More Efficient Sampling for Tensor Decomposition With Worst-Case Guarantees

Osman Asif Malik [1]

Abstract

Recent papers have developed alternating least squares (ALS) methods for CP and tensor ring decomposition with a per-iteration cost which is sublinear in the number of input tensor entries for low-rank decomposition. However, the per-iteration cost of these methods still has an exponential dependence on the number of tensor modes when parameters are chosen to achieve certain worst-case guarantees. In this paper, we propose sampling-based ALS methods for the CP and tensor ring decompositions whose cost does not have this exponential dependence, thereby significantly improving on the previous state-of-the-art. We provide a detailed theoretical analysis and also apply the methods in a feature extraction experiment.

1. Introduction

Tensor decomposition has recently emerged as an important tool in machine learning and data mining (Papalexakis et al., 2016; Cichocki et al., 2016; 2017; Ji et al., 2019). Examples of applications include parameter reduction in neural networks (Novikov et al., 2015; Garipov et al., 2016; Yang et al., 2017; Yu et al., 2017; Ye et al., 2018), understanding expressiveness of deep neural networks (Cohen et al., 2016; Khrulkov et al., 2018), supervised learning (Stoudenmire & Schwab, 2017; Novikov et al., 2016), filter learning (Hazan et al., 2005; Rigamonti et al., 2013), image factor analysis and recognition (Vasilescu & Terzopoulos, 2002; Liu et al., 2019), multimodal feature fusion (Hou et al., 2019), natural language processing (Lei et al., 2014), feature extraction (Bengua et al., 2015), and tensor completion (Wang et al., 2017). Due to their multidimensional nature, tensors are inherently plagued by the curse of dimensionality. Indeed, simply storing an N-way tensor with each dimension equal to I requires I^N numbers. Tensors are also fundamentally more difficult to decompose than matrices (Hillar & Lim, 2013). Many tensor decompositions correspond to difficult non-convex optimization problems. A popular approach for tackling these optimization problems is to use alternating least squares (ALS). While ALS works well for smaller tensors, the per-iteration cost for an N-way tensor of size $I \times \cdots \times I$ is $\Omega(I^N)$ since each iteration requires solving a number of least squares problems with the data tensor entries as the dependent variables. To address this issue, several recent works have developed sampling-based ALS methods for the CP decomposition (Cheng et al., 2016; Larsen & Kolda, 2020) and tensor ring decomposition (Malik & Becker, 2021). When the target rank is small enough, they have a per-iteration cost which is *sublinear* in the number of input tensor entries while still retaining approximation guarantees for each least squares solve with high probability. However, the cost of these methods still has an exponential dependence on N: $\Omega(R^{N+1})$ for the CP decomposition and $\Omega(R^{2N+2})$ for the tensor ring decomposition, where R is the relevant notion of rank. Unlike matrix rank, both the CP and tensor ring ranks of a tensor can exceed the mode dimension I, in which case the previous methods would no longer have sublinear per-iteration cost. This leads us to the following question:

> Can we construct ALS algorithms for tensor decomposition with a per-iteration cost *which does not depend exponentially on* N and which has guarantees for each least squares solve?

In this paper, we show that this is indeed possible for both the CP and tensor ring[1] decompositions with high probability relative error guarantees. Like the previous works mentioned above, we also use approximate leverage score sampling. Unlike those previous works which use quite coarse approximations to the leverage scores, we are able to sample from a distribution which is much closer to the exact one. We do this by using ideas for fast leverage score estimation from Drineas et al. (2012) combined with the recently developed recursive sketch by Ahle et al. (2020). We also design sampling schemes for both the CP and tensor ring decompositions which allow us to avoid computing the whole sampling distribution which otherwise would cost

[1]Applied Mathematics & Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, USA. Correspondence to: Osman Asif Malik <oamalik@lbl.gov>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

[1]Our results are also relevant for the popular tensor train decomposition (Oseledets, 2010; 2011) since it is a special case of the tensor ring decomposition.

$\Omega(I^{N-1})$. We provide a detailed theoretical analysis and run various experiments including one on feature extraction.

When theoretical guarantees are required our methods scale much better with tensor order than the previous methods. However, these benefits only occur when the tensor order is sufficiently high. We therefore expect our methods to provide benefits mainly when decomposing higher order tensors.

2. Related Work

CP Decomposition Cheng et al. (2016) propose SPALS, the first ALS algorithm for CP decomposition with a per-iteration cost sublinear in the number of input tensor entries. They use leverage scores sampling to speed up computation of the matricized-tensor-times-Khatri–Rao product, a key kernel which arises in the ALS algorithm for CP decomposition. Larsen & Kolda (2020) propose CP-ARLS-LEV which uses leverage score sampling to reduce the size of the least squares problems in the ALS algorithm for CP decomposition. In addition to several practical algorithmic improvements, their relative error guarantees improve on the weaker additive error guarantees provided by Cheng et al. (2016). Other papers that develop randomized algorithms for the CP decomposition include Wang et al. (2015), Battaglino et al. (2018), Yang et al. (2018) and Aggour et al. (2020). There are also works that use both conventional and stochastic optimization approaches (Sorber et al., 2012; 2013; Kolda & Hong, 2020).

Tensor Ring Decomposition Yuan et al. (2019a) develop a randomized method for the tensor ring decomposition which first compresses the input tensor by applying Gaussian sketches to each mode. The compressed tensor is then decomposed using standard deterministic decomposition algorithms. This decomposition is then combined with the sketches to get a decomposition of the original tensor. Ahmadi-Asl et al. (2020) develop several randomized variants of the deterministic TR-SVD algorithm by replacing the SVDs with their randomized counterpart. Malik & Becker (2021) propose TR-ALS-Sampled which is an ALS algorithm with a per-iteration cost sublinear in the number of input tensor entries. It uses leverage score sampling to reduce the size of the least squares problems in the standard ALS algorithm. Other works that develop randomized methods for tensor ring decomposition include Espig et al. (2012) and Khoo et al. (2019).

Other Works on Tensor Decomposition Papers that develop randomized methods for other tensor decompositions include the works by Drineas & Mahoney (2007), Tsourakakis (2010), da Costa et al. (2016), Malik & Becker (2018), Sun et al. (2020), Minster et al. (2020) and Fahrbach et al. (2021) for the Tucker decomposition; Biagioni et al. (2015) and Malik & Becker (2020a) for the tensor interpolative decomposition; Zhang et al. (2018) and Tarzanagh & Michailidis (2018) for t-product-based decompositions; and Huber et al. (2017) and Che & Wei (2019) for the tensor train decomposition. Papers that use skeleton approximation and other sampling-based techniques include those by Mahoney et al. (2008), Oseledets et al. (2008), Oseledets & Tyrtyshnikov (2010), Caiafa & Cichocki (2010) and Friedland et al. (2011).

Sketching and Sampling A large body of research has been generated over the last two decades focusing on sketching and sampling techniques in numerical linear algebra; see, e.g., the review papers by Halko et al. (2011), Mahoney (2011), Woodruff (2014) and Martinsson & Tropp (2020). Of particular relevance to our work are those papers that develop sketching and sampling techniques for efficient application to matrices whose columns have Kronecker product structure. These include sketches with particular row structure (Biagioni et al., 2015; Sun et al., 2018; Rakhshan & Rabusseau, 2020; 2021; Iwen et al., 2021), the Kronecker fast Johnson–Lindenstrauss transform (Battaglino et al., 2018; Jin et al., 2020; Malik & Becker, 2020b; Bamberger et al., 2021), TensorSketch (Pagh, 2013; Pham & Pagh, 2013; Avron et al., 2014; Diao et al., 2018), sampling-based sketches (Cheng et al., 2016; Diao et al., 2019; Larsen & Kolda, 2020; Fahrbach et al., 2021), and recursive sketches (Ahle et al., 2020).

3. Preliminaries

By tensor, we mean a multidimensional array containing real numbers. We will refer to a tensor with N indices as an N-way or mode-N tensor. We use bold Euler script letters (e.g., $\boldsymbol{\mathcal{X}}$) for tensors with three or more modes, bold uppercase letters (e.g., $\boldsymbol{A}$) for matrices, bold lowercase letters (e.g., $\boldsymbol{x}$) for vectors, and lowercase regular letters (e.g., x) for scalars. We indicate specific entries of objects with parentheses. For example, $\boldsymbol{\mathcal{X}}(i,j,k)$ is the entry on position (i,j,k) in $\boldsymbol{\mathcal{X}}$, and $\boldsymbol{x}(i)$ is the ith entry in $\boldsymbol{x}$. A colon denotes all elements along a certain mode. For example, $\boldsymbol{\mathcal{X}}(:,k,:)$ is the kth lateral slice of $\boldsymbol{\mathcal{X}}$, and $\boldsymbol{X}(i,:)$ is the ith row of $\boldsymbol{X}$. We will sometimes use superscripts in parentheses to denote a sequence of objects (e.g., $\boldsymbol{A}^{(1)},\dots,\boldsymbol{A}^{(N)}$). For a positive integer n, we use the notation $[n] \stackrel{\text{def}}{=} \{1,\dots,n\}$. We use $\otimes$ and $\odot$ to denote the Kronecker and Khatri–Rao products, respectively (defined in Section A). By *compact SVD*, we mean an SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ where $\boldsymbol{\Sigma} \in \mathbb{R}^{\operatorname{rank}(\boldsymbol{A})\times\operatorname{rank}(\boldsymbol{A})}$ and $\boldsymbol{U},\boldsymbol{V}$ have $\operatorname{rank}(\boldsymbol{A})$ columns. The ith canonical basis vector is denoted by $\boldsymbol{e}_i$. We denote the indicator of a random event A by $\operatorname{Ind}\{A\}$, which is 1 if A occurs and 0 otherwise. For indices $i_1 \in [I_1],\dots,i_N \in [I_N]$, the notation $\overline{i_1\cdots i_N} \stackrel{\text{def}}{=}$

$1 + \sum_{n=1}^{N}(i_n - 1)\prod_{j=1}^{n-1} I_j$ will be useful when working with unfolded tensors. See Section A for definitions of the asymptotic notation we use.

Definition 1. The *classical mode-n unfolding* of $\mathcal{X}$ is the matrix $\boldsymbol{X}_{(n)} \in \mathbb{R}^{I_n \times \Pi_{j\neq n} I_j}$ defined elementwise via

$$\boldsymbol{X}_{(n)}(i_n, \overline{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N}) \stackrel{\text{def}}{=} \mathcal{X}(i_1, \ldots, i_N). \quad (1)$$

The *mode-n unfolding* of $\mathcal{X}$ is the matrix $\boldsymbol{X}_{[n]} \in \mathbb{R}^{I_n \times \Pi_{j\neq n} I_j}$ defined elementwise via

$$\boldsymbol{X}_{[n]}(i_n, \overline{i_{n+1} \cdots i_N i_1 \cdots i_{n-1}}) \stackrel{\text{def}}{=} \mathcal{X}(i_1, \ldots, i_N). \quad (2)$$

3.1. Tensor Decomposition

We first introduce the CP decomposition. Consider an N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$. A rank-$R$ *CP decomposition* of $\mathcal{X}$ is of the form

$$\mathcal{X}(i_1, \ldots, i_N) = \sum_{r=1}^{R} \prod_{j=1}^{N} \boldsymbol{A}^{(j)}(i_j, r), \quad (3)$$

where each $\boldsymbol{A}^{(j)} \in \mathbb{R}^{I_j \times R}$ is called a *factor matrix*. We use $\mathrm{CP}(\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)})$ to denote the tensor in (3). The problem of computing a rank-R CP decomposition of a data tensor $\mathcal{X}$ can be formulated as

$$\operatorname*{arg\,min}_{\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}} \| \mathrm{CP}(\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}) - \mathcal{X} \|_{\mathrm{F}}. \quad (4)$$

Unfortunately, this problem is non-convex and difficult to solve exactly. ALS is the "workhorse" algorithm for solving this problem approximately (Kolda & Bader, 2009). With ALS, we consider the objective in (4), but only solve with respect to one of the factor matrices at a time while keeping the others fixed:

$$\operatorname*{arg\,min}_{\boldsymbol{A}^{(n)}} \| \mathrm{CP}(\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}) - \mathcal{X} \|_{\mathrm{F}}. \quad (5)$$

The problem in (5) can be rewritten as the linear least squares problem

$$\operatorname*{arg\,min}_{\boldsymbol{A}^{(n)}} \| \boldsymbol{A}^{\neq n} \boldsymbol{A}^{(n)\top} - \boldsymbol{X}_{(n)}^{\top} \|_{\mathrm{F}}, \quad (6)$$

where $\boldsymbol{A}^{\neq n} \in \mathbb{R}^{(\Pi_{j\neq n} I_j) \times R}$ is defined as

$$\boldsymbol{A}^{\neq n} \stackrel{\text{def}}{=} \boldsymbol{A}^{(N)} \odot \cdots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \cdots \odot \boldsymbol{A}^{(1)}. \quad (7)$$

By repeatedly updating each factor matrix one at a time via (6), we get the standard CP-ALS algorithm outlined in Algorithm 1. For further details on the CP decomposition, see Kolda & Bader (2009).

Next, we introduce the tensor ring decomposition. For $n \in [N]$, let $\mathcal{G}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ be 3-way tensors with $R_0 = R_N$. A rank-$(R_1, \ldots, R_N)$ *tensor ring decomposition* of $\mathcal{X}$ is of the form

$$\mathcal{X}(i_1, \ldots, i_N) = \sum_{r_1, \ldots, r_N} \prod_{n=1}^{N} \mathcal{G}^{(n)}(r_{n-1}, i_n, r_n), \quad (8)$$

where each r_n in the sum goes from 1 to R_n and $r_0 = r_N$, and each $\mathcal{G}^{(n)}$ is called a *core tensor*. We use $\mathrm{TR}(\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)})$ to denote the tensor in (8). Finding the best possible rank-$(R_1, \ldots, R_N)$ tensor ring decomposition of a tensor $\mathcal{X}$ is difficult. With an ALS approach we can update a single core tensor at a time by solving the following problem:

$$\operatorname*{arg\,min}_{\mathcal{G}^{(n)}} \| \mathrm{TR}(\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)}) - \mathcal{X} \|_{\mathrm{F}}. \quad (9)$$

To reformulate this problem into a linear least squares problem we will need the following definition.

Definition 2. By merging all cores except the nth, we get a *subchain tensor* $\mathcal{G}^{\neq n} \in \mathbb{R}^{R_n \times (\Pi_{j\neq n} I_j) \times R_{n-1}}$ defined elementwise via

$$\mathcal{G}^{\neq n}(r_n, \overline{i_{n+1} \ldots i_N i_1 \ldots i_{n-1}}, r_{n-1}) \stackrel{\text{def}}{=} \sum_{\substack{r_1, \ldots, r_{n-2} \\ r_{n+1}, \ldots, r_N}} \prod_{\substack{j=1 \\ j\neq n}}^{N} \mathcal{G}^{(j)}(r_{j-1}, i_j, r_j). \quad (10)$$

The problem in (9) can now be written as the linear least squares problem

$$\mathcal{G}^{(n)} = \operatorname*{arg\,min}_{\mathcal{G}} \| \boldsymbol{G}_{[2]}^{\neq n} \boldsymbol{G}_{(2)}^{\top} - \boldsymbol{X}_{[n]}^{\top} \|_{\mathrm{F}}. \quad (11)$$

By repeatedly updating each core tensor one at a time via (11), we get the standard TR-ALS algorithm outlined in Algorithm 2. For further details on the tensor ring decomposition, see Zhao et al. (2016).

