbf{I} - \mathbf{Q}_n\mathbf{Q}_n^{\top}$. Next, define

$$\mathbf{\Phi}_n^{Q} = \mathbf{\Phi}_n^{\top}\mathbf{Q}_n, \qquad \mathbf{\Phi}_n^{Q^{\perp}} = \mathbf{\Phi}_n^{\top}\mathbf{Q}_n^{\perp}. \tag{B.1}$$

Recall that the DRMs $\mathbf{\Phi}_n$ are i.i.d. Gaussian. Hence conditional on $\mathbf{Q}_n$, $\mathbf{\Phi}_n^{Q}$ and $\mathbf{\Phi}_n^{Q^{\perp}}$ are independent.

B.1. Decomposition of Core Approximation Error. In this section, we characterize the difference between the one and two pass core approximations $\mathcal{W}_1 - \mathcal{W}_2 = \mathcal{W}_1 - \mathcal{X} \times_1 \mathbf{Q}_1^{\top} \cdots \times_N \mathbf{Q}_N^{\top}$.

LEMMA B.1. *Suppose that $\mathbf{\Phi}_n$ has full column rank for each $n \in [N]$. Then*

$$\mathcal{W}_1 - \mathcal{W}_2 = \mathcal{W}_1 - \mathcal{X} \times_1 \mathbf{Q}_1^{\top} \cdots \times_N \mathbf{Q}_N^{\top} = \sum_{(i_1,\ldots,i_N)\in\{0,1\}^N, \sum_{j=1}^N i_j \geqslant 1} \mathcal{Y}_{i_1\ldots i_N},$$

where

$$\begin{aligned}
\mathcal{Y}_{i_1\ldots i_N} = \mathcal{X} &\times_1 \left(\mathbb{1}_{i_1=0}\mathbf{Q}_1^{\top} + \mathbb{1}_{i_1=1}(\mathbf{\Phi}_1^{Q_1})^{\dagger}\mathbf{\Phi}_1^{Q_1^{\perp}}(\mathbf{Q}_1^{\perp})^{\top}\right) \\
&\times_2 \cdots \times_N \left(\mathbb{1}_{i_N=0}\mathbf{Q}_N^{\top} + \mathbb{1}_{i_1=1}(\mathbf{\Phi}_N^{Q_N})^{\dagger}\mathbf{\Phi}_N^{Q_N^{\perp}}(\mathbf{Q}_N^{\perp})^{\top}\right).
\end{aligned} \tag{B.2}$$

Proof. Let $\mathcal{H}$ be the core sketch from 4.1. Write $\mathcal{W}_1$ as

$$\begin{aligned}\mathcal{W}_1 &= \mathcal{H} \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \times_2 \cdots \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger \\ &= (\mathcal{X} - \hat{\mathcal{X}}_2) \times_1 \mathbf{\Phi}_1^\top \times_2 \cdots \times_N \mathbf{\Phi}_N^\top \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \times_2 \cdots \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger \\ &\quad + \hat{\mathcal{X}}_2 \times_1 \mathbf{\Phi}_1^\top \times_2 \cdots \times_N \mathbf{\Phi}_N^\top \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \times_2 \cdots \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger.\end{aligned}$$

Using the fact that $(\mathbf{\Phi}_n^\top \mathbf{Q}_n)^\dagger (\mathbf{\Phi}_n^\top \mathbf{Q}_n) = \mathbf{I}$, we can simplify the second term as

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

which is exactly the two pass core approximation $\mathcal{W}_2$. Therefore

$$\mathcal{W}_1 - \mathcal{W}_2 = (\mathcal{X} - \tilde{\mathcal{X}}) \times_1 \mathbf{\Phi}_1^\top \times_2 \cdots \times_N \mathbf{\Phi}_N^\top \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \times_2 \cdots \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger.$$

We continue to simplify this difference:

$$\begin{aligned}&(\mathcal{X} - \tilde{\mathcal{X}}) \times_1 \mathbf{\Phi}_1^\top \times_2 \cdots \times_N \mathbf{\Phi}_N^\top \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \times_2 \cdots \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger \\ &\quad = (\mathcal{X} - \tilde{\mathcal{X}}) \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \mathbf{\Phi}_1^\top \times_2 \cdots \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger \mathbf{\Phi}_N^\top \\ &\quad = (\mathcal{X} - \tilde{\mathcal{X}}) \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \mathbf{\Phi}_1^\top (\mathbf{Q}_1 \mathbf{Q}_1^\top + \mathbf{Q}_1^\perp (\mathbf{Q}_1^\perp)^\top) \ldots \\ &\quad \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger \mathbf{\Phi}_N^\top (\mathbf{Q}_N \mathbf{Q}_N^\top + \mathbf{Q}_N^\perp (\mathbf{Q}_N^\perp)^\top) \\ &\quad = (\mathcal{X} - \tilde{\mathcal{X}}) \times_1 (\mathbf{Q}_1^\top + (\mathbf{\Phi}_1^Q)^\dagger \mathbf{\Phi}_1^{Q^\perp} (\mathbf{Q}_1^\perp)^\top) \times_2 \ldots \\ &\quad \times_N (\mathbf{Q}_N^\top + (\mathbf{\Phi}_N^{Q_N})^\dagger \mathbf{\Phi}_N^{Q_N^\perp} (\mathbf{Q}_N^\perp)^\top).\end{aligned} \tag{B.3}$$