Algorithm 1: CP-ALS

Input: $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, rank R
Output: Factor matrices $\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}$
Initialize factor matrices $\boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}$
while *termination criteria not met* **do**
  for $n = 1, \ldots, N$ **do**
    $\boldsymbol{A}^{(n)} = \arg\min_{\boldsymbol{A}} \| \boldsymbol{A}^{\neq n} \boldsymbol{A}^{\top} - \boldsymbol{X}_{(n)}^{\top} \|_{\mathrm{F}}$
return $\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}$

3.2. Recursive Sketching

Ahle et al. (2020) present two variants of their recursive sketch. The first one, which we will use, combines CountSketch and TensorSketch into a single sketch which can be

Algorithm 2: TR-ALS

Input: $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, ranks $R_1, \ldots, R_N$
Output: Core tensors $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)}$
Initialize core tensors $\mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(N)}$
while *termination criteria not met* **do**
  for $n = 1, \ldots, N$ **do**
    $\mathcal{G}^{(n)} = \arg\min_{\mathcal{G}} \|\boldsymbol{G}_{[2]}^{\neq n} \boldsymbol{G}_{(2)}^{\top} - \boldsymbol{X}_{[n]}^{\top}\|_{\mathrm{F}}$
return $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)}$

applied efficiently to Kronecker structured vectors. CountSketch was first introduced by Charikar et al. (2004) and extended to the linear algebra setting by Clarkson & Woodruff (2017). Recall that a function $h : [I] \to [J]$ is said to be *k-wise independent* if it is chosen from a family of functions such that for any k distinct $i_1, \ldots, i_k \in [I]$ the values $h(i_1), \ldots, h(i_k)$ are independent random variables uniformly distributed in $[J]$ (Pagh, 2013).

Definition 3. Let $h : [I] \to [J]$ and $s : [I] \to \{-1, +1\}$ be 3- and 4-wise independent functions, respectively. The *CountSketch* matrix $\boldsymbol{C} \in \mathbb{R}^{J \times I}$ is defined elementwise via $\boldsymbol{C}(j, i) \stackrel{\text{def}}{=} s(i) \cdot \mathrm{Ind}\{h(i) = j\}$.

The TensorSketch was developed in a series of papers by Pagh (2013), Pham & Pagh (2013), Avron et al. (2014) and Diao et al. (2018).

Definition 4. Let $h_1, h_2 : [I] \to [J]$ and $s_1, s_2 : [I] \to \{-1, +1\}$ be 3- and 4-wise independent functions, respectively. Define $h : [I] \times [I] \to [J]$ via

$$h(i_1, i_2) \stackrel{\text{def}}{=} (h_1(i_1) + h_2(i_2) \mod J) + 1. \tag{12}$$

The *degree-two TensorSketch* matrix $\boldsymbol{T} \in \mathbb{R}^{J \times I^2}$ is defined elementwise via

$$\boldsymbol{T}(j, \overline{i_1 i_2}) \stackrel{\text{def}}{=} s(i_1) s(i_2) \cdot \mathrm{Ind}\{h(i_1, i_2) = j\}. \tag{13}$$

We are now ready to describe the recursive sketch of Ahle et al. (2020). It is easiest to understand it if we consider its application to Kronecker structured vectors. Consider $\boldsymbol{x} = \boldsymbol{x}_1 \otimes \cdots \otimes \boldsymbol{x}_N \in \mathbb{R}^{I_1 \cdots I_N}$, where each $\boldsymbol{x}_n \in \mathbb{R}^{I_n}$.[2] Suppose first that $N = 2^q$ is a power of 2. The first step of the recursive sketch is to apply an independent CountSketch matrix $\boldsymbol{C}_n \in \mathbb{R}^{J \times I_n}$ to each $\boldsymbol{x}_n$:

$$\boldsymbol{y}_n^{(0)} \stackrel{\text{def}}{=} \boldsymbol{C}_n \boldsymbol{x}_n \in \mathbb{R}^J, \quad n \in [N]. \tag{14}$$

The vectors $\boldsymbol{y}_n^{(0)}$ are then combined pairwise using independent degree-two TensorSketches $\boldsymbol{T}_n^{(1)} \in \mathbb{R}^{J \times J^2}$:

$$\boldsymbol{y}_n^{(1)} \stackrel{\text{def}}{=} \boldsymbol{T}_n^{(1)}(\boldsymbol{y}_{2n-1}^{(0)} \otimes \boldsymbol{y}_{2n}^{(0)}), \quad n \in [N/2]. \tag{15}$$

[2] Ahle et al. (2020) consider the case when each $\boldsymbol{x}_n$ has the same length. We consider a slightly more general definition here since this allows us to work with tensors whose modes are of different size.

This process is then repeated: At each step, pairs of length-J vectors are combined using independent TensorSketches of size $J \times J^2$. The mth step is

$$\boldsymbol{y}_n^{(m)} \stackrel{\text{def}}{=} \boldsymbol{T}_n^{(m)}(\boldsymbol{y}_{2n-1}^{(m-1)} \otimes \boldsymbol{y}_{2n}^{(m-1)}), \quad n \in [N/2^m]. \tag{16}$$

When $m = q$, we are left with a single vector $\boldsymbol{y}_1^{(q)} \in \mathbb{R}^J$. The mapping $\boldsymbol{x} \mapsto \boldsymbol{y}_1^{(q)}$, which we denote by $\boldsymbol{\Psi}_{J,(I_n)_{n=1}^{2^q}}$, is the recursive sketch. If N is not a power of 2, we choose $q \stackrel{\text{def}}{=} \lceil \log_2 N \rceil$ and define the recursive sketch as $\boldsymbol{\Psi}_{J,(I_n)_{n=1}^{N}} : \boldsymbol{x} \mapsto \boldsymbol{\Psi}_{J,(\tilde{I}_n)_{n=1}^{2^q}}(\boldsymbol{x} \otimes \boldsymbol{e}_1^{\otimes(2^q - N)})$, where $\boldsymbol{e}_1$ is the first canonical basis vector of length $I_{\max} \stackrel{\text{def}}{=} \max_{n \in [N]} I_n$, and each $\tilde{I}_n \stackrel{\text{def}}{=} I_n$ for $n \leq N$ and $\tilde{I}_n \stackrel{\text{def}}{=} I_{\max}$ if $n > N$. We refer to $\boldsymbol{\Psi}_{J,(I_n)_{n=1}^{N}}$ as a $(J, (I_n)_{n=1}^{N})$-recursive sketch. It is in fact linear, and when $N = 2^q$ we can write $\boldsymbol{\Psi}_{J,(I_n)_{n=1}^{N}}$ as a product of $q + 1$ matrices:

$$\boldsymbol{\Psi}_{J,(I_n)_{n=1}^{N}} = \boldsymbol{T}^{(q)} \boldsymbol{T}^{(q-1)} \cdots \boldsymbol{T}^{(1)} \boldsymbol{C}, \tag{17}$$

where $\boldsymbol{C} \stackrel{\text{def}}{=} \bigotimes_{n=1}^{N} \boldsymbol{C}_n$ is a $J^N \times \prod_n I_n$ matrix and $\boldsymbol{T}^{(m)} \stackrel{\text{def}}{=} \bigotimes_{n=1}^{2^{q-m}} \boldsymbol{T}_n^{(m)}$ is a $J^{2^{q-m}} \times J^{2^{q-m+1}}$ matrix. Figure 1 illustrates the recursive sketch for $N = 4$.

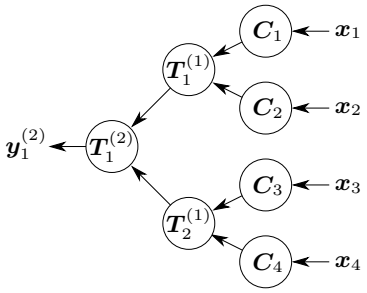

Figure 1. Illustration of the recursive sketch applied to a vector $\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \boldsymbol{x}_3 \otimes \boldsymbol{x}_4$. This is an adaption of Figure 1 in Ahle et al. (2020).

The recursive sketch is a subspace embedding with high probability.

Definition 5. A matrix $\boldsymbol{\Psi} \in \mathbb{R}^{J \times I}$ is called a γ-*subspace embedding* for a matrix $\boldsymbol{A} \in \mathbb{R}^{I \times R}$ if

$$\left| \|\boldsymbol{\Psi} \boldsymbol{A} \boldsymbol{x}\|_2^2 - \|\boldsymbol{A} \boldsymbol{x}\|_2^2 \right| \leq \gamma \|\boldsymbol{A} \boldsymbol{x}\|_2^2 \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^R. \tag{18}$$

The recursive sketch has the remarkable feature that the embedding dimension required for subspace embedding guarantees does not depend exponentially on N. See Theorem 1 in Ahle et al. (2020) or Theorem 17 for a precise statement.

3.3. Leverage Score Sampling

Leverage score sampling is a popular technique for a variety of problems in numerical linear algebra. For an in-depth

discussion, see Mahoney (2011) and Woodruff (2014).

Definition 6. Let $\boldsymbol{A} \in \mathbb{R}^{I \times R}$ and suppose $\boldsymbol{U} \in \mathbb{R}^{I \times \operatorname{rank}(\boldsymbol{A})}$ contains the left singular vectors of $\boldsymbol{A}$. The ith *leverage score* of $\boldsymbol{A}$ is defined as $\ell_i(\boldsymbol{A}) \stackrel{\text{def}}{=} \|\boldsymbol{U}(i,:)\|_2^2$ for $i \in [I]$.

Definition 7. Let $\boldsymbol{q} \in \mathbb{R}^I$ be a probability distribution and let $f : [J] \to [I]$ be a random map such that each $f(j)$ is independent and distributed according to $\boldsymbol{q}$. Define $\boldsymbol{S} \in \mathbb{R}^{J \times I}$ elementwise via

$$\boldsymbol{S}(j,i) \stackrel{\text{def}}{=} \operatorname{Ind}\{f(j) = i\} / \sqrt{J \boldsymbol{q}(f(j))}. \tag{19}$$

We call $\boldsymbol{S}$ a *sampling matrix with parameters* $(J, \boldsymbol{q})$, or $\boldsymbol{S} \sim \mathcal{D}(J, \boldsymbol{q})$ for short. Let $\boldsymbol{A} \in \mathbb{R}^{I \times R}$ be nonzero and suppose $\beta \in (0,1]$. Define the distribution $\boldsymbol{p} \in \mathbb{R}^I$ via $\boldsymbol{p}(i) \stackrel{\text{def}}{=} \ell_i(\boldsymbol{A}) / \operatorname{rank}(\boldsymbol{A})$. We say that $\boldsymbol{S} \sim \mathcal{D}(J, \boldsymbol{q})$ is a *leverage score sampling matrix for* $(\boldsymbol{A}, \beta)$ if $\boldsymbol{q}(i) \geq \beta \boldsymbol{p}(i)$ for all $i \in [I]$.

For a least squares problem $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2$ where $\boldsymbol{A} \in \mathbb{R}^{I \times R}$ has many more rows than columns, we can use sampling to reduce the size of the problem to $\min_{\boldsymbol{x}} \|\boldsymbol{S}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{S}\boldsymbol{y}\|_2$. We would ideally like to sample according to the distribution $\boldsymbol{p}$ in Definition 7, but this requires computing $\boldsymbol{U}$ in Definition 6 (e.g., via the SVD or QR decomposition) which costs $O(IR^2)$. This is the same cost as solving the full least squares problem and is therefore too expensive. However, as shown by Drineas et al. (2012), the leverage scores can be accurately estimated in less time. Theorem 8 is a variant of Lemma 9 by Drineas et al. (2012). They consider the case when $\boldsymbol{\Psi}$ is a fast Johnson–Lindenstrauss transform instead of a subspace embedding.

Theorem 8. *Let $\boldsymbol{A} \in \mathbb{R}^{I \times R}$ where $I > R$, $\gamma \in (0,1)$ and suppose $\boldsymbol{\Psi}\boldsymbol{A} = \boldsymbol{U}_1 \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^\top$ is a compact SVD. Define*

$$\tilde{\ell}_i(\boldsymbol{A}) \stackrel{\text{def}}{=} \|\boldsymbol{e}_i^\top \boldsymbol{A} \boldsymbol{V}_1 \boldsymbol{\Sigma}_1^{-1}\|_2^2. \tag{20}$$

Suppose that $\boldsymbol{\Psi}$ is a γ-subspace embedding for $\boldsymbol{A}$. Then

$$|\ell_i(\boldsymbol{A}) - \tilde{\ell}_i(\boldsymbol{A})| \leq \frac{\gamma}{1-\gamma} \ell_i(\boldsymbol{A}) \quad \textit{for all } i \in [I]. \tag{21}$$

A proof of Theorem 8 appears in Section B.1.

4. Efficient Sampling for Tensor Decomposition

In this section we present our proposed sampling schemes for the CP and tensor ring decompositions of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$. We will refer to these methods as CP-ALS-ES and TR-ALS-ES, respectively, where "ES" is short for "Efficient Sampling."

4.1. CP Decomposition

Each least squares solve on line 4 in Algorithm 1 involves all entries in $\boldsymbol{\mathcal{X}}$. To reduce the size of this problem, we sample rows according to an approximate leverage score distribution computed as in Theorem 8 with $\boldsymbol{\Psi}$ chosen to be a recursive sketch. Theorem 9 shows that such a sampling approach yields relative error guarantees for the CP-ALS least squares problem.

Theorem 9. *Let $\boldsymbol{A}^{\neq n}$ be defined as in (7). Define the vector $\boldsymbol{v} \stackrel{\text{def}}{=} \begin{bmatrix} N, & \cdots & n+1, & n-1, & \cdots & 1 \end{bmatrix}$ and suppose $\varepsilon, \delta \in (0,1)$. Suppose the estimates $\tilde{\ell}_i(\boldsymbol{A}^{\neq n})$ are computed as in Theorem 8, with $\boldsymbol{\Psi} \in \mathbb{R}^{J_1 \times \Pi_{j \neq n} I_j}$ chosen to be a $(J_1, (I_{\boldsymbol{v}(j)})_{j=1}^{N-1})$-recursive sketch. Moreover, suppose $\boldsymbol{S} \in \mathbb{R}^{J_2 \times \Pi_{j \neq n} I_j}$ is a sampling matrix with parameters $(J_2, \boldsymbol{q})$ where $\boldsymbol{q}(i) \propto \tilde{\ell}_i(\boldsymbol{A}^{\neq n})$. If*

$$J_1 \gtrsim N R^2 / \delta, \tag{22}$$

$$J_2 \gtrsim R \max\big(\log(R/\delta), 1/(\varepsilon\delta)\big), \tag{23}$$

then $\tilde{\boldsymbol{A}} \stackrel{\text{def}}{=} \arg\min_{\boldsymbol{A}} \|\boldsymbol{S}\boldsymbol{A}^{\neq n}\boldsymbol{A}^\top - \boldsymbol{S}\boldsymbol{X}_{(n)}^\top\|_{\mathrm{F}}$ satisfies the following with probability at least $1 - \delta$:

$$\|\boldsymbol{A}^{\neq n}\tilde{\boldsymbol{A}}^\top - \boldsymbol{X}_{(n)}^\top\|_{\mathrm{F}} \leq (1+\varepsilon) \min_{\boldsymbol{A}} \|\boldsymbol{A}^{\neq n}\boldsymbol{A}^\top - \boldsymbol{X}_{(n)}^\top\|_{\mathrm{F}}. \tag{24}$$

A proof of Theorem 9 is provided in Section B.2. It combines well-known results for leverage score sampling with an efficient leverage score estimation procedure. The estimation procedure follows ideas by Drineas et al. (2012) but uses the recursive sketch by Ahle et al. (2020) instead of the fast Johnson–Lindenstrauss transform that Drineas et al. use.

The dependence on R in (23) is optimal in the sense that it cannot be improved when rows are sampled i.i.d. (Dereziński & Warmuth, 2018). It is a significant improvement over the current state-of-the-art sampling-based ALS method by Larsen & Kolda (2020) which requires $O(R^{N-1} \max(\log(R/\delta), 1/(\delta\varepsilon)))$ samples to achieve relative error guarantees. The method by Cheng et al. (2016) requires $O(R^N \log(I_n/\delta)/\varepsilon^2)$ samples and only achieves weaker additive error guarantees.

In Sections 4.1.1 and 4.1.2 we discuss how to compute the approximate solution $\tilde{\boldsymbol{A}}$ in Theorem 9 efficiently. In Section 4.1.3 we compare the complexity of our method to that of other CP decomposition methods.

4.1.1. Step 1: Computing $\boldsymbol{\Psi}\boldsymbol{A}^{\neq n}$

The columns of $\boldsymbol{A}^{\neq n}$ are Kronecker products, so applying the recursive sketch $\boldsymbol{\Psi}$ to $\boldsymbol{A}^{\neq n}$ efficiently is straightforward. Let $q \stackrel{\text{def}}{=} \lceil \log_2(N-1) \rceil$. First, independent CountSketches $\boldsymbol{C}_j$ with J_1 rows and an appropriate number of columns are applied:

$$\boldsymbol{Y}_j^{(0)} \stackrel{\text{def}}{=} \begin{cases} \boldsymbol{C}_j \boldsymbol{A}^{(\boldsymbol{v}(j))} & \text{if } 1 \leq j \leq N-1, \\ \boldsymbol{C}_j \boldsymbol{e}_1 \boldsymbol{1}_{1 \times R} & \text{if } N-1 < j \leq 2^q, \end{cases} \tag{25}$$

where $\boldsymbol{v}$ is defined as in Theorem 9 and $\mathbf{1}_{1\times R}$ is a length-R row vector of ones. Then, independent TensorSketches are applied recursively:

$$\boldsymbol{Y}_j^{(m)} = \boldsymbol{T}_j^{(m)}(\boldsymbol{Y}_{2j-1}^{(m-1)} \odot \boldsymbol{Y}_{2j}^{(m-1)}), \quad j \in [2^{q-m}], \tag{26}$$

for $m = 1, \dots, q$, where each $\boldsymbol{T}_j^{(m)} \in \mathbb{R}^{J_1 \times J_1^2}$. The final output is $\boldsymbol{Y}_1^{(q)} = \boldsymbol{\Psi}\boldsymbol{A}^{\neq n}$.

4.1.2. Step 2: Drawing Samples Efficiently

Since a row index $i \in [\prod_{j\neq n} I_j]$ of $\boldsymbol{A}^{\neq n}$ can be written as $i = \overline{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N}$ where each $i_j \in [I_j]$, we can sample an index $i \in [\prod_{j\neq n} I_j]$ by sampling subindices $i_j \in [I_j]$ for each $j \neq n$. By sampling the subindices in sequence one after another we avoid computing all entries in $\boldsymbol{q}$ which otherwise would cost $\Omega(\prod_{j\neq n} I_j)$. We use an abbreviated notation to denote the probability of drawing subsequences of indices. For example, $\mathbb{P}(i_1)$ denotes the probability that the first index is i_1, and $\mathbb{P}((i_j)_{j\leq m, j\neq n})$ denotes the probability that the first m indices (excluding the nth) are $i_1, \dots, i_{n-1}, i_{n+1}, \dots, i_m$.

Lemma 10. *Let $\boldsymbol{\Psi}\boldsymbol{A}^{\neq n} = \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^\top$ be a compact SVD and define $\boldsymbol{\Phi} \stackrel{\text{def}}{=} \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1})^\top$. The normalization constant for the distribution $\boldsymbol{q}$ with $\boldsymbol{q}(i) \propto \tilde{\ell}_i(\boldsymbol{A}^{\neq n})$ is*

$$C \stackrel{\text{def}}{=} \sum_i \tilde{\ell}_i(\boldsymbol{A}^{\neq n}) = \sum_{r,k} \boldsymbol{\Phi}(r,k) \cdot \prod_{j\neq n} (\boldsymbol{A}^{(j)\top}\boldsymbol{A}^{(j)})(r,k). \tag{27}$$

The marginal probability of drawing $(i_j)_{j\leq m, j\neq n}$ is

$$\mathbb{P}((i_j)_{j\leq m, j\neq n}) = \frac{1}{C}\sum_{r,k} \boldsymbol{\Phi}(r,k) \cdot \Big(\prod_{\substack{j\leq m \\ j\neq n}} \boldsymbol{A}^{(j)}(i_j, r)\boldsymbol{A}^{(j)}(i_j, k)\Big)\Big(\prod_{\substack{j>m \\ j\neq n}} (\boldsymbol{A}^{(j)\top}\boldsymbol{A}^{(j)})(r,k)\Big). \tag{28}$$

In (27) and (28), the summations are over $i \in [\prod_{j\neq n} I_j]$ and $r, k \in [R]$.

The proof of Lemma 10 is given in Section B.3. We now describe the sampling procedure by first describing how to sample the first index i_1 (or i_2, if $n = 1$), followed by all subsequent indices.

Sampling First Index Suppose $n \neq 1$. We compute the probability of sampling i_1 for all $i_1 \in [I_1]$ via (28) and sample an index $i_1 \in [I_1]$ from that distribution. If $n = 1$, we do this for the second index i_2 instead.

Sampling Subsequent Indices After drawing i_1 (or i_2, if $n = 1$), all subsequent indices can be drawn one at a time *conditionally* on the previous indices. Suppose we have drawn indices $(i_j)_{j<m, j\neq n}$. The conditional distribution of i_m (or i_{m+1}, if $n = m$) given the previously drawn indices is then

$$\mathbb{P}(i_m \mid (i_j)_{j<m, j\neq n}) = \frac{\mathbb{P}((i_j)_{j\leq m, j\neq n})}{\mathbb{P}((i_j)_{j<m, j\neq n})}. \tag{29}$$

We compute the conditional probability in (29) for all $i_m \in [I_m]$ via (28) and draw a sample i_m from that distribution.

Once the J_2 samples in $[\prod_{j\neq n} I_j]$ have been drawn, the matrix $\boldsymbol{S}\boldsymbol{A}^{\neq n}$ can be computed without forming $\boldsymbol{A}^{\neq n}$. The matrix $\boldsymbol{S}\boldsymbol{X}_{(n)}^\top$ can be computed by extracting only J_2 rows from $\boldsymbol{X}_{(n)}^\top$.

4.1.3. Complexity Analysis

If J_1 and J_2 are chosen as in (22) and (23), and if we assume that $I_j = I$ for all $j \in [N]$ and ignore log factors, then the per-iteration complexity for our method CP-ALS-ES is $\tilde{O}(N^2R^3(R + NI/\varepsilon)/\delta)$. In Table 1, we compare this to the complexity of other ALS-based methods for CP decomposition (see Section 2). Our method is the only one that does not have an exponential per-iteration dependence on N. See Section C for a detailed complexity analysis.

Table 1. Comparison of leading order computational cost for various CP decomposition methods. We ignore log factors and assume that $I_j = I$ for all $j \in [N]$. #iter is the number of ALS iterations. SPALS has an additional upfront cost of $\mathrm{nnz}(\mathcal{X})$.

Method	Complexity
CP-ALS	#iter $\cdot\, N(N+I)I^{N-1}R$
SPALS	#iter $\cdot\, N(N+I)R^{N+1}/\varepsilon^2$
CP-ARLS-LEV	#iter $\cdot\, N(R+I)R^N/(\delta\varepsilon)$
CP-ALS-ES (our)	#iter $\cdot\, N^2R^3(R+NI/\varepsilon)/\delta$

4.2. Tensor Ring Decomposition

The least squares problem for TR-ALS on line 4 in Algorithm 2 also involves all entries in $\mathcal{X}$. We use an approach similar to that for the CP decomposition, which yields the following approximation guarantees for the TR-ALS least squares problem.

Theorem 11. *Let $\boldsymbol{G}_{[2]}^{\neq n}$ be the mode-2 unfolding of the subchain tensor $\mathcal{G}^{\neq n}$ (see Definitions 1 and 2). Define the vector $\boldsymbol{w} \stackrel{\text{def}}{=} [n-1, \ \cdots \ 1, \ N, \ \cdots \ n+1]$ and suppose $\varepsilon, \delta \in (0,1)$. Suppose the estimates $\tilde{\ell}_i(\boldsymbol{G}_{[2]}^{\neq n})$ are computed as in Theorem 8, with $\boldsymbol{\Psi} \in \mathbb{R}^{J_1 \times \Pi_{j\neq n} I_j}$ chosen to be a $(J_1, (I_{\boldsymbol{w}(j)})_{j=1}^{N-1})$-recursive sketch. Moreover, suppose $\boldsymbol{S} \in \mathbb{R}^{J_2 \times \Pi_{j\neq n} I_j}$ is a sampling matrix with parameters $(J_2, \boldsymbol{q})$ where $\boldsymbol{q}(i) \propto \tilde{\ell}_i(\boldsymbol{G}_{[2]}^{\neq n})$. If*

$$J_1 \gtrsim N(R_{n-1}R_n)^2/\delta, \tag{30}$$

$$J_2 \gtrsim R_{n-1}R_n \max\big(\log(R_{n-1}R_n/\delta), 1/(\varepsilon\delta)\big), \tag{31}$$

then $\tilde{\boldsymbol{G}} \stackrel{\text{def}}{=} \arg\min_{\boldsymbol{G}} \|\boldsymbol{S}\boldsymbol{G}_{[2]}^{\neq n}\boldsymbol{G}^\top - \boldsymbol{S}\boldsymbol{X}_{[n]}^\top\|_{\mathrm{F}}$ *satisfies the following with probability at least* $1-\delta$*:*

$$\|\boldsymbol{G}_{[2]}^{\neq n}\tilde{\boldsymbol{G}}^\top - \boldsymbol{X}_{[n]}^\top\|_{\mathrm{F}} \leq (1+\varepsilon)\min_{\boldsymbol{G}} \|\boldsymbol{G}_{[2]}^{\neq n}\boldsymbol{G}^\top - \boldsymbol{X}_{[n]}^\top\|_{\mathrm{F}}. \tag{32}$$

A proof of Theorem 11 is provided in Section B.4. The proof uses similar steps as the proof of Theorem 9. The main difference is the structure and number of columns of the least squares design matrix.

Since $\boldsymbol{G}_{[2]}^{\neq n}$ has $R_{n-1}R_n$ columns, the sample complexity in (31) has optimal rank dependence in the sense discussed in Section 4.1. This is a significant improvement over the current state-of-the-art sampling-based ALS method by Malik & Becker (2021) which requires $O((\prod_j R_j^2)\max(\log(R_{n-1}R_n/\delta)), 1/(\varepsilon\delta))$ samples to achieve relative error guarantees.

In Sections 4.2.1 and 4.2.2 we discuss how to compute the approximate solution $\tilde{\boldsymbol{G}}$ in Theorem 11 efficiently. In Section 4.2.3 we compare the complexity of our method to that of other tensor ring methods.

4.2.1. Step 1: Computing $\boldsymbol{\Psi}\boldsymbol{G}_{[2]}^{\neq n}$

Although $\boldsymbol{G}_{[2]}^{\neq n}$ has a more complicated structure than $\boldsymbol{A}^{\neq n}$, $\boldsymbol{\Psi}$ can still be applied efficiently to $\boldsymbol{G}_{[2]}^{\neq n}$. We describe a scheme for computing the column $\boldsymbol{\Psi}\boldsymbol{G}_{[2]}^{\neq n}(:, \overline{r_{n-1}r_n})$ below, and give a more detailed motivation in Section B.5. Let $q \stackrel{\text{def}}{=} \lceil \log_2(N-1) \rceil$. Define matrices $\boldsymbol{H}^{(j)}$ for $j \in [2^q]$ as follows: Let $\boldsymbol{H}^{(1)} \in \mathbb{R}^{I_{n-1}\times R_{n-2}}$ be a matrix with columns $\boldsymbol{H}^{(1)}(:,k) \stackrel{\text{def}}{=} \boldsymbol{G}_{[2]}^{(n-1)}(:, \overline{r_{n-1}k})$ for $k \in [R_{n-2}]$. Let $\boldsymbol{H}^{(j)} \stackrel{\text{def}}{=} \boldsymbol{G}_{[2]}^{(\boldsymbol{w}(j))} \in \mathbb{R}^{I_{\boldsymbol{w}(j)}\times R_{\boldsymbol{w}(j)}R_{\boldsymbol{w}(j)-1}}$ for $2 \leq j \leq N-2$. Let $\boldsymbol{H}^{(N-1)} \in \mathbb{R}^{I_{n+1}\times R_{n+1}}$ be a matrix with columns $\boldsymbol{H}^{(N-1)}(:,k) \stackrel{\text{def}}{=} \boldsymbol{G}_{[2]}^{(n+1)}(:, \overline{kr_n})$ for $k \in [R_{n+1}]$. Let $\boldsymbol{H}^{(j)} \stackrel{\text{def}}{=} \boldsymbol{e}_1 \in \mathbb{R}^{\max_{j\neq n} I_j}$ be a column vector for $N \leq j \leq 2^q$. Next, define

$$\boldsymbol{Y}_j^{(0)} \stackrel{\text{def}}{=} \boldsymbol{C}_j\boldsymbol{H}^{(j)}, \quad j \in [2^q], \tag{33}$$

$$K_j^{(0)} \stackrel{\text{def}}{=} \begin{cases} R_{\boldsymbol{w}(j)} & \text{if } 2 \leq j \leq N-1, \\ 1 & \text{if } j = 1 \text{ or } N \leq j \leq 2^q+1. \end{cases} \tag{34}$$

The TensorSketch matrices are then applied recursively as follows. For each $m = 1, \ldots, q$, compute

$$\boldsymbol{Y}_j^{(m)}(:, \overline{k_1k_3}) = \sum_{k_2 \in [K_{2j}^{(m-1)}]} \boldsymbol{T}_j^{(m)}\big(\boldsymbol{Y}_{2j-1}^{(m-1)}(:, \overline{k_1k_2}) \otimes \boldsymbol{Y}_{2j}^{(m-1)}(:, \overline{k_2k_3})\big) \tag{35}$$

for each $k_1 \in [K_{2j-1}^{(m-1)}], k_3 \in [K_{2j+1}^{(m-1)}], j \in [2^{q-m}]$. For each $m = 1, \ldots, q$, also compute

$$K_j^{(m)} \stackrel{\text{def}}{=} K_{2j-1}^{(m-1)}, \quad j \in [2^{q-m}+1]. \tag{36}$$

We prove the following in Section B.5.

Lemma 12. $\boldsymbol{Y}_1^{(q)}$ *satisfies* $\boldsymbol{Y}_1^{(q)} = \boldsymbol{\Psi}\boldsymbol{G}_{[2]}^{\neq n}(:, \overline{r_{n-1}r_n})$.

The entire matrix $\boldsymbol{\Psi}\boldsymbol{G}_{[2]}^{\neq n}$ can be computed by repeating the steps above for each column $\overline{r_{n-1}r_n} \in [R_{n-1}R_n]$.

4.2.2. Step 2: Drawing Samples Efficiently

The sampling approach for the tensor ring decomposition is similar to the approach for the CP decomposition which we described in Section 4.1.2.