Many terms in this sum are zero. We use the following two facts:

1. $(\mathcal{X} - \tilde{\mathcal{X}}) \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top = 0$.
2. For each $n \in [N]$, $\tilde{\mathcal{X}} \times_n (\mathbf{\Phi}_n^{Q_n})^\dagger \mathbf{\Phi}_n^{Q_n^\perp} (\mathbf{Q}_n^\perp)^\top = 0$.

Here 0 means a tensor with all zero elements. These facts can be obtained from the exchange rule of the mode product and the orthogonality between $\mathbf{Q}_n^\perp$ and $\mathbf{Q}_n$. Using these two facts, we find that only the terms $\mathcal{Y}_{i_1 \ldots i_N}$ (defined in (B.2)) remain in the expression. Therefore, to complete the proof, we write (B.3) as ∎

$$\sum_{(i_1,\ldots,i_N) \in \{0,1\}^N, \sum n=1^N i_n \neq 0} \mathcal{Y}_{i_1 \ldots i_N}.$$

B.2. Probabilistic Core Error Bound. In this section, we derive a probabilistic error bound based on the core error decomposition from Lemma B.1.

LEMMA B.2. *Sketch the tensor $\mathcal{X}$ using a Tucker sketch with parameters $\mathbf{k}$ and $\mathbf{s} > 2\mathbf{k}$ with i.i.d. Gaussian $\mathcal{N}(0,1)$ DRMs. Define $\Delta = \max_{n=1}^N \frac{k_n}{s_n - k_n - 1}$. Then for any natural numbers $1 \leqslant \rho < \mathbf{k} - 1$,*

$$\mathbb{E}\|\mathcal{W}_1 - \mathcal{X} \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top\|_F^2 \leqslant \Delta \left[\sum_{n=1}^N \left(1 + \frac{\rho_n}{k_n - \rho_n - 1} \right) (\tau_{\rho_n}^{(n)})^2 \right].$$

Proof. It suffices to show

$$\mathbb{E}\left[\|\mathcal{W}_1 - \mathcal{X} \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top\|_F^2 \mid \mathbf{\Omega}_1, \cdots, \mathbf{\Omega}_N \right] \leqslant \Delta \|\mathcal{X} - \hat{\mathcal{X}}_2\|_F^2. \tag{B.4}$$

Then take the expectation with respect to $\mathbf{\Omega}_1, \cdots, \mathbf{\Omega}_N$ and apply results in 5.1 to bound $\|\mathcal{X} - \hat{\mathcal{X}}_2\|_F^2$ to finish the proof. To show B.4, we will use the fact that the core DRMs $\{\mathbf{\Omega_n}\}_{n\in[N]}$ are independent of the factor matrix DRMs $\{\mathbf{\Phi_n}\}_{n\in[N]}$, and that the randomness in each factor matrix approximation $\mathbf{Q}_n$ comes solely from $\mathbf{\Omega}_n$.

For $i \in \{0,1\}^N$, define $\mathcal{B}_{i_1\dots i_N} =$

$$\mathcal{X} \times_1 (\mathbb{1}_{i_1=0}\mathbf{Q}_1\mathbf{Q}_1^\top + \mathbb{1}_{i_1=1}\mathbf{Q}_1^\perp(\mathbf{Q}_1^\perp)^\top) \cdots \times_N (\mathbb{1}_{i_N=0}\mathbf{Q}_N\mathbf{Q}_N^\top + \mathbb{1}_{i_N=1}\mathbf{Q}_N^\perp(\mathbf{Q}_N^\perp)^\top).$$

B.1 decomposes the core error as the sum of $\mathcal{Y}_{i_1\cdots i_n}$ where $\sum_{n=1}^N i_n \geqslant 1$. Applying D.1 and using the orthogonal invariance of the Frobenius norm, we observe

$$\mathbb{E}\left[\|\mathcal{Y}_{i_1\dots i_N}\|_F^2 \mid \mathbf{\Omega}_1 \cdots \mathbf{\Omega}_N\right] = \left(\prod_{n=1}^N \Delta_n^{i_n}\right) \|\mathcal{B}_{i_1\dots i_N}\|_F^2 \leqslant \Delta \|\mathcal{B}_{i_1\dots i_N}\|_F^2$$

when $\sum_{n=1}^N i_n \geqslant 1$, where $\Delta_n = \frac{k_n}{s_n - k_n - 1} < 1$ and $\Delta = \max_{n=1}^N \Delta_n$.