Lemma 13. *Let* $\boldsymbol{\Psi}\boldsymbol{G}_{[2]}^{\neq n} = \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^\top$ *be a compact SVD and define* $\boldsymbol{\Phi} \stackrel{\text{def}}{=} \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1})^\top$*. The normalization constant for the distribution* $\boldsymbol{q}$ *with* $\boldsymbol{q}(i) \propto \tilde{\ell}_i(\boldsymbol{G}_{[2]}^{\neq n})$ *is*

$$C \stackrel{\text{def}}{=} \sum_i \tilde{\ell}_i(\boldsymbol{G}_{[2]}^{\neq n}) = \sum_{\substack{r_1,\ldots,r_N \\ k_1,\ldots,k_N}} \boldsymbol{\Phi}(\overline{r_{n-1}r_n}, \overline{k_{n-1}k_n}) \cdot \prod_{j\neq n} (\boldsymbol{G}_{[2]}^{(j)\top}\boldsymbol{G}_{[2]}^{(j)})(\overline{r_jr_{j-1}}, \overline{k_jk_{j-1}}). \tag{37}$$

The marginal probability of drawing $(i_j)_{j\leq m, j\neq n}$ *is*

$$\mathbb{P}((i_j)_{j\leq m, j\neq n}) = \frac{1}{C}\sum_{\substack{r_1,\ldots,r_N \\ k_1,\ldots,k_N}} \boldsymbol{\Phi}(\overline{r_{n-1}r_n}, \overline{k_{n-1}k_n}) \cdot \Big(\prod_{\substack{j\leq m \\ j\neq n}} \boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{r_jr_{j-1}})\boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{k_jk_{j-1}})\Big) \cdot \Big(\prod_{\substack{j>m \\ j\neq n}} (\boldsymbol{G}_{[2]}^{(j)\top}\boldsymbol{G}_{[2]}^{(j)})(\overline{r_jr_{j-1}}, \overline{k_jk_{j-1}})\Big). \tag{38}$$

In (37) *and* (38)*, the summations are over* $i \in [\prod_{j\neq n} I_j]$ *and* $r_j, k_j \in [R_j]$ *for each* $j \in [N]$.

The proof of Lemma 13 is given in Section B.6. The sampling procedure itself is the same as for the CP decomposition. The distribution $(\mathbb{P}(i_1))_{i_1=1}^{I_1}$ is computed via (38) and an index i_1 is sampled; if $n = 1$, these computations are done for i_2 instead of i_1. All subsequent indices are then sampled conditionally on the previous indices. This is done by computing the conditional distribution in (29) by using (38). The expression in (38) can be computed efficiently despite the exponential number of terms in the summation; see Remark 20 for details.

Once the J_2 samples in $[\prod_{j\neq n} I_j]$ have been drawn, the matrix $\boldsymbol{S}\boldsymbol{G}_{[2]}^{\neq n}$ can be computed without forming $\boldsymbol{G}_{[2]}^{\neq n}$. We describe this in detail in Remark 21. The matrix $\boldsymbol{S}\boldsymbol{X}_{[2]}^\top$ can be computed by extracting only J_2 rows from $\boldsymbol{X}_{[2]}^\top$.

4.2.3. Complexity Analysis

If J_1 and J_2 are chosen as in (30) and (31), and if we assume that $R_j = R$ and $I_j = I$ for all $j \in [N]$ and ignore log

factors, then the per-iteration complexity for our method TR-ALS-ES is $\tilde{O}(N^3R^9/\delta + N^3IR^8/(\varepsilon\delta))$. In Table 2, we compare this to the complexity of several other methods for tensor ring decomposition (see Section 2). rTR-ALS refers to the method by Yuan et al. (2019a) with each I compressed to K and with TR-ALS as the deterministic algorithm. TR-SVD-Rand refers to Algorithm 7 in Ahmadi-Asl et al. (2020). Our method is the only one that does not have an explicit exponential dependence on N. See Section C for a detailed complexity analysis.

Table 2. Comparison of leading order computational cost for various tensor ring decomposition methods. We ignore log factors and assume that $R_j = R$ and $I_j = I$ for all $j \in [N]$. #iter is the number of ALS iterations.

Method	Complexity
TR-ALS	#iter $\cdot\, NI^NR^2$
rTR-ALS	NI^NK + #iter $\cdot\, NK^NR^2$
TR-SVD	$I^{N+1} + I^NR^3$
TR-SVD-Rand	I^NR^2
TR-ALS-Sampled	#iter $\cdot\, NIR^{2N+2}/(\varepsilon\delta)$
TR-ALS-ES (our)	#iter $\cdot\, N^3R^8(R + I/\varepsilon)/\delta$

5. Experiments

The experiments are run in Matlab R2021b on a laptop computer with an Intel Core i7-1185G7 CPU and 32 GB of RAM. Our code is available at `https://github.com/OsmanMalik/TD-ALS-ES`. Additional experiment details are in Section D.

5.1. Sampling Distribution Comparison

We first compare the sampling distributions used by our methods with those used by the previous state-of-the-art—CP-ARLS-LEV by Larsen & Kolda (2020) for the CP decomposition and TR-ALS-Sampled by Malik & Becker (2021) for the tensor ring decomposition—when solving the least squares problems in (5) and (9). We run standard CP-ALS and TR-ALS on a real data tensor to get realistic factor matrices and core tensors when defining the design matrices $\boldsymbol{A}^{\neq n}$ and $\boldsymbol{G}_{[2]}^{\neq n}$. We get the real data tensor $\mathcal{X} \in \mathbb{R}^{16\times\cdots\times16}$ by reshaping a 4096×4096 gray scale image of a tabby cat into a 6-way tensor and then appropriately permuting the modes, a process called visual data tensorization (Yuan et al., 2019b). We then consider the least squares problems corresponding to an update of the 6th factor matrix or core tensor. As a performance measure, we compute the KL-divergence of the approximate distribution $\boldsymbol{q}$ from the exact leverage score sampling distribution $\boldsymbol{p}$ in Definition 7. Tables 3 and 4 report the results for different J_1 and ranks. The results show that our methods sample from a distribution much closer to the exact leverage score distribution when J_1 is as small as $J_1 = 1000$. See Figures 2–5 for a graphical comparison.

Table 3. KL-divergence (lower is better) of the approximated sampling distribution from the exact one for a CP-ALS least squares problem (5) with target rank R.

Method	$R = 10$	$R = 20$
CP-ARLS-LEV	0.2342	0.1853
CP-ALS-ES (J_1 = 1e+4)	0.0005	0.0006
CP-ALS-ES (J_1 = 1e+3)	0.0151	0.0070
CP-ALS-ES (J_1 = 1e+2)	0.1416	0.2173

Table 4. KL-divergence (lower is better) of the approximated sampling distribution from the exact one for a TR-ALS least squares problem (11) with target rank $(R, \ldots, R)$.

Method	$R = 3$	$R = 5$
TR-ALS-Sampled	0.3087	0.1279
TR-ALS-ES (J_1 = 1e+4)	0.0005	0.0007
TR-ALS-ES (J_1 = 1e+3)	0.0076	0.0070
TR-ALS-ES (J_1 = 1e+2)	0.1565	0.1831

5.2. Feature Extraction

Next, we run a benchmark feature extraction experiment on the COIL-100 image dataset (Nene et al., 1996) with a setup similar to that in Zhao et al. (2016) and Malik & Becker (2021). The data consists of 7200 color images of size 128×128 pixels, each belonging to one of 100 different classes. The data is arranged into a $128 \times 128 \times 3 \times 7200$ tensor which is decomposed using either a rank-25 CP decomposition or a rank-$(5, 5, 5, 5)$ tensor ring decomposition. The mode-4 factor matrix or core tensor is then used as a feature matrix in a k-NN algorithm with $k = 1$ and 10-fold cross validation. For our CP-ALS-ES, we use $J_1 = 10000$ and $J_2 = 2000$, and for our TR-ALS-ES we use $J_1 = 10000$ and $J_2 = 1000$. For the CP decomposition, we compare against CP-ALS in Tensor Toolbox (Bader & Kolda, 2006; Bader et al., 2021); CPD-ALS, CPD-MINF and CPD-NLS in Tensorlab (Vervliet et al., 2016); and our own implementation of CP-ARLS-LEV. For the tensor ring decomposition, we compare against the implementations of TR-ALS and TR-ALS-Sampled provided by Malik & Becker (2021). For CP-ARLS-LEV and TR-ALS-Sampled we use 2000 and 1000 samples, respectively. All iterative methods are run for 40 iterations.

Table 5 shows the average decomposition time, decomposition error, and classification accuracy for the various methods over 10 repetitions of the experiment[3]. For the CP decomposition, the two randomized methods are faster than the

[3]TR-ALS was only run once due to how long it takes to run.

competing methods. Our method takes about as long to run as CP-ARLS-LEV. This does not contradict our earlier analysis, which was a *worst-case* analysis. Real-world datasets are typically better behaved than the worst case, which is why CP-ARLS-LEV requires no more samples than CP-ALS-ES in this example. For the tensor ring decomposition, the randomized methods are substantially faster than the deterministic one. Our method is a bit slower here than TR-ALS-Sampled. All methods achieve good classification accuracy and similar decomposition errors.

Table 5. Run time, decomposition error and classification accuracy when using tensor decomposition for feature extraction.

Method	Time (s)	Err.	Acc. (%)
CP-ALS (Ten. Toolbox)	43.6	0.31	99.2
CPD-ALS (Tensorlab)	68.4	0.31	99.0
CPD-MINF (Tensorlab)	102.3	0.34	99.7
CPD-NLS (Tensorlab)	107.8	0.31	92.1
CP-ARLS-LEV	28.7	0.32	98.5
CP-ALS-ES (our)	27.9	0.32	98.3
TR-ALS	9813.7	0.31	99.3
TR-ALS-Sampled	9.9	0.33	98.5
TR-ALS-ES (our)	28.5	0.33	98.0

Remark 14. If k-NN was applied directly to the uncompressed images its cost would scale linearly or worse with the number of tensor entries. Due to the sublinear per-iteration complexity of our proposed methods, the cost of the entire decomposition is sublinear if the number of iterations is chosen appropriately. While fixing the number of iterations is not guaranteed to produce a good decomposition, we expect this to work well on typical datasets. Once the decomposition is computed each image has a representation of much lower dimension which makes applying k-NN cheaper. This leads to a reduction in the overall classification cost. See Section D.4 for a discussion on handling new images that were not part of the initial decomposition.

5.3. Demonstration of Improved Complexity

We construct a synthetic 10-way tensor that demonstrates the improved sampling and computational complexity of our proposed CP-ALS-ES over CP-ARLS-LEV. It is constructed via (3) from factor matrices $\boldsymbol{A}^{(n)} \in \mathbb{R}^{6\times 4}$ for $n \in [10]$ with $\boldsymbol{A}^{(n)}(1,1) = 4$, each $\boldsymbol{A}^{(n)}(i,j)$ drawn i.i.d. from a Gaussian distribution for $2 \leq i,j \leq 6$, and all other entries zero. Additionally, i.i.d. Gaussian noise with standard deviation 0.01 is added to all entries of the tensor. Both methods are run for 20 iterations with a target rank of 4 and are initialized using the randomized range finding approach proposed by Larsen & Kolda (2020), Appendix F. CP-ARLS-LEV requires $J = 6^8 \approx 1.7\text{e+}6$ samples to get an accurate solution, taking 350 seconds. By contrast, our CP-ALS-ES only requires a recursive sketch size of $J_1 = 1000$ and $J_2 = 50$ samples to get an accurate solution, taking only 4.8 seconds. Our method improves the sampling complexity and compute time by 4 and almost 2 orders of magnitude, respectively. A similar example for the tensor ring decomposition is provided in Section D.5.

6. Discussion and Conclusion

We have shown that it is possible to construct ALS algorithms with guarantees for both the CP and tensor ring decompositions of an N-way tensor with a per-iteration cost which does not depend exponentially on N. In the regime of high-dimensional tensors (i.e., with many modes), this is a substantial improvement over the previous state-of-the-art which had a per-iteration cost of $\Omega(R^{N+1})$ and $\Omega(R^{2N+2})$ for the CP and tensor ring decompositions, respectively, where R is the relevant notion of rank.

We again want to emphasize that this paper considers *worst-case* guarantees. As we saw in Section 5, real datasets typically behave better than the worst case. For such datasets, CP-ARLS-LEV and TR-ALS-Sampled usually yield good results even with fewer (i.e., not exponential in N) number of samples than what worst-case analysis might suggest. In such cases, those methods can be faster than the methods we propose in this paper. Nonetheless, we believe that our methods are still useful for those cases when worst-case performance is critical.

We also want to point out that, unlike their deterministic counterparts, our methods cannot guarantee a monotonically decreasing objective value. The other randomized methods we compare with have the same deficiency.

The sampling formulas (28) and (38) are sums of products where each feature matrix or core tensor (except the nth) appears twice. This can exacerbate issues with ill-conditioned factor matrices or core tensors. A particular concern is that catastrophic cancellation can occur, which will prevent accurate computation of the probabilities. Addressing this issue is an interesting direction for future research.

All the experiments in this paper involve dense tensors. However, our methods can also be applied to sparse tensors with only minor adjustments to the code. Thorough empirical evaluation of our methods on sparse tensors is therefore another interesting direction for future research.

Acknowledgements

We thank the anonymous reviewers for their feedback which helped improve the paper. This work was supported by the Laboratory Directed Research and Development Program of Lawrence Berkeley National Laboratory under U.S. Department of Energy Contract No. DE-AC02-05CH11231.

References

Aggour, K. S., Gittens, A., and Yener, B. Adaptive sketching for fast and convergent canonical polyadic decomposition. In *International Conference on Machine Learning*. PMLR, 2020.

Ahle, T. D., Kapralov, M., Knudsen, J. B., Pagh, R., Velingker, A., Woodruff, D. P., and Zandieh, A. Oblivious sketching of high-degree polynomial kernels. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 141–160. SIAM, 2020.

Ahmadi-Asl, S., Cichocki, A., Phan, A. H., Asante-Mensah, M. G., Mousavi, F., Oseledets, I., and Tanaka, T. Randomized algorithms for fast computation of low-rank tensor ring model. *Machine Learning: Science and Technology*, 2020.

Avron, H., Nguyen, H. L., and Woodruff, D. P. Subspace embeddings for the polynomial kernel. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, pp. 2258–2266, Cambridge, MA, USA, 2014. MIT Press.

Bader, B. W. and Kolda, T. G. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. *ACM Transactions on Mathematical Software (TOMS)*, 32(4):635–653, 2006.

Bader, B. W., Kolda, T. G., and others. MATLAB Tensor Toolbox, Version 3.2.1. Available online at https://www.tensortoolbox.org, 2021.

Bamberger, S., Krahmer, F., and Ward, R. Johnson-Lindenstrauss embeddings with Kronecker structure. *arXiv preprint arXiv:2106.13349*, 2021.

Battaglino, C., Ballard, G., and Kolda, T. G. A practical randomized CP tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 39(2):876–901, 2018.

Bengua, J. A., Phien, H. N., and Tuan, H. D. Optimal feature extraction and classification of tensors via matrix product state decomposition. In *2015 IEEE International Congress on Big Data*, pp. 669–672. IEEE, 2015.

Biagioni, D. J., Beylkin, D., and Beylkin, G. Randomized interpolative decomposition of separated representations. *Journal of Computational Physics*, 281(C):116–134, January 2015. ISSN 0021-9991. doi: 10.1016/j.jcp.2014.10.009.

Caiafa, C. F. and Cichocki, A. Generalizing the column–row matrix decomposition to multi-way arrays. *Linear Algebra and its Applications*, 433(3):557–573, 2010.

Charikar, M., Chen, K., and Farach-Colton, M. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, January 2004. ISSN 0304-3975. doi: 10.1016/S0304-3975(03)00400-6.

Che, M. and Wei, Y. Randomized algorithms for the approximations of Tucker and the tensor train decompositions. *Advances in Computational Mathematics*, 45(1):395–428, 2019.

Cheng, D., Peng, R., Liu, Y., and Perros, I. SPALS: Fast alternating least squares via implicit leverage scores sampling. In *Advances In Neural Information Processing Systems*, pp. 721–729, 2016.

Cichocki, A., Lee, N., Oseledets, I., Phan, A.-H., Zhao, Q., and Mandic, D. P. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429, 2016.

Cichocki, A., Phan, A.-H., Zhao, Q., Lee, N., Oseledets, I., Sugiyama, M., and Mandic, D. P. Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives. *Foundations and Trends® in Machine Learning*, 9(6):431–673, 2017.

Clarkson, K. L. and Woodruff, D. P. Low-rank approximation and regression in input sparsity time. *Journal of the ACM*, 63(6):54:1–54:45, February 2017. ISSN 0004-5411. doi: 10.1145/3019134.

Cohen, N., Sharir, O., and Shashua, A. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pp. 698–728, 2016.

da Costa, M. N., Lopes, R. R., and Romano, J. M. T. Randomized methods for higher-order subspace separation. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pp. 215–219. IEEE, 2016.

Dereziński, M. and Warmuth, M. K. Reverse iterative volume sampling for linear regression. *The Journal of Machine Learning Research*, 19(1):853–891, 2018.

Diao, H., Song, Z., Sun, W., and Woodruff, D. Sketching for Kronecker product regression and P-splines. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, pp. 1299–1308, 2018.

Diao, H., Jayaram, R., Song, Z., Sun, W., and Woodruff, D. P. Optimal sketching for Kronecker product regression and low rank approximation. *arXiv preprint arXiv:1909.13384*, 2019.

Drineas, P. and Mahoney, M. W. A randomized algorithm for a tensor-based generalization of the singular value decomposition. *Linear algebra and its applications*, 420 (2-3):553–571, 2007.

Drineas, P., Kannan, R., and Mahoney, M. W. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006a.

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, pp. 1127–1136, 2006b.

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008.

Drineas, P., Mahoney, M. W., Muthukrishnan, S., and Sarlós, T. Faster least squares approximation. *Numerische Mathematik*, 117(2):219–249, 2011.

Drineas, P., Magdon-Ismail, M., Mahoney, M. W., and Woodruff, D. P. Fast approximation of matrix coherence and statistical leverage. *The Journal of Machine Learning Research*, 13(1):3475–3506, 2012.

Espig, M., Naraparaju, K. K., and Schneider, J. A note on tensor chain approximation. *Computing and Visualization in Science*, 15(6):331–344, 2012.

Fahrbach, M., Ghadiri, M., and Fu, T. Fast low-rank tensor decomposition by ridge leverage score sampling. *arXiv preprint arXiv:2107.10654*, 2021.

Friedland, S., Mehrmann, V., Miedlar, A., and Nkengla, M. Fast low rank approximations of matrices and tensors. *Electronic Journal of Linear Algebra*, 22:1031–1048, 2011. ISSN 1081-3810. doi: 10.13001/1081-3810.1489.

Garipov, T., Podoprikhin, D., Novikov, A., and Vetrov, D. Ultimate tensorization: Compressing convolutional and FC layers alike. *arXiv preprint arXiv:1611.03214*, 2016.

Golub, G. H. and Van Loan, C. F. *Matrix Computations*. Johns Hopkins University Press, Baltimore, fourth edition, 2013. ISBN 978-1-4214-0794-4.

Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, May 2011.

Hazan, T., Polak, S., and Shashua, A. Sparse image coding using a 3D non-negative tensor factorization. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pp. 50–57. IEEE, 2005.

Hillar, C. J. and Lim, L.-H. Most tensor problems are NP-hard. *Journal of the ACM (JACM)*, 60(6):45, 2013.

Horn, R. A. and Johnson, C. R. *Topics in Matrix Analysis*. Cambridge University Press, 1994.

Hou, M., Tang, J., Zhang, J., Kong, W., and Zhao, Q. Deep multimodal multilinear fusion with high-order polynomial pooling. In *Advances in Neural Information Processing Systems*, pp. 12136–12145, 2019.

Howell, R. R. On asymptotic notation with multiple variables. Technical Report 2007-4, Dept. of Computing and Information Sciences, Kansas State University, 2008.

Huber, B., Schneider, R., and Wolf, S. A randomized tensor train singular value decomposition. In *Compressed Sensing and Its Applications*, pp. 261–290. Springer, 2017.

Iwen, M. A., Needell, D., Rebrova, E., and Zare, A. Lower memory oblivious (tensor) subspace embeddings with fewer random bits: Modewise methods for least squares. *SIAM Journal on Matrix Analysis and Applications*, 42 (1):376–416, 2021. doi: 10.1137/19M1308116.

Ji, Y., Wang, Q., Li, X., and Liu, J. A survey on tensor techniques and applications in machine learning. *IEEE Access*, 7:162950–162990, 2019.

Jin, R., Kolda, T. G., and Ward, R. Faster Johnson-Lindenstrauss transforms via Kronecker products. *Information and Inference: A Journal of the IMA*, October 2020. ISSN 2049-8772. doi: 10.1093/imaiai/iaaa028.

Khoo, Y., Lu, J., and Ying, L. Efficient construction of tensor ring representations from sampling. *arXiv preprint arXiv:1711.00954*, 2019.

Khrulkov, V., Novikov, A., and Oseledets, I. Expressive power of recurrent neural networks. In *International Conference on Learning Representations*, 2018.

Kolda, T. G. and Bader, B. W. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, August 2009. ISSN 0036-1445. doi: 10.1137/07070111X.

Kolda, T. G. and Hong, D. Stochastic gradients for large-scale tensor decomposition. *SIAM Journal on Mathematics of Data Science*, 2(4):1066–1095, January 2020. doi: 10.1137/19M1266265.

Larsen, B. W. and Kolda, T. G. Practical leverage-based sampling for low-rank tensor decomposition. *arXiv preprint arXiv:2006.16438v3*, 2020.

Lei, T., Xin, Y., Zhang, Y., Barzilay, R., and Jaakkola, T. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1381–1391, 2014.

Liu, D., Ran, S.-J., Wittek, P., Peng, C., García, R. B., Su, G., and Lewenstein, M. Machine learning by unitary tensor network of hierarchical tree structure. *New Journal of Physics*, 21(7):073059, 2019.

Mahoney, M. W. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2): 123–224, 2011.

Mahoney, M. W., Maggioni, M., and Drineas, P. Tensor-CUR decompositions for tensor-based data. *SIAM Journal on Matrix Analysis and Applications*, 30(3):957–987, 2008.

Malik, O. A. and Becker, S. Low-rank Tucker decomposition of large tensors using TensorSketch. In *Advances in Neural Information Processing Systems*, pp. 10096–10106, 2018.

Malik, O. A. and Becker, S. Fast randomized matrix and tensor interpolative decomposition using CountSketch. *Advances in Computational Mathematics*, 46(6): 76, October 2020a. ISSN 1572-9044. doi: 10.1007/s10444-020-09816-9.

Malik, O. A. and Becker, S. Guarantees for the Kronecker fast Johnson–Lindenstrauss transform using a coherence and sampling argument. *Linear Algebra and its Applications*, 602:120–137, October 2020b. ISSN 0024-3795. doi: 10.1016/j.laa.2020.05.004.

Malik, O. A. and Becker, S. A sampling-based method for tensor ring decomposition. In *International Conference on Machine Learning*, pp. 7400–7411. PMLR, 2021.

Martinsson, P.-G. and Tropp, J. A. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020. doi: 10.1017/S0962492920000021.

Minster, R., Saibaba, A. K., and Kilmer, M. E. Randomized algorithms for low-rank tensor decompositions in the Tucker format. *SIAM Journal on Mathematics of Data Science*, 2(1):189–215, 2020.

Nene, S. A., Nayar, S. K., and Murase, H. Columbia Object Image Library (COIL-100). Technical Report CUCS-006-96, Columbia University, 1996.

Novikov, A., Podoprikhin, D., Osokin, A., and Vetrov, D. P. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, pp. 442–450, 2015.

Novikov, A., Trofimov, M., and Oseledets, I. Exponential machines. *arXiv preprint arXiv:1605.03795*, 2016.

Oseledets, I. and Tyrtyshnikov, E. TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

Oseledets, I. V. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130–2145, 2010.

Oseledets, I. V. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Oseledets, I. V., Savostianov, D. V., and Tyrtyshnikov, E. E. Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM Journal on Matrix Analysis and Applications*, 30(3):939–956, 2008.

Pagh, R. Compressed matrix multiplication. *ACM Transactions on Computation Theory*, 5(3):9:1–9:17, August 2013. ISSN 1942-3454. doi: 10.1145/2493252.2493254.

Papalexakis, E. E., Faloutsos, C., and Sidiropoulos, N. D. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):1–44, 2016.

Pham, N. and Pagh, R. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pp. 239–247, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2174-7. doi: 10.1145/2487575.2487591.

Rakhshan, B. and Rabusseau, G. Tensorized random projections. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108, pp. 3306–3316. PMLR, August 2020.

Rakhshan, B. T. and Rabusseau, G. Rademacher random projections with tensor networks. In *NeurIPS Workshop on Quantum Tensor Networks in Machine Learning*, 2021.

Rigamonti, R., Sironi, A., Lepetit, V., and Fua, P. Learning separable filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2754–2761, 2013.

Sorber, L., Barel, M. V., and Lathauwer, L. D. Unconstrained optimization of real functions in complex variables. *SIAM Journal on Optimization*, 22(3):879–898, January 2012. ISSN 1052-6234. doi: 10.1137/110832124.

Sorber, L., Van Barel, M., and De Lathauwer, L. Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms, and a new generalization. *SIAM Journal on Optimization*, 23(2):695–720, January 2013. ISSN 1052-6234. doi: 10.1137/120868323.

Stoudenmire, E. M. and Schwab, D. J. Supervised learning with quantum-inspired tensor networks. *arXiv preprint arXiv:1605.05775*, May 2017.

Sun, Y., Guo, Y., Tropp, J. A., and Udell, M. Tensor random projection for low memory dimension reduction. In *NeurIPS Workshop on Relational Representation Learning*, 2018.

Sun, Y., Guo, Y., Luo, C., Tropp, J., and Udell, M. Low-rank Tucker approximation of a tensor from streaming data. *SIAM Journal on Mathematics of Data Science*, 2 (4):1123–1150, 2020.

Tarzanagh, D. A. and Michailidis, G. Fast randomized algorithms for t-product based tensor operations and decompositions with applications to imaging data. *SIAM Journal on Imaging Sciences*, 11(4):2629–2664, 2018.

Tsourakakis, C. E. MACH: Fast randomized tensor decompositions. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pp. 689–700. SIAM, 2010.

Vasilescu, M. A. O. and Terzopoulos, D. Multilinear analysis of image ensembles: Tensorfaces. In *European Conference on Computer Vision*, pp. 447–460. Springer, 2002.

Vervliet, N., Debals, O., Sorber, L., Van Barel, M., and De Lathauwer, L. Tensorlab 3.0. Available online at https://www.tensorlab.net, March 2016.

Wang, W., Aggarwal, V., and Aeron, S. Efficient low rank tensor ring completion. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5697–5705, 2017.

Wang, Y., Tung, H.-Y., Smola, A. J., and Anandkumar, A. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems*, pp. 991–999, 2015.

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

Yang, B., Zamzam, A., and Sidiropoulos, N. D. ParaSketch: Parallel tensor factorization via sketching. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pp. 396–404. SIAM, 2018.

Yang, Y., Krompass, D., and Tresp, V. Tensor-train recurrent neural networks for video classification. *arXiv preprint arXiv:1707.01786*, 2017.

Ye, J., Wang, L., Li, G., Chen, D., Zhe, S., Chu, X., and Xu, Z. Learning compact recurrent neural networks with block-term tensor decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9378–9387, 2018.

Yu, R., Zheng, S., Anandkumar, A., and Yue, Y. Long-term forecasting using higher order tensor RNNs. *arXiv preprint arXiv:1711.00073*, 2017.

Yuan, L., Li, C., Cao, J., and Zhao, Q. Randomized tensor ring decomposition and its application to large-scale data reconstruction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2127–2131. IEEE, 2019a.

Yuan, L., Zhao, Q., Gui, L., and Cao, J. High-order tensor completion via gradient-based optimization under tensor train format. *Signal Processing: Image Communication*, 73:53–61, 2019b.

Zhang, J., Saibaba, A. K., Kilmer, M. E., and Aeron, S. A randomized tensor singular value decomposition based on the t-product. *Numerical Linear Algebra with Applications*, 25:e2179, 2018.

Zhao, Q., Zhou, G., Xie, S., Zhang, L., and Cichocki, A. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.

A. Further Details on Notation

In this section we provide some further details on the notation used in this paper to help make the paper self contained. The *Kronecker product* of two matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{k\times \ell}$ is denoted by $\boldsymbol{A} \otimes \boldsymbol{B} \in \mathbb{R}^{mk\times n\ell}$ and is defined as

$$\boldsymbol{A} \otimes \boldsymbol{B} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{A}(1,1)\boldsymbol{B} & \boldsymbol{A}(1,2)\boldsymbol{B} & \cdots & \boldsymbol{A}(1,n)\boldsymbol{B} \\ \boldsymbol{A}(2,1)\boldsymbol{B} & \boldsymbol{A}(2,2)\boldsymbol{B} & \cdots & \boldsymbol{A}(2,n)\boldsymbol{B} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{A}(m,1)\boldsymbol{B} & \boldsymbol{A}(m,2)\boldsymbol{B} & \cdots & \boldsymbol{A}(m,n)\boldsymbol{B} \end{bmatrix}. \tag{39}$$

The *Khatri–Rao product*, sometimes called the columnwise Kronecker product, of two matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{k\times n}$ is denoted by $\boldsymbol{A} \odot \boldsymbol{B} \in \mathbb{R}^{mk\times n}$ and is defined as

$$\boldsymbol{A} \odot \boldsymbol{B} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{A}(:,1) \otimes \boldsymbol{B}(:,1) & \boldsymbol{A}(:,2) \otimes \boldsymbol{B}(:,2) & \cdots & \boldsymbol{A}(:,n) \otimes \boldsymbol{B}(:,n) \end{bmatrix}. \tag{40}$$

It is not obvious how to extend the standard asymptotic notation for single-variable functions to multi-variable functions (Howell, 2008). Suppose f and g are positive functions over some parameters $x_1, \ldots, x_n$. We say that a function $f(x_1, \ldots, x_n)$ is $O(g(x_1, \ldots, x_n))$ if there exists a constant $C > 0$ such that $f(x_1, \ldots, x_n) \leq Cg(x_1, \ldots, x_n)$ for all valid values of the parameters $x_1, \ldots, x_n$. The notation $\tilde{O}$ means the same as O but with polylogarithmic factors ignored. We say that a function $f(x_1, \ldots, x_n)$ is $\Omega(g(x_1, \ldots, x_n))$, or alternatively write $f(x_1, \ldots, x_n) \gtrsim g(x_1, \ldots, x_n)$, if there exists a constant $C > 0$ such that $f(x_1, \ldots, x_n) \geq Cg(x_1, \ldots, x_n)$ for all valid values of the parameters $x_1, \ldots, x_n$.

B. Missing Proofs

B.1. Proof of Theorem 8

We first state and prove Lemma 15. It is similar to Lemma 5 in Drineas et al. (2012) and Lemma 4.1 in Drineas et al. (2006b) which consider the case when $\boldsymbol{\Psi}$ is a fast Johnson–Lindenstrauss transform and a sampling matrix, respectively, instead of a subspace embedding.