Suppose $\mathbf{q}_1, \mathbf{q}_2 \in \{0,1\}^N$ are index (binary) vectors of length N. For different indices $\mathbf{q}_1$ and $\mathbf{q}_2$, there exists some $1 \leqslant r \leqslant N$ such that their r-th element is different. Without loss of generality, assume $\mathbf{q}_1(r) = 0$ and $\mathbf{q}_2(r) = 1$ to see

$$\langle \mathcal{B}_{q_1}, \mathcal{B}_{q_2} \rangle = \langle \dots \mathbf{Q}_r^\top \mathbf{Q}_r^\perp \dots \rangle = 0. \tag{B.5}$$

Similarly we can show that the inner product between $\mathcal{Y}_{q_1}$ and $\mathcal{Y}_{q_2}$ is zero with different $\mathbf{q}_1, \mathbf{q}_2$. Noticing that $\mathcal{B}_{0,\dots,0} = \hat{\mathcal{X}}_2$, we have

$$\|\mathcal{X} - \hat{\mathcal{X}}_2\|_F^2 = \left\| \sum_{(i_1,\dots,i_N)\in\{0,1\}^N, \sum_{n=1}^N i_n \geqslant 1} \mathcal{B}_{i_1\dots i_N} \right\|_F^2 = \sum_{\substack{(i_1,\dots,i_N)\in\{0,1\}^N, \\ \sum_{n=1}^N i_n \geqslant 1}} \|\mathcal{B}_{i_1\dots i_N}\|_F^2.$$

Putting all these together and using the Pythagorean theorem, to show B.4:

$$\begin{aligned}
&\mathbb{E}\left[\|\mathcal{W} - \mathcal{X} \times_1 \mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N^\top\|_F^2 \mid \mathbf{\Omega}_1, \cdots, \mathbf{\Omega}_N\right] \\
&= \sum_{(i_1,\dots,i_N)\in\{0,1\}^N, \sum_{n=1}^N i_n \geqslant 1} \mathbb{E}\left[\|\mathcal{Y}_{i_1\dots i_N}\|_F^2 \mid \mathbf{\Omega_1}, \dots, \mathbf{\Omega}_N\right] \\
&\leqslant \Delta \left(\sum_{(i_1,\dots,i_N)\in\{0,1\}^N, \sum_{n=1}^N i_n \geqslant 1} \|\mathcal{B}_{i_1\dots i_N}\|_F^2 \right) = \Delta \|\mathcal{X} - \hat{\mathcal{X}}_2\|_F^2.
\end{aligned}$$

∎

Appendix C. Proof of fixed rank approximation lemma.

Proof of 4.1. The target tensor to be approximated is $\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N$ is apparently in the space $\{\mathcal{X} : \mathcal{X}^{(n)} \in \mathbf{col}(\mathbf{Q}_n)\}$. For any approximation $\hat{\mathcal{X}}$, we can project it into this space as

$$\hat{\mathcal{X}} \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \times_2 \cdots \times_N \mathbf{Q}_N\mathbf{Q}_N^\top$$

and by Pythagorean theory,

$$\begin{aligned}
\|\hat{\mathcal{X}} - \mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N\|_F &\leqslant \|\hat{\mathcal{X}} - \hat{\mathcal{X}} \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N\mathbf{Q}_N^\top\|_F^2 \\
&+ \|\hat{\mathcal{X}} \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \cdots \times_N \mathbf{Q}_N\mathbf{Q}_N^\top - \mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N\|_F^2,
\end{aligned} \tag{C.1}$$

which indicates that the optimal Tucker decomposition resides in the space $\{\mathcal{X} : \mathcal{X}^{(n)} \in \mathbf{col}(\mathbf{Q}_n)\}$. Suppose $[\![\mathcal{W}; \mathbf{V}_1, \cdots, \mathbf{V}_N]\!]$ is the optimal solution to the problem, since its unfolding is in the space spanned by $\mathbf{Q}_n$, each $\mathbf{V}_n$ can be written as $\mathbf{Q}_n\mathbf{U}_n$ for some orthogonal matrix $\mathbf{U}_n \in \mathbb{R}^{k_n \times r_n}$. Then, noticing orthogonal transformation does not change Frobenius norm,

$$\begin{aligned} &\|\mathcal{W} \times_1 \mathbf{Q}_1 \times \cdots \times_N \mathbf{Q}_N - \mathcal{G} \times_1 \mathbf{Q}_1\mathbf{U}_1 \times \cdots \times_N \mathbf{Q}_N\mathbf{U}_N\|_F \\ &= \|\mathcal{W} - \mathcal{G} \times_1 \mathbf{U}_1 \times \cdots \times_N \mathbf{U}_N\|_F \geqslant \|\mathcal{W} - [\![\mathcal{W}]\!]_{\mathbf{r}}\|_F \\ &= \|\mathcal{W} \times_1 \mathbf{Q}_1 \times \cdots \times_N \mathbf{Q}_N - [\![\mathcal{W}]\!]_{\mathbf{r}} \times_1 \mathbf{Q}_1 \times \cdots \times_N \mathbf{Q}_N\|_F. \end{aligned} \tag{C.2}$$

This finishes the proof. ∎

Appendix D. Technical Lemmas.