Lemma 15. *Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{I\times R}$ where $I > R$. Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ be a compact SVD of $\boldsymbol{A}$. Suppose $\boldsymbol{\Psi}$ is a γ-subspace embedding for $\boldsymbol{A}$ with $\gamma \in (0,1)$, and let $\boldsymbol{\Psi}\boldsymbol{U} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{W}^\top$ be a compact SVD. Then, the following hold:*

(i) $\operatorname{rank}(\boldsymbol{\Psi}\boldsymbol{A}) = \operatorname{rank}(\boldsymbol{\Psi}\boldsymbol{U}) = \operatorname{rank}(\boldsymbol{A})$,

(ii) $\|\boldsymbol{I} - \boldsymbol{\Lambda}^{-2}\|_2 \leq \gamma/(1-\gamma)$,

(iii) $(\boldsymbol{\Psi}\boldsymbol{A})^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\Psi}\boldsymbol{U})^\dagger$.

Proof. The proof follows similar arguments as those used in the proof of Lemma 4.1 in Drineas et al. (2006b). Since $\boldsymbol{\Psi}$ is a γ-subspace embedding for $\boldsymbol{A}$, we have that

$$(1-\gamma)\|\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{x}\|_2^2 \leq \|\boldsymbol{\Psi}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{x}\|_2^2 \leq (1+\gamma)\|\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{x}\|_2^2 \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^R. \tag{41}$$

Let $r \stackrel{\text{def}}{=} \operatorname{rank}(\boldsymbol{A})$. Since $\boldsymbol{\Sigma}\boldsymbol{V}^\top \in \mathbb{R}^{r\times R}$ is full rank, and using unitary invariance of the spectral norm, it follows that

$$(1-\gamma)\|\boldsymbol{y}\|_2^2 \leq \|\boldsymbol{\Psi}\boldsymbol{U}\boldsymbol{y}\|_2^2 \leq (1+\gamma)\|\boldsymbol{y}\|_2^2 \quad \text{for all } \boldsymbol{y} \in \mathbb{R}^r. \tag{42}$$

Using Theorem 8.6.1 in Golub & Van Loan (2013), this in turn implies that

$$1-\gamma \leq \sigma_i^2(\boldsymbol{\Psi}\boldsymbol{U}) \leq 1+\gamma \quad \text{for all } i \in [r]. \tag{43}$$

Consequently, $\operatorname{rank}(\boldsymbol{\Psi}\boldsymbol{U}) = r = \operatorname{rank}(\boldsymbol{A})$. Moreover, since $\operatorname{rank}(\boldsymbol{\Psi}\boldsymbol{A}) = \operatorname{rank}(\boldsymbol{\Psi}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top)$ and $\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is full rank, it follows that $\operatorname{rank}(\boldsymbol{\Psi}\boldsymbol{A}) = \operatorname{rank}(\boldsymbol{\Psi}\boldsymbol{U})$. This completes the proof of (i).

Next, note that

$$\|\boldsymbol{I} - \boldsymbol{\Lambda}^{-2}\|_2 = \max_{i\in[r]} \left|1 - \frac{1}{\sigma_i^2(\boldsymbol{\Psi}\boldsymbol{U})}\right| = \max_{i\in[r]} \left|\frac{\sigma_i^2(\boldsymbol{\Psi}\boldsymbol{U}) - 1}{\sigma_i^2(\boldsymbol{\Psi}\boldsymbol{U})}\right| \leq \frac{\gamma}{1-\gamma}, \tag{44}$$

where the inequality follows from the bound in (43). This completes the proof of (ii).

We may write

$$(\boldsymbol{\Psi A})^\dagger = (\boldsymbol{Q\Lambda W}^\top \boldsymbol{\Sigma V}^\top)^\dagger = \boldsymbol{V}(\boldsymbol{\Lambda W}^\top \boldsymbol{\Sigma})^\dagger \boldsymbol{Q}^\top \tag{45}$$

where the second equality follows since $\boldsymbol{Q}$ and $\boldsymbol{V}$ have orthonormal columns. Since $\operatorname{rank}(\boldsymbol{\Psi U}) = \operatorname{rank}(\boldsymbol{A})$ due to (i), the matrix $\boldsymbol{\Lambda W}^\top \in \mathbb{R}^{r\times r}$ is invertible, and therefore $\boldsymbol{\Lambda W}^\top \boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}$ is invertible, and hence

$$(\boldsymbol{\Lambda W}^\top \boldsymbol{\Sigma})^\dagger = \boldsymbol{\Sigma}^{-1}\boldsymbol{W}\boldsymbol{\Lambda}^{-1}. \tag{46}$$

Consequently, combining (45) and (46) we have

$$(\boldsymbol{\Psi A})^\dagger = \boldsymbol{V\Sigma}^{-1}\boldsymbol{W\Lambda}^{-1}\boldsymbol{Q}^\top = \boldsymbol{V\Sigma}^{-1}(\boldsymbol{\Psi U})^\dagger. \tag{47}$$

This completes the proof of (iii). □

We are now ready to prove the statement in Theorem 8.

Proof of Theorem 8. Our proof is similar to the proof of Lemma 9 in Drineas et al. (2012). Let $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\top$ be a compact SVD, $r \overset{\text{def}}{=} \operatorname{rank}(\boldsymbol{A})$ and suppose $i \in [I]$. From Definition 6, we have

$$\ell_i(\boldsymbol{A}) = \|\boldsymbol{U}(i,:)\|_2^2 = \boldsymbol{e}_i^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{e}_i. \tag{48}$$

Moreover,

$$\tilde{\ell}_i(\boldsymbol{A}) = \|\boldsymbol{e}_i^\top \boldsymbol{A}\boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{U}_1^\top\|_2^2 = \|\boldsymbol{e}_i^\top \boldsymbol{A}(\boldsymbol{\Psi A})^\dagger\|_2^2 = \|\boldsymbol{e}_i^\top \boldsymbol{U}(\boldsymbol{\Psi U})^\dagger\|_2^2 = \boldsymbol{e}_i^\top \boldsymbol{U}(\boldsymbol{\Psi U})^\dagger(\boldsymbol{\Psi U})^{\dagger\top}\boldsymbol{U}^\top \boldsymbol{e}_i, \tag{49}$$

where the first equality follows from the definition of $\tilde{\ell}_i(\boldsymbol{A})$ in (20) and the unitary invariance of the spectral norm, and the third equality follows from Lemma 15 (iii). From (48) and (49), we have

$$\begin{aligned}
|\ell_i(\boldsymbol{A}) - \tilde{\ell}_i(\boldsymbol{A})| &= |\boldsymbol{e}_i^\top \boldsymbol{U}\big(\boldsymbol{I} - (\boldsymbol{\Psi U})^\dagger(\boldsymbol{\Psi U})^{\dagger\top}\big)\boldsymbol{U}^\top \boldsymbol{e}_i| \\
&\le \|\boldsymbol{e}_i^\top \boldsymbol{U}\|_2 \cdot \|(\boldsymbol{I} - (\boldsymbol{\Psi U})^\dagger(\boldsymbol{\Psi U})^{\dagger\top})\boldsymbol{U}^\top \boldsymbol{e}_i\|_2 \\
&\le \|\boldsymbol{e}_i^\top \boldsymbol{U}\|_2 \cdot \|\boldsymbol{I} - (\boldsymbol{\Psi U})^\dagger(\boldsymbol{\Psi U})^{\dagger\top}\|_2 \cdot \|\boldsymbol{U}^\top \boldsymbol{e}_i\|_2 \\
&= \|\boldsymbol{I} - (\boldsymbol{\Psi U})^\dagger(\boldsymbol{\Psi U})^{\dagger\top}\|_2 \cdot \ell_i(\boldsymbol{A}),
\end{aligned} \tag{50}$$

where the first inequality follows from Cauchy–Schwarz inequality, and the second inequality follows from the definition of the matrix spectral norm. Let $\boldsymbol{\Psi U} = \boldsymbol{Q\Lambda W}^\top$ be a compact SVD. It follows that

$$\|\boldsymbol{I} - (\boldsymbol{\Psi U})^\dagger(\boldsymbol{\Psi U})^{\dagger\top}\|_2 = \|\boldsymbol{I} - \boldsymbol{W\Lambda}^{-2}\boldsymbol{W}^\top\|_2. \tag{51}$$

From Lemma 15 (i), it follows that $\boldsymbol{W}$ is $r \times r$, hence $\boldsymbol{WW}^\top = \boldsymbol{I}$. Consequently, and using unitary invariance of the spectral norm,

$$\|\boldsymbol{I} - \boldsymbol{W\Lambda}^{-2}\boldsymbol{W}^\top\|_2 = \|\boldsymbol{I} - \boldsymbol{\Lambda}^{-2}\|_2. \tag{52}$$

Combining (50), (51) and (52), we get

$$\big|\ell_i(\boldsymbol{A}) - \tilde{\ell}_i(\boldsymbol{A})\big| \le \|\boldsymbol{I} - \boldsymbol{\Lambda}^{-2}\|_2 \cdot \ell_i(\boldsymbol{A}) \le \frac{\gamma}{1-\gamma}\ell_i(\boldsymbol{A}), \tag{53}$$

where the last inequality follows from Lemma 15 (ii). This completes the proof. □

B.2. Proof of Theorem 9

We first state some results that we will need for this proof. Lemma 16 follows from Lemma 4.2.10 in Horn & Johnson (1994).

Lemma 16. *For matrices $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$ and $\boldsymbol{N}_1, \ldots, \boldsymbol{N}_n$ of appropriate sizes,*

$$(\boldsymbol{M}_1 \otimes \cdots \otimes \boldsymbol{M}_n) \cdot (\boldsymbol{N}_1 \otimes \cdots \otimes \boldsymbol{N}_n) = (\boldsymbol{M}_1\boldsymbol{N}_1) \otimes \cdots \otimes (\boldsymbol{M}_n\boldsymbol{N}_n). \tag{54}$$

Theorem 17 follows directly from Theorem 1 in Ahle et al. (2020) and its proof.[4]

Theorem 17. *Let $\boldsymbol{A} \in \mathbb{R}^{I^N \times R}$. Let $\boldsymbol{\Psi} \in \mathbb{R}^{J \times I^N}$ be the $(J, (I)_{j=1}^N)$-recursive sketch described in Section 3.2. If $J \gtrsim NR^2/(\gamma^2\delta)$, then $\boldsymbol{\Psi}$ is a γ-subspace embedding for $\boldsymbol{A}$ with probability at least $1-\delta$.*

It is easy to generalize Theorem 17 to the setting when $\boldsymbol{\Psi}$ is a $(J, (I_j)_{j=1}^N)$-recursive sketch where the I_j are not necessarily all equal.

Corollary 18. *Let $\boldsymbol{A} \in \mathbb{R}^{\Pi_{j=1}^N I_j \times R}$. Let $\boldsymbol{\Psi} \in \mathbb{R}^{J \times \Pi_{j=1}^N I_j}$ be the $(J, (I_j)_{j=1}^N)$-recursive sketch described in Section 3.2. If $J \gtrsim NR^2/(\gamma^2\delta)$, then $\boldsymbol{\Psi}$ is a γ-subspace embedding for $\boldsymbol{A}$ with probability at least $1-\delta$.*

Proof. Let $q \overset{\text{def}}{=} \lceil \log_2(N) \rceil$, $I_{\max} \overset{\text{def}}{=} \max_{j \in [N]} I_j$, and $\tilde{I}_j \overset{\text{def}}{=} I_j$ for $j \leq N$ and $\tilde{I}_j \overset{\text{def}}{=} I_{\max}$ for $j > N$. Let $\mathbf{1}_{1\times R}$ denote a length-R row vector of all ones. From the definition of the recursive sketch in Section 3.2 and the factorization in (17), we have

$$\begin{aligned}
\boldsymbol{\Psi}\boldsymbol{A} &= \boldsymbol{\Psi}_{J,(\tilde{I}_j)_{j=1}^{2^q}}\left(\boldsymbol{A} \odot (\boldsymbol{e}_1^{\otimes(2^q-N)}\mathbf{1}_{1\times R})\right) \\
&= \boldsymbol{T}^{(q)}\boldsymbol{T}^{(q-1)}\cdots\boldsymbol{T}^{(1)}\Big(\bigotimes_{j=1}^{2^q}\boldsymbol{C}_j\Big)\left(\boldsymbol{A} \odot (\boldsymbol{e}_1^{\otimes(2^q-N)}\mathbf{1}_{1\times R})\right) \\
&= \boldsymbol{T}^{(q)}\boldsymbol{T}^{(q-1)}\cdots\boldsymbol{T}^{(1)}\Bigg(\Big(\Big(\bigotimes_{j=1}^{N}\boldsymbol{C}_j\Big)\boldsymbol{A}\Big) \odot \Big(\Big(\bigotimes_{j=N+1}^{2^q}\boldsymbol{C}_j\boldsymbol{e}_1\Big)\mathbf{1}_{1\times R}\Big)\Bigg),
\end{aligned} \tag{55}$$

where the last equality follows from Lemma 16. Define two index sets

$$\mathcal{I} \overset{\text{def}}{=} [I_1] \times \cdots \times [I_N] \quad \text{and} \quad \mathcal{I}^c \overset{\text{def}}{=} [I_{\max}]^N \setminus \mathcal{I}. \tag{56}$$

Let $\hat{\boldsymbol{A}} \in \mathbb{R}^{I_{\max}^N \times R}$ be an augmented version of $\boldsymbol{A}$ defined as

$$\hat{\boldsymbol{A}}(\overline{i_N \cdots i_1}, :) = \begin{cases} \boldsymbol{A}(\overline{i_N \cdots i_1}, :) & \text{if } (i_1, \ldots, i_N) \in \mathcal{I}, \\ 0 & \text{if } (i_1, \ldots, i_N) \in \mathcal{I}^c. \end{cases} \tag{57}$$

Let $\hat{\boldsymbol{\Psi}} \in \mathbb{R}^{J \times I_{\max}^N}$ be the $(J, (I_{\max})_{j=1}^N)$-recursive sketch which uses independent CountSketches defined as

$$\hat{\boldsymbol{C}}_j \in \mathbb{R}^{J \times I_{\max}} \overset{\text{def}}{=} \begin{bmatrix} \boldsymbol{C}_j & \tilde{\boldsymbol{C}}_j \end{bmatrix}, \quad j \in [2^q], \tag{58}$$

where the matrices $\boldsymbol{C}_j$ are the same as in (55), and each $\tilde{\boldsymbol{C}}_j$ is an independent CountSketch of size $J \times (I_{\max} - I_j)$. Again using the factorization in (17), we have

$$\begin{aligned}
\hat{\boldsymbol{\Psi}}\hat{\boldsymbol{A}} &= \boldsymbol{\Psi}_{J,(I_{\max})_{j=1}^{2^q}}\left(\hat{\boldsymbol{A}} \odot \left(\boldsymbol{e}_1^{\otimes(2^q-N)}\mathbf{1}_{1\times R}\right)\right) \\
&= \boldsymbol{T}^{(q)}\boldsymbol{T}^{(q-1)}\cdots\boldsymbol{T}^{(1)}\Big(\bigotimes_{j=1}^{2^q}\hat{\boldsymbol{C}}_j\Big)\left(\hat{\boldsymbol{A}} \odot (\boldsymbol{e}_1^{\otimes(2^q-N)}\mathbf{1}_{1\times R})\right) \\
&= \boldsymbol{T}^{(q)}\boldsymbol{T}^{(q-1)}\cdots\boldsymbol{T}^{(1)}\Bigg(\Big(\Big(\bigotimes_{j=1}^{N}\hat{\boldsymbol{C}}_j\Big)\hat{\boldsymbol{A}}\Big) \odot \Big(\Big(\bigotimes_{j=N+1}^{2^q}\hat{\boldsymbol{C}}_j\boldsymbol{e}_1\Big)\mathbf{1}_{1\times R}\Big)\Bigg),
\end{aligned} \tag{59}$$

where the last equality follows from Lemma 16. From the definition of matrix multiplication, we have

$$\Big(\bigotimes_{j=1}^{N}\hat{\boldsymbol{C}}_j\Big)\hat{\boldsymbol{A}} = \sum_{(i_1,\ldots,i_N)\in\mathcal{I}\cup\mathcal{I}^c}\Big(\bigotimes_{j=1}^{N}\hat{\boldsymbol{C}}_j\Big)(:, \overline{i_N \cdots i_1})\hat{\boldsymbol{A}}(\overline{i_N \cdots i_1}, :). \tag{60}$$

[4] Since we are not considering regularized least squares problems, the statistical dimension s_λ in Ahle et al. (2020) just becomes equal to the number of columns of $\boldsymbol{A}$, which is R in our case. The statement of Theorem 1 in Ahle et al. (2020) uses $\delta = 1/10$, but the statement for general δ is easy to infer from their proof of the theorem.