D.1. Random projections of matrices. Proofs for lemmas in this section appear in [16, chapters 9 and 10].

LEMMA D.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,*

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

LEMMA D.2. *Suppose that $\mathbf{A}$ is a real $m \times n$ matrtix with singular value $\sigma_1 \geqslant \sigma_2 \geqslant \cdots$, choose a target rank $k \geqslant 2$ and an oversampling parameter $p \geqslant 2$, where $k + p \leqslant \min\{m, n\}$. 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 is*

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

Appendix E. More Algorithms. This section provides detailed implementations for a linear sketch appropriate to a streaming setting (Algorithm E.1) or a distributed setting (Algorithm E.2).

Algorithm E.1 Linear Update to Sketches

```
function SketchLinearUpdate(F, V_1, ..., V_N, H; θ_1, θ_2)
    for n = 1, ..., N do
        V_n ← θ_1 V_n + θ_2 F^(n) Ω_n
    end for
    H ← θ_1 H + θ_2 F ×_1 Φ_1 × ··· ×_N Φ_N
    return (V_1, ..., V_N, H)
end function
```

Appendix F. Scrambled Subsampled Randomized Fourier Transform.

In order to reduce the cost of storing the test matrices, in particular, $\mathbf{\Omega}_1, \ldots, \mathbf{\Omega}_N$, we can use the Scrambled Subsampled Randomized Fourier Transform (SSRFT). To reduce the dimension of a matrix, $\mathbf{X} \in \mathbb{R}^{m\times n}$, along either the row or the column to size k, we define the SSRFT map $\mathbf{\Xi}$ as:

$$\mathbf{\Xi} = \begin{cases} \mathbf{R}\mathbf{F}^\top\mathbf{\Pi}\mathbf{F}\mathbf{\Pi}^\top \in \mathbb{F}^{k\times m} & \text{(Row linear transform)} \\ (\bar{\mathbf{R}}\bar{\mathbf{F}}^\top\bar{\mathbf{\Pi}}\bar{\mathbf{F}}\bar{\mathbf{\Pi}}^\top)^\top \in \mathbb{F}^{n\times k} & \text{(Column linear transform),} \end{cases}$$

Algorithm E.2 Sketching in Distributed Setting

Require: $\mathcal{X}_i$ is the part of the tensor $\mathcal{X}$ at local machine i and $\mathcal{X} = \sum_{i=1}^{m} \mathcal{X}_i$.

function ComputeSketchDistributed($\mathcal{X}_1, \ldots, \mathcal{X}_m$)
  Send the same random generating environment to every local machine.
  Generate the same DRM at each local machine.
  for $i = 1 \ldots m$ **do**
    $(\mathbf{V}_1^{(i)}, \cdots, \mathbf{V}_n^{(i)}, \mathcal{H}^{(i)}) \leftarrow$ ComputeSketch($\mathcal{X}_i$)
  end for
  for $j = 1 \ldots n$ **do**
    $\mathbf{V}_j \leftarrow \sum_{i=1}^{m} \mathbf{V}_j^{(i)}$
  end for
  $\mathcal{H} \leftarrow \sum_{i=1}^{m} \mathcal{H}^{(i)}$
  return $(\mathbf{V}_1, \ldots, \mathbf{V}_n, \mathcal{H})$
end function

where $\mathbf{\Pi}, \mathbf{\Pi}' \in \mathbb{R}^{m\times m}, \bar{\mathbf{\Pi}}, \bar{\mathbf{\Pi}}' \in \mathbb{R}^{n\times n}$ are signed permutation matrices. That is, the matrix has exactly one non-zero entry, 1 or -1 with equal probability, in each row and column. $\mathbf{F} \in \mathbb{F}^{m\times m}, \mathbf{F} \in \mathbb{F}^{n\times n}$ denote the discrete cosine transform ($\mathbb{F} = \mathbb{R}$) or the discrete fourier transform ($\mathbb{F} = \mathbb{C}$). The matrix $\mathbf{R}, \bar{\mathbf{R}}$ is the restriction to k coordinates chosen uniformly at random.

In practice, we implement the SSRFT as in Algorithm F.1. It takes only $\mathcal{O}(m)$ or $\mathcal{O}(n)$ bits to store $\mathbf{\Xi}$, compared to $\mathcal{O}(km)$ or $\mathcal{O}(kn)$ for Gaussian or uniform random map. The cost of applying $\mathbf{\Xi}$ to a vector is $\mathcal{O}(n \log n)$ or $\mathcal{O}(m \log m)$ arithmetic operations for fast Fourier transform and $\mathcal{O}(n \log k)$ or $\mathcal{O}(m \log k)$ for fast cosine transform. Though in practice, SSRFT behaves similarly to the Gaussian random map, its analysis is less comprehensive [9, 31, 2] than the Gaussian case.

Algorithm F.1 Scrambled Subsampled Randomized Fourier Transform (Row Linear Transform)

Require: $\mathbf{X} \in \mathbb{R}^{m\times n}, \mathcal{F} = \mathbb{R}$, **randperm** creates a random permutaion vector, and **randsign** creates a random sign vector. **dct** denotes the discrete cosine transform.

function SSRFT($\mathbf{X}$)
  coords $\leftarrow$ **randperm**(m,k)
  $\mathbf{perm}_j \leftarrow$ **randperm**(m) for $j = 1, 2$
  $\mathbf{sgn}_j \leftarrow$ **randsign**(m) for $j = 1, 2$
  $\mathbf{X} \leftarrow$ **dct**$(\mathbf{sgn}_1 \cdot \mathbf{X}[\mathbf{perm}_1, :])$ ▷ elementwise product
  $\mathbf{X} \leftarrow$ **dct**$(\mathbf{sgn}_2 \cdot \mathbf{X}[\mathbf{perm}_2, :])$
  return $\mathbf{X}$[**coords**, :]
end function

Appendix G. TensorSketch. Many authors have developed methods to perform dimension reduction efficiently. In particular [15] proposed a method called tensor sketching aiming to solve least square problem with design matrix has kroneck product structure. [25] applied this technique to their one pass Tucker decomposition. Here we review the definition of tensor sketch and how it be applied in [25].

CountSketch. [12] proposed the CountSketch method. A comprehensive theoretical analysis in the context of low-rank approximation problems appears in [11]. To

compute the sketch $\mathbf{X}\boldsymbol{\Omega} \in \mathbb{R}^{d\times k}$ for $\mathbf{X} \in \mathbb{R}^{m\times d}$, CountSketch defines $\boldsymbol{\Omega} = \mathbf{D}\boldsymbol{\Phi}$, where

1. $\mathbf{D} \in \mathbb{R}^{d\times d}$ is a diagonal matrix with each diagonal entry equal to $(-1, 1)$ with probability $(1/2, 1/2)$.
2. $\boldsymbol{\Phi} \in \mathbb{R}^{d\times k}$ is the matrix form of a Hashing function.

In total, these two matrices have $2d$ non-zero entries in total, thus requiring much less storage than the standard kd entries. Furthermore, these two matrices can act as an operator on each column of $\mathbf{X}$ and require only $\mathcal{O}(kd)$ operations.

TensorSketch. [25] proposes to use the countsketch inside the HOOI method for Tucker decomposition. They apply sketching method solve least square problem appearing in (2.4) and (2.5) in Algorithm 2.2. They use J_1, J_2 to denote the reduced dimension. Using a standard random map, it will need J_1-by-$I_{(-n)}$ random matrix for (2.4) and a J_2-by-$\prod_{n=1}^{N} I_n$ random matrix to compute (2.5).

But as shown in [25], these two stages can be expressed as

$$\text{(G.1)}\quad \text{For } n = 1, \dots, N, \text{update } \mathbf{U}^{(n)} = \underset{\mathbf{U}\in\mathbb{R}^{I_n\times R_n}}{\arg\min} \left\| \left(\bigotimes_{\substack{i=N \\ i\neq n}}^{1} \mathbf{U}^{(i)} \right) \mathbf{G}_{(n)}^{\top} \mathbf{U}^{\top} - \mathbf{Y}_{(n)}^{\top} \right\|_F^2 .$$

$$\text{(G.2)}\qquad \text{Update } \mathcal{G} = \underset{\mathcal{Z}\in\mathbb{R}^{R_1\times\cdots\times R_N}}{\arg\min} \left\| \left(\bigotimes_{i=N}^{1} \mathbf{U}^{(i)} \right) \mathbf{vec}\, \mathcal{Z} - \mathbf{vec}\, \mathcal{Y} \right\|_2^2 ,$$

where $\mathcal{Y}$ is the original data. $\forall i \in [n], \mathbf{U}_i$ is the factor matrix, and $\mathcal{G}$ is the core tensor. $R_1, \dots, R_N$ denote the rank of the data.