Due to (58), it follows that $\hat{\boldsymbol{C}}_j(:, i_j) = \boldsymbol{C}_j(:, i_j)$ when $i_j \in [I_j]$, and consequently

$$\Big(\bigotimes_{j=1}^{N} \hat{\boldsymbol{C}}_j\Big)(:, \overline{i_N \cdots i_1}) = \bigotimes_{j=1}^{N} \hat{\boldsymbol{C}}_j(:, i_j) = \bigotimes_{j=1}^{N} \boldsymbol{C}_j(:, i_j) = \Big(\bigotimes_{j=1}^{N} \boldsymbol{C}_j\Big)(:, \overline{i_N \cdots i_1}) \quad \text{for all } (i_1, \ldots, i_N) \in \mathcal{I}. \tag{61}$$

Using (57) and (61), we can simplify (60) to

$$\Big(\bigotimes_{j=1}^{N} \hat{\boldsymbol{C}}_j\Big)\hat{\boldsymbol{A}} = \sum_{(i_1,\ldots,i_N)\in\mathcal{I}} \Big(\bigotimes_{j=1}^{N} \boldsymbol{C}_j\Big)(:, \overline{i_N \cdots i_1})\boldsymbol{A}(\overline{i_N \cdots i_1}, :) = \Big(\bigotimes_{j=1}^{N} \boldsymbol{C}_j\Big)\boldsymbol{A}. \tag{62}$$

Similarly, since the first column of each $\hat{\boldsymbol{C}}_j$ and $\boldsymbol{C}_j$ are the same,

$$\Big(\bigotimes_{j=N+1}^{2^q} \hat{\boldsymbol{C}}_j \boldsymbol{e}_1\Big)\mathbf{1}_{1\times R} = \Big(\bigotimes_{j=N+1}^{2^q} \boldsymbol{C}_j \boldsymbol{e}_1\Big)\mathbf{1}_{1\times R}. \tag{63}$$

Equations (55), (59), (62) and (63) together now imply that

$$\boldsymbol{\Psi}\boldsymbol{A} = \hat{\boldsymbol{\Psi}}\hat{\boldsymbol{A}}. \tag{64}$$

Moreover, it follows immediately from (57) that

$$\|\boldsymbol{A}\boldsymbol{x}\|_2 = \|\hat{\boldsymbol{A}}\boldsymbol{x}\|_2 \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^R. \tag{65}$$

Theorem 17 implies that

$$\mathbb{P}\big(\big|\|\hat{\boldsymbol{\Psi}}\hat{\boldsymbol{A}}\boldsymbol{x}\|_2^2 - \|\hat{\boldsymbol{A}}\boldsymbol{x}\|_2^2\big| \le \gamma\|\hat{\boldsymbol{A}}\boldsymbol{x}\|_2^2 \text{ for all } \boldsymbol{x} \in \mathbb{R}^R\big) \ge 1 - \delta. \tag{66}$$

Due to (64) and (65), this implies that

$$\mathbb{P}\big(\big|\|\boldsymbol{\Psi}\boldsymbol{A}\boldsymbol{x}\|_2^2 - \|\boldsymbol{A}\boldsymbol{x}\|_2^2\big| \le \gamma\|\boldsymbol{A}\boldsymbol{x}\|_2^2 \text{ for all } \boldsymbol{x} \in \mathbb{R}^R\big) \ge 1 - \delta, \tag{67}$$

which is what we wanted to show. □

Theorem 19 is a well-known result. Since slightly different variations of it have appeared in the literature (Drineas et al., 2006b; 2008; 2011; Larsen & Kolda, 2020) we provide a proof sketch just to give the reader some idea of how to derive the version we use.

Theorem 19. *Let $\boldsymbol{A} \in \mathbb{R}^{I\times R}$ be a matrix, and suppose $\boldsymbol{S} \sim \mathcal{D}(J, \boldsymbol{q})$ is a leverage score sampling matrix for $(\boldsymbol{A}, \beta)$ where $\beta \in (0, 1]$, and that $\varepsilon, \delta \in (0, 1)$. Moreover, define* $\text{OPT} \stackrel{\text{def}}{=} \min_{\boldsymbol{X}} \|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{Y}\|_{\text{F}}$ *and* $\tilde{\boldsymbol{X}} \stackrel{\text{def}}{=} \arg\min_{\boldsymbol{X}} \|\boldsymbol{S}\boldsymbol{A}\boldsymbol{X} - \boldsymbol{S}\boldsymbol{Y}\|_{\text{F}}$. *If*

$$J > \frac{4R}{\beta}\max\Big(\frac{4}{3(\sqrt{2}-1)^2}\ln\Big(\frac{4R}{\delta}\Big), \frac{1}{\varepsilon\delta}\Big), \tag{68}$$

then the following holds with probability at least $1 - \delta$:

$$\|\boldsymbol{A}\tilde{\boldsymbol{X}} - \boldsymbol{Y}\|_{\text{F}} \le (1 + \varepsilon)\text{OPT}. \tag{69}$$

Proof sketch. Let $\boldsymbol{U} \in \mathbb{R}^{I\times \text{rank}(\boldsymbol{A})}$ contain the left singular vectors of $\boldsymbol{A}$, and define $\boldsymbol{Y}^{\perp} \stackrel{\text{def}}{=} (\boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^{\top})\boldsymbol{Y}$. According to a matrix version[5] of Lemma 1 by Drineas et al. (2011), the statement in (69) holds if both

$$\sigma_{\min}^2(\boldsymbol{S}\boldsymbol{U}) \ge \frac{1}{\sqrt{2}} \tag{70}$$

and

$$\|\boldsymbol{U}^{\top}\boldsymbol{S}^{\top}\boldsymbol{S}\boldsymbol{Y}^{\perp}\|_{\text{F}}^2 \le \frac{\varepsilon}{2}\text{OPT}^2. \tag{71}$$

[5] See Lemma S1 in Malik & Becker (2021).

To complete the proof, it is therefore sufficient to show that $\boldsymbol{S}$ satisfies both (70) and (71) with probability at least $1-\delta$. Using Lemma S2 in Malik & Becker (2021), which is the same as Theorem 2.11 in Woodruff (2014) but with a slightly smaller constant, one can show that the condition (70) is satisfied with probability at least $1-\delta/2$ if

$$J > \frac{16}{3(\sqrt{2}-1)^2}\frac{R}{\beta}\ln\Big(\frac{4R}{\delta}\Big). \tag{72}$$

Next, using Lemma 8 in Drineas et al. (2006a), it follows that

$$\mathbb{E}\|\boldsymbol{U}^\top\boldsymbol{S}^\top\boldsymbol{S}\boldsymbol{Y}^\perp\|_{\mathrm{F}}^2 \leq \frac{1}{J\beta}R\cdot\mathrm{OPT}^2. \tag{73}$$

Markov's inequality together with the assumption

$$J > \frac{4R}{\beta\varepsilon\delta} \tag{74}$$

then implies that

$$\mathbb{P}\Big(\|\boldsymbol{U}^\top\boldsymbol{S}^\top\boldsymbol{S}\boldsymbol{Y}^\perp\|_{\mathrm{F}}^2 > \frac{\varepsilon}{2}\mathrm{OPT}^2\Big) \leq \frac{2R}{J\varepsilon\beta} < \frac{\delta}{2}. \tag{75}$$

If (68) is satisfied, then both (72) and (74) are satisfied, and consequently (70) and (71) are both true with probability at least $1-\delta$. □

We are now ready to prove the statement in Theorem 9.

Proof of Theorem 9. Let E_1 denote the event that $\boldsymbol{\Psi}$ is a $1/3$-subspace embedding for $\boldsymbol{A}^{\neq n}$. Following the notation used in Theorem 8, let $\boldsymbol{\Psi}\boldsymbol{A}^{\neq n} = \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^\top$ be a compact SVD. Let E_2 denote the event that (24) is true.

According to Corollary 18, we can guarantee that $\mathbb{P}(E_1)\geq 1-\delta/2$ if we choose J_1 as in (22). With $\gamma = 1/3$ in Theorem 8, the estimates $\tilde{\ell}_i(\boldsymbol{A}^{\neq n})$ satisfy

$$\frac{1}{2}\ell_i(\boldsymbol{A}^{\neq n}) \leq \tilde{\ell}_i(\boldsymbol{A}^{\neq n}) \leq \frac{3}{2}\ell_i(\boldsymbol{A}^{\neq n}). \tag{76}$$

Consequently,

$$\sum_{i=1}^{\Pi_{j\neq n}I_j}\tilde{\ell}_i(\boldsymbol{A}^{\neq n}) \leq \frac{3}{2}\sum_{i=1}^{\Pi_{j\neq n}I_j}\ell_i(\boldsymbol{A}^{\neq n}) = \frac{3}{2}\operatorname{rank}(\boldsymbol{A}^{\neq n}). \tag{77}$$

Therefore, since $\boldsymbol{q}(i)\propto\tilde{\ell}_i(\boldsymbol{A}^{\neq n})$, it follows by combining (76) and (77) that

$$\boldsymbol{q}(i) = \frac{\tilde{\ell}_i(\boldsymbol{A}^{\neq n})}{\sum_{i=1}^{\Pi_{j\neq n}I_j}\tilde{\ell}_i(\boldsymbol{A}^{\neq n})} \geq \frac{1}{3}\frac{\ell_i(\boldsymbol{A}^{\neq n})}{\operatorname{rank}(\boldsymbol{A}^{\neq n})}. \tag{78}$$

In view of Definition 7, Theorem 8 therefore implies that $\boldsymbol{S}\sim\mathcal{D}(J_2,\boldsymbol{q})$ is a leverage score sampling matrix for $(\boldsymbol{A}^{\neq n}, 1/3)$ if the event E_1 is true. From Theorem 19, it then follows that $\mathbb{P}(E_2\mid E_1)\geq 1-\delta/2$ if J_2 is chosen as in (23). With the choices of J_1 and J_2 above we now have

$$\mathbb{P}(E_2)\geq\mathbb{P}(E_1,E_2) = \mathbb{P}(E_1)\mathbb{P}(E_2\mid E_1)\geq(1-\delta/2)^2\geq 1-\delta \tag{79}$$

which is what we wanted to show. □

B.3. Proof of Lemma 10

Recall that $\boldsymbol{\Phi}\overset{\text{def}}{=}\boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1})^\top$, where $\boldsymbol{\Psi}\boldsymbol{A}^{\neq n} = \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^\top$ is a compact SVD. From (20) we have

$$\tilde{\ell}_i(\boldsymbol{A}^{\neq n}) = \boldsymbol{e}_i^\top\boldsymbol{A}^{\neq n}\boldsymbol{\Phi}\boldsymbol{A}^{\neq n\top}\boldsymbol{e}_i = (\boldsymbol{A}^{\neq n}\boldsymbol{\Phi}\boldsymbol{A}^{\neq n\top})(i,i) = \sum_{r,k}\boldsymbol{\Phi}(r,k)\cdot\prod_{j\neq n}\boldsymbol{A}^{(j)}(i_j,r)\boldsymbol{A}^{(j)}(i_j,k), \tag{80}$$

where the last equality follows from the definition of $\boldsymbol{A}^{\neq n}$ in (7), and $i = \overline{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N}$. Using (80), we can compute the normalization constant C as

$$C \stackrel{\text{def}}{=} \sum_i \tilde{\ell}_i(\boldsymbol{A}^{\neq n}) = \sum_{r,k} \boldsymbol{\Phi}(r,k) \cdot \prod_{j \neq n} \sum_{i_j} \boldsymbol{A}^{(j)}(i_j, r) \boldsymbol{A}^{(j)}(i_j, k) = \sum_{r,k} \boldsymbol{\Phi}(r,k) \cdot \prod_{j \neq n} (\boldsymbol{A}^{(j)\top} \boldsymbol{A}^{(j)})(r,k), \tag{81}$$

which proves (27).

To make notation a bit less cumbersome, we will use the abbreviated notation

$$\sum_{\{i_j\}_{j>m, j\neq n}} \quad \text{to denote} \quad \begin{cases} \sum_{i_{m+1}} \cdots \sum_{i_{n-1}} \sum_{i_{n+1}} \cdots \sum_{i_N} & \text{if } n > m, \\ \sum_{i_{m+1}} \cdots \sum_{i_N} & \text{otherwise.} \end{cases} \tag{82}$$

Similar abbreviated notation will also be used later on for other indices. We can again use (80) to compute the marginal probabilities of drawing $(i_j)_{j\leq m, j\neq n}$ as

$$\begin{aligned} \mathbb{P}((i_j)_{j\leq m, j\neq n}) &= \frac{1}{C} \sum_{\{i_j\}_{j>m, j\neq n}} \tilde{\ell}_i(\boldsymbol{A}^{\neq n}) \\ &= \frac{1}{C} \sum_{\{i_j\}_{j>m, j\neq n}} \Big(\sum_{r,k} \boldsymbol{\Phi}(r,k) \prod_{j\neq n} \boldsymbol{A}^{(j)}(i_j, r) \boldsymbol{A}^{(j)}(i_j, k) \Big) \\ &= \frac{1}{C} \sum_{r,k} \boldsymbol{\Phi}(r,k) \Big(\prod_{\substack{j\leq m \\ j\neq n}} \boldsymbol{A}^{(j)}(i_j, r) \boldsymbol{A}^{(j)}(i_j, k) \Big) \Big(\prod_{\substack{j>m \\ j\neq n}} (\boldsymbol{A}^{(j)\top} \boldsymbol{A}^{(j)})(r,k) \Big), \end{aligned} \tag{83}$$

which proves (28).

B.4. Proof of Theorem 11

The strategy of this proof is similar to that for the proof of Theorem 9 given in Section B.2. Let E_1 denote the event that $\boldsymbol{\Psi}$ is a $1/3$-subspace embedding for $\boldsymbol{G}_{[2]}^{\neq n}$. Following the notation used in Theorem 8, let $\boldsymbol{\Psi} \boldsymbol{G}_{[2]}^{\neq n} = \boldsymbol{U}_1 \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^\top$ be a compact SVD. Let E_2 denote the event that (32) is true.