As what shown in [12], [25] proposes to apply tensorSketch to the Kronecker product structure of the input matrix in the sketch construction, i.e. $\otimes_{\substack{i=1\\ i\neq n}}^{N} \mathbf{U}_i$ in (G.1) and $\otimes_{i=1}^{N} \mathbf{U}_i$ in (G.2). TensorSketch method combines the CountSketch of each factor matrix via the Khatri-Rao product and Fast Fourier Transform. Consider sketching $\otimes_{i=1}^{N} \mathbf{U}_i$ in (G.2). TensorSketch is defined as

$$\text{(G.3)}\qquad \boldsymbol{\Omega}\mathbf{X} = \mathrm{FFT}^{-1}\left(\odot_{n=1}^{N} \left(\mathrm{FFT}\big(\text{CountSketch}^{(n)}(\mathbf{U}^{(n)})\big)^{\top} \right)^{\top} \right)$$

By only storing $\text{CountSketch}^{(1)}, \dots, \text{CountSketch}^{(N)}$, TensorSketch only requires $2\sum_{i=1}^{N} I_n$ storage. Therefore, the storage cost of the sketch is dominated by the sketch size, $NR^{n-1}J_1 + J_2R^n \approx NKR^{2n-2} + KR^{2n}$, when $J_1 = KR^{n-1}, J_2 = KR^n$.

Appendix H. Time and Storage Complexity.

H.1. Comparison Between Algorithm 4.4 and T.-TS [25]. Here we compare the time and storage complexity of the two extant methods for streaming Tucker approximation: our one-pass method, and T.-TS [25].

To compare the storage and time costs of both T.-TS and the one-pass algorithm, we separate the cost into two parts: one for forming the sketch, the other for each iteration of ALS. Assume the tensor to approximate has equal side lengths $I_1 = \cdots = I_N = I$ and that the target rank for each mode is R.

The suggested default parameters for the sketch in [25] are $J_1 = 10R^{N-1}$ and $J_2 = 10R^N$. Our suggested default parameters are $k = 2r, s = 2k + 1$. Under the choice of the default parameter, we compare the the cost of storage and time in Table 3 and Table 4. In most problems with data not perfectly low Tucker rank, i.e. $R > 4$, the

suggested default setting of T.-TS typically leads to a higher storage cost. Moreover, our algorithm uses less storage and is faster to compute, particularly for tensors with many modes N.

However, the evaluation of the two algorithms should not be solely based on their default setups. If the memory constraint is set to be the same, our one-pass algorithm performs much better in the low-memory case, but slightly worse in the case with very high-memory as in Figure 4. The memory of our suggested setting typically implies a much smaller memory usage than their suggested setting.

H.2. Computational Complexity of Algorithm 4.4. Here, we will derive a fine-grained computation complexity for our one pass fixed-rank approximation algorithm.

In the sketching stage of the streaming algorithm, we need to first compute the factor sketches, $\mathbf{G}_n = \mathbf{X}\mathbf{\Omega}_n, n \in [N]$ with $kN\hat{I}$ flops in total. Then, we need to 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 $\frac{s(1-\delta_1^N)}{1-\delta_1}\bar{I}$. Then, in the approximation 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 $\mathbf{Q}_1, \ldots, \mathbf{Q}_N$. 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. Overall, the sketch computation dominates the total time complexity.

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 less than the streaming algorithm with a constant factor. Note: we can use the randomized SVD in the first step to improve the computational cost to $\bar{I}N \log k + \sum_{n=1}^N (I_n + I_{(-n)})k^2$ [16].

Algorithm		Storage Cost ($I = o(r^{2N})$)
T.-TS	Sketching	$\mathcal{O}(r^{2N})$
	Recovery	$\mathcal{O}(r^{2N})$
Algorithm 4.3 (One Pass)	Sketching	$\mathcal{O}(4^N r^N)$
	Recovery	$\mathcal{O}(4^N r^N)$

Table 3: Storage complexity of Algorithm 4.3 and T.-TS on tensor $\mathcal{X} \in \mathbb{R}^{I \times \cdots \times I}$. Algorithm 4.3 uses parameters $(k, s) = (2r, 4r + 1)$ and uses a TRP composed of Gaussian DRMs inside the Tucker sketch. T.-TS uses default values for hyper-parameters: $J_1 = 10r^{N-1}, J_2 = 10r^N$.

Appendix I. More Numerics.

This section provides more numerical results on simulated datasets in Figure 9, Figure 10, Figure 11, and Figure 12.

We also provide more numerical results on real datasets in Figure 13.

Algorithm		Time Cost ($I = o(r^{2N})$)
T.-TS	Sketching	$\mathcal{O}(N\text{nnz}(\mathcal{X}))$
	Recovery	$\mathcal{O}(NIr^N + Nr^{2N-1} + r^{2N})$
Algorithm 4.3 (One Pass)	Sketching	$\mathcal{O}(Nr\ \text{nnz}(\mathcal{X})))$
	Recovery	$\mathcal{O}(Nr^{N+1})$

Table 4: Time complexity of Algorithm 4.3 and T.-TS on tensor $\mathcal{X} \in \mathbb{R}^{I\times\cdots\times I}$. Algorithm 4.3 uses parameters $(k, s) = (2r, 4r + 1)$ and uses a TRP composed of Gaussian DRMs inside the Tucker sketch. T.-TS uses default values for hyper-parameters: $J_1 = 10r^{N-1}, J_2 = 10r^N$.

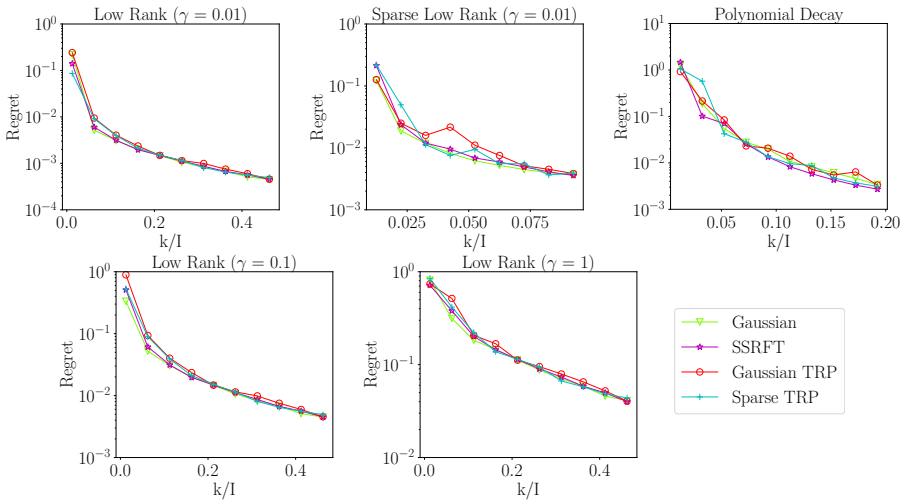


Fig. 9: We approximate 3D synthetic tensors (see subsection 6.1) with $I = 400$, using our one-pass algorithm with $r = 5$ and varying k ($s = 2k+1$), using a variety of DRMs in the Tucker sketch: Gaussian, SSRFT, Gaussian TRP, or Sparse TRP.

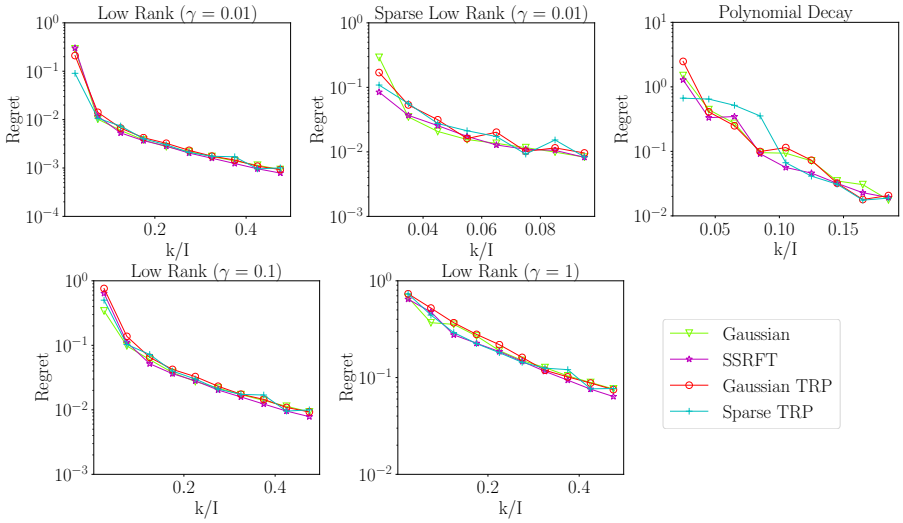


Fig. 10: We approximate 3D synthetic tensors (see subsection 6.1) with $I = 200$, using our one-pass algorithm with $r = 5$ and varying k ($s = 2k+1$), using a variety of DRMs in the Tucker sketch: Gaussian, SSRFT, Gaussian TRP, or Sparse TRP.

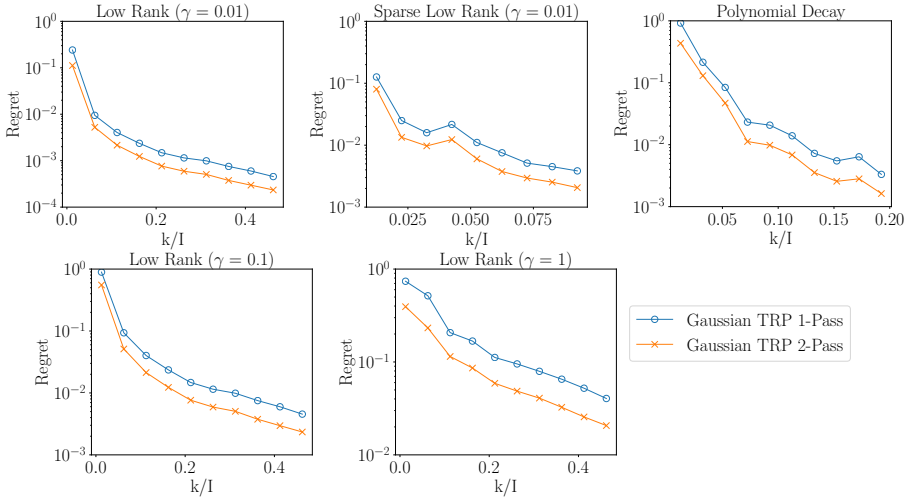


Fig. 11: We approximate 3D synthetic tensors (see subsection 6.1) with $I = 400$, using our one-pass and two-pass algorithms with $r = 5$ and varying k ($s = 2k + 1$), using the Gaussian TRP in the Tucker sketch.

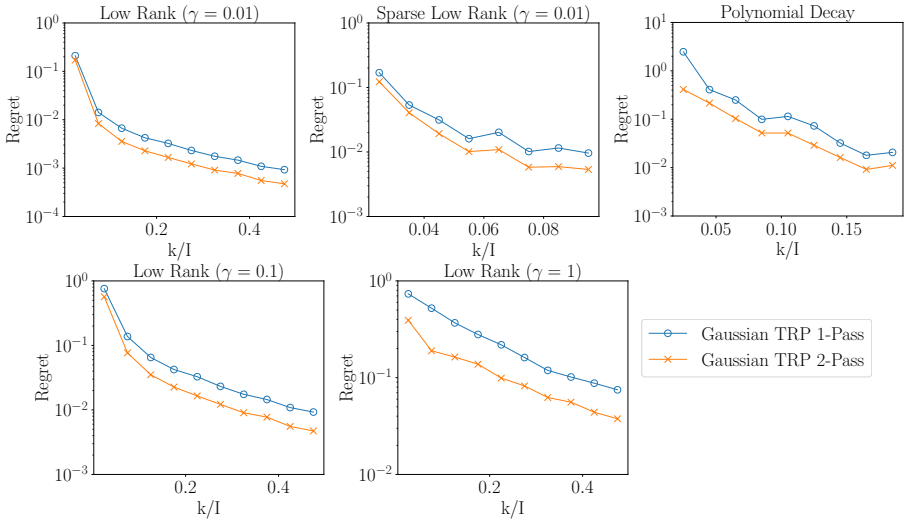


Fig. 12: We approximate 3D synthetic tensors (see subsection 6.1) with $I = 200$, using our one-pass and two-pass algorithms with $r = 5$ and varying k ($s = 2k + 1$), using the Gaussian TRP in the Tucker sketch.

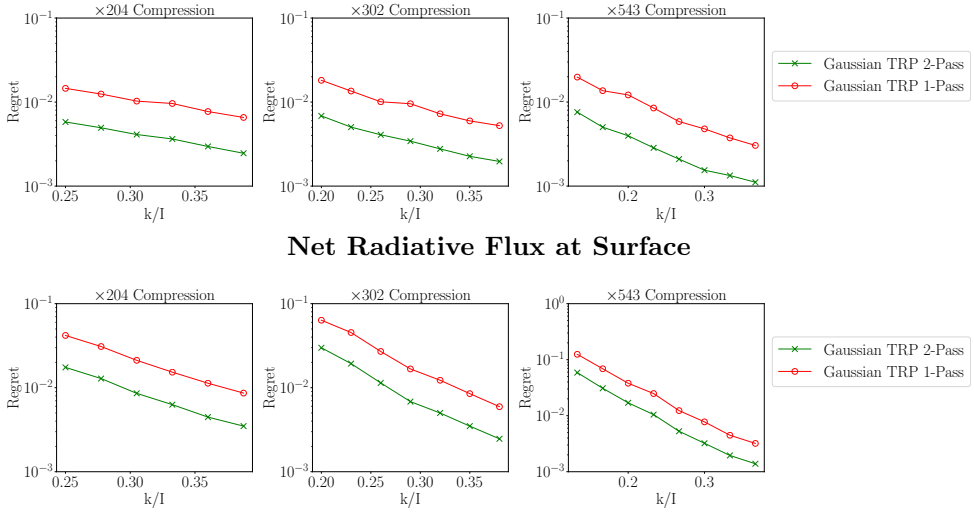


Fig. 13: We approximate the net radiative flux and dust aerosol burden data using our one-pass and two-pass algorithms using Gaussian TRP. We compare the performance under different ranks ($r/I = 0.125, 0.2, 0.067$). The dataset comes from the CESM CAM. The dust aerosol burden measures the amount of aerosol contributed by the dust. The net radiative flux determines the energy received by the earth surface through radiation.