The matrix $\boldsymbol{G}_{[2]}^{\neq n}$ is of size $\prod_{j\neq n} I_j \times R_{n-1} R_n$. According to Corollary 18, we can therefore guarantee that $\mathbb{P}(E_1) \geq 1 - \delta/2$ if we choose J_1 as in (30). Following the same line of reasoning as in the proof of Theorem 9, we can show that the choice $\gamma = 1/3$ in Theorem 8 combined with the fact $\boldsymbol{q}(i) \propto \tilde{\ell}_i(\boldsymbol{G}_{[2]}^{\neq n})$ implies that

$$\boldsymbol{q}(i) = \frac{\tilde{\ell}_i(\boldsymbol{G}_{[2]}^{\neq n})}{\sum_{i=1}^{\Pi_{j\neq n} I_j} \tilde{\ell}_i(\boldsymbol{G}_{[2]}^{\neq n})} \geq \frac{1}{3} \frac{\ell_i(\boldsymbol{G}_{[2]}^{\neq n})}{\operatorname{rank}(\boldsymbol{G}_{[2]}^{\neq n})}. \tag{84}$$

In view of Definition 7, Theorem 8 therefore implies that $\boldsymbol{S} \sim \mathcal{D}(J_2, \boldsymbol{q})$ is a leverage score sampling matrix for $(\boldsymbol{G}_{[2]}^{\neq n}, 1/3)$ if the event E_1 is true. From Theorem 19, it then follows that $\mathbb{P}(E_2 \mid E_1) \geq 1 - \delta/2$ if J_2 is chosen as in (31). With the choices of J_1 and J_2 above and the formula (79), we have that $\mathbb{P}(E_2) \geq 1 - \delta$, which is what we wanted to show.

B.5. Proof of Lemma 12

It follows directly from Definitions 1 and 2 that

$$\boldsymbol{G}_{[2]}^{\neq n}(\overline{i_{n+1} \cdots i_N i_1 \cdots i_{n-1}}, \overline{r_{n-1} r_n}) = \sum_{\{r_j\}_{j\neq n-1, n}} \prod_{j=1}^{N-1} \boldsymbol{G}_{[2]}^{(\boldsymbol{w}(j))}(i_{\boldsymbol{w}(j)}, \overline{r_{\boldsymbol{w}(j)} r_{\boldsymbol{w}(j)-1}}), \tag{85}$$

and therefore the columns of $\boldsymbol{G}_{[2]}^{\neq n}$ can be written as

$$\boldsymbol{G}_{[2]}^{\neq n}(:, \overline{r_{n-1} r_n}) = \sum_{\{r_j\}_{j\neq n-1, n}} \bigotimes_{j=1}^{N-1} \boldsymbol{G}_{[2]}^{(\boldsymbol{w}(j))}(:, \overline{r_{\boldsymbol{w}(j)} r_{\boldsymbol{w}(j)-1}}). \tag{86}$$

Let $q \stackrel{\text{def}}{=} \lceil \log_2(N-1) \rceil$. Using the definition of the recursive sketch in Section 3.2 and the factorization in (17), we have

$$\boldsymbol{\Psi} \boldsymbol{G}_{[2]}^{\neq n}(:, \overline{r_{n-1} r_n}) = \boldsymbol{T}^{(q)} \boldsymbol{T}^{(q-1)} \cdots \boldsymbol{T}^{(1)} \boldsymbol{C} \sum_{\{r_j\}_{j \neq n-1, n}} \left(\left(\bigotimes_{j=1}^{N-1} \boldsymbol{G}_{[2]}^{(\boldsymbol{w}(j))}(:, \overline{r_{\boldsymbol{w}(j)} r_{\boldsymbol{w}(j)-1}}) \right) \otimes \boldsymbol{e}_1^{\otimes (2^q - (N-1))} \right). \tag{87}$$

The notation in the equation above is quite cumbersome. In particular, the ordering of the matrices $\boldsymbol{G}_{[2]}^{(j)}$ in the Kronecker product is somewhat awkward. To alleviate the issue somewhat, we define $\boldsymbol{H}^{(j)}$ for $j \in [2^q]$ as we did in Section 4.2.1:

- Let $\boldsymbol{H}^{(1)} \in \mathbb{R}^{I_{n-1} \times R_{n-2}}$ be a matrix with columns $\boldsymbol{H}^{(1)}(:, k) \stackrel{\text{def}}{=} \boldsymbol{G}_{[2]}^{(n-1)}(:, \overline{r_{n-1} k})$ for $k \in [R_{n-2}]$.
- Let $\boldsymbol{H}^{(j)} \stackrel{\text{def}}{=} \boldsymbol{G}_{[2]}^{(\boldsymbol{w}(j))} \in \mathbb{R}^{I_{\boldsymbol{w}(j)} \times R_{\boldsymbol{w}(j)} R_{\boldsymbol{w}(j)-1}}$ for $2 \leq j \leq N-2$.
- Let $\boldsymbol{H}^{(N-1)} \in \mathbb{R}^{I_{n+1} \times R_{n+1}}$ be a matrix with columns $\boldsymbol{H}^{(N-1)}(:, k) \stackrel{\text{def}}{=} \boldsymbol{G}_{[2]}^{(n+1)}(:, \overline{k r_n})$ for $k \in [R_{n+1}]$.
- Let $\boldsymbol{H}^{(j)} \stackrel{\text{def}}{=} \boldsymbol{e}_1 \in \mathbb{R}^{\max_{j \neq n} I_j}$ be a column vector for $N \leq j \leq 2^q$.

Moreover, we also define the numbers $K_j^{(0)}$ for $j \in [2^q + 1]$ as in Section 4.2.1:

$$K_j^{(0)} \stackrel{\text{def}}{=} \begin{cases} R_{\boldsymbol{w}(j)} & \text{if } 2 \leq j \leq N-1, \\ 1 & \text{otherwise.} \end{cases} \tag{88}$$

With this new notation, we can write (87) as

$$\boldsymbol{\Psi} \boldsymbol{G}_{[2]}^{\neq n}(:, \overline{r_{n-1} r_n}) = \boldsymbol{T}^{(q)} \boldsymbol{T}^{(q-1)} \cdots \boldsymbol{T}^{(1)} \boldsymbol{C} \sum_{\{k_j\}_{j=1}^{2^q+1}} \bigotimes_{j=1}^{2^q} \boldsymbol{H}^{(j)}(:, \overline{k_j k_{j+1}}), \tag{89}$$

where each summation index k_j goes over values $k_j \in [K_j^{(0)}]$. Using Lemma 16, Equation (89) can be written as

$$\begin{aligned} \boldsymbol{\Psi} \boldsymbol{G}_{[2]}^{\neq n}(:, \overline{r_{n-1} r_n}) &= \boldsymbol{T}^{(q)} \boldsymbol{T}^{(q-1)} \cdots \boldsymbol{T}^{(1)} \sum_{\{k_j\}_{j=1}^{2^q+1}} \bigotimes_{j=1}^{2^q} \boldsymbol{C}_j \boldsymbol{H}^{(j)}(:, \overline{k_j k_{j+1}}) \\ &= \boldsymbol{T}^{(q)} \boldsymbol{T}^{(q-1)} \cdots \boldsymbol{T}^{(1)} \sum_{\{k_j\}_{j=1}^{2^q+1}} \bigotimes_{j=1}^{2^q} \boldsymbol{Y}_j^{(0)}(:, \overline{k_j k_{j+1}}), \end{aligned} \tag{90}$$

where $\boldsymbol{Y}_j^{(0)}$ was defined in (33). Recalling that $\boldsymbol{T}^{(1)} \stackrel{\text{def}}{=} \bigotimes_{j=1}^{2^{q-1}} \boldsymbol{T}_j^{(1)}$, we may further rewrite (90) as

$$\begin{aligned} \boldsymbol{\Psi} \boldsymbol{G}_{[2]}^{\neq n}(:, \overline{r_{n-1} r_n}) &= \boldsymbol{T}^{(q)} \boldsymbol{T}^{(q-1)} \cdots \boldsymbol{T}^{(2)} \left(\bigotimes_{j=1}^{2^{q-1}} \boldsymbol{T}_j^{(1)} \right) \sum_{\{k_j\}_{j=1}^{2^q+1}} \bigotimes_{j=1}^{2^{q-1}} (\boldsymbol{Y}_{2j-1}^{(0)}(:, \overline{k_{2j-1} k_{2j}}) \otimes \boldsymbol{Y}_{2j}^{(0)}(:, \overline{k_{2j} k_{2j+1}})) \\ &= \boldsymbol{T}^{(q)} \boldsymbol{T}^{(q-1)} \cdots \boldsymbol{T}^{(2)} \sum_{\{k_{2j-1}\}_{j=1}^{2^{q-1}+1}} \bigotimes_{j=1}^{2^{q-1}} \sum_{k_{2j}} \boldsymbol{T}_j^{(1)} (\boldsymbol{Y}_{2j-1}^{(0)}(:, \overline{k_{2j-1} k_{2j}}) \otimes \boldsymbol{Y}_{2j}^{(0)}(:, \overline{k_{2j} k_{2j+1}})) \\ &= \boldsymbol{T}^{(q)} \boldsymbol{T}^{(q-1)} \cdots \boldsymbol{T}^{(2)} \sum_{\{k_{2j-1}\}_{j=1}^{2^{q-1}+1}} \bigotimes_{j=1}^{2^{q-1}} \boldsymbol{Y}_j^{(1)}(:, \overline{k_{2j-1} k_{2j+1}}), \end{aligned} \tag{91}$$

where the second equality follows from Lemma 16, and each $\boldsymbol{Y}_j^{(1)}$ is defined as in (35). Defining $K_j^{(1)} \stackrel{\text{def}}{=} K_{2j-1}^{(0)}$ for $j \in [2^{q-1} + 1]$, we can further rewrite the equation above as

$$\boldsymbol{\Psi} \boldsymbol{G}_{[2]}^{\neq n}(:, \overline{r_{n-1} r_n}) = \boldsymbol{T}^{(q)} \boldsymbol{T}^{(q-1)} \cdots \boldsymbol{T}^{(2)} \sum_{\{k_j\}_{j=1}^{2^{q-1}+1}} \bigotimes_{j=1}^{2^{q-1}} \boldsymbol{Y}_j^{(1)}(:, \overline{k_j k_{j+1}}), \tag{92}$$

where each summation index k_j now goes over the values $k_j \in [K_j^{(1)}]$. In general, for $m \in [q]$, we have

$$\begin{aligned}
&\boldsymbol{T}^{(q)}\boldsymbol{T}^{(q-1)}\cdots\boldsymbol{T}^{(m)} \sum_{\{k_j\}_{j=1}^{2^{q-m+1}+1}} \bigotimes_{j=1}^{2^{q-m+1}} \boldsymbol{Y}_j^{(m-1)}(:, \overline{k_j k_{j+1}}) \\
&= \boldsymbol{T}^{(q)}\boldsymbol{T}^{(q-1)}\cdots\boldsymbol{T}^{(m+1)} \sum_{\{\ell_j\}_{j=1}^{2^{q-m}+1}} \bigotimes_{j=1}^{2^{q-m}} \boldsymbol{Y}_j^{(m)}(:, \overline{\ell_j \ell_{j+1}}),
\end{aligned} \tag{93}$$

where the summation indices k_j and ℓ_j take on values $k_j \in [K_j^{(m-1)}]$ and $\ell_j \in [K_j^{(m)}]$, respectively, where $K_j^{(m)} \stackrel{\text{def}}{=} K_{2j-1}^{(m-1)}$ for $j \in [2^{q-m}+1]$, and where each $\boldsymbol{Y}_j^{(m)}$ is defined as in (35). Combining (92) and (93), it follows by induction that

$$\boldsymbol{\Psi}\boldsymbol{G}_{[2]}^{\neq n}(:, \overline{r_{n-1}r_n}) = \sum_{k_1 \in [K_1^{(q)}]} \sum_{k_2 \in [K_2^{(q)}]} \boldsymbol{Y}_1^{(q)}(:, \overline{k_1 k_2}) = \boldsymbol{Y}_1^{(q)}, \tag{94}$$

where the last equality follows since $K_1^{(q)} = K_2^{(q)} = 1$.

B.6. Proof of Lemma 13

Throughout the following computations the summation indices go over $i = \overline{i_{n+1}\cdots i_N i_1 \cdots i_{n-1}} \in [\prod_{j\neq n} I_j]$ with $i_j \in [I_j]$ and $r_j, k_j \in [R_j]$ for each $j \in [N]$. Recall that $\boldsymbol{\Phi} \stackrel{\text{def}}{=} \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1})^\top$, where $\boldsymbol{\Psi}\boldsymbol{G}_{[2]}^{\neq n} = \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^\top$ is a compact SVD. From (20) we have

$$\begin{aligned}
\tilde{\ell}_i(\boldsymbol{G}_{[2]}^{\neq n}) &= \boldsymbol{e}_i^\top \boldsymbol{G}_{[2]}^{\neq n}\boldsymbol{\Phi}\boldsymbol{G}_{[2]}^{\neq n\top}\boldsymbol{e}_i = \big(\boldsymbol{G}_{[2]}^{\neq n}\boldsymbol{\Phi}\boldsymbol{G}_{[2]}^{\neq n\top}\big)(i,i) \\
&= \sum_{\substack{r_{n-1},r_n \\ k_{n-1},k_n}} \boldsymbol{G}_{[2]}^{\neq n}(i, \overline{r_{n-1}r_n})\boldsymbol{\Phi}(\overline{r_{n-1}r_n}, \overline{k_{n-1}k_n})\boldsymbol{G}_{[2]}^{\neq n}(i, \overline{k_{n-1}k_n}).
\end{aligned} \tag{95}$$

From Definitions 1 and 2 it follows that[6]

$$\boldsymbol{G}_{[2]}^{\neq n}(i, \overline{r_{n-1}r_n}) = \boldsymbol{G}_{[2]}^{\neq n}(\overline{i_{n+1}\cdots i_N i_1 \cdots i_{n-1}}, \overline{r_{n-1}r_n}) = \sum_{\{r_j\}_{j\neq n-1,n}} \prod_{j\neq n} \boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{r_j r_{j-1}}). \tag{96}$$

Using (95) and (96), we have

$$\begin{aligned}
C &\stackrel{\text{def}}{=} \sum_i \tilde{\ell}_i(\boldsymbol{G}_{[2]}^{\neq n}) \\
&= \sum_i \sum_{\substack{r_{n-1},r_n \\ k_{n-1},k_n}} \Big(\sum_{\{r_j\}_{j\neq n-1,n}} \prod_{j\neq n} \boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{r_j r_{j-1}}) \Big) \boldsymbol{\Phi}(\overline{r_{n-1}r_n}, \overline{k_{n-1}k_n}) \Big(\sum_{\{k_j\}_{j\neq n-1,n}} \prod_{j\neq n} \boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{k_j k_{j-1}}) \Big) \\
&= \sum_{\substack{r_1,\ldots,r_N \\ k_1,\ldots,k_N}} \boldsymbol{\Phi}(\overline{r_{n-1}r_n}, \overline{k_{n-1}k_n}) \prod_{j\neq n} \Big(\sum_{i_j} \boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{r_j r_{j-1}}) \boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{k_j k_{j-1}}) \Big) \\
&= \sum_{\substack{r_1,\ldots,r_N \\ k_1,\ldots,k_N}} \boldsymbol{\Phi}(\overline{r_{n-1}r_n}, \overline{k_{n-1}k_n}) \prod_{j\neq n} \big(\boldsymbol{G}_{[2]}^{(j)\top}\boldsymbol{G}_{[2]}^{(j)} \big)(\overline{r_j r_{j-1}}, \overline{k_j k_{j-1}}),
\end{aligned} \tag{97}$$

which proves the expression in (37).

[6] The only difference between (85) and (96) is that the terms in the product are arranged in a different order.

Moreover, using (95) and (96) we have that the marginal probability of drawing $(i_j)_{j\leq m, j\neq n}$ is

$$\begin{aligned}
\mathbb{P}((i_j)_{j\leq m,j\neq n}) &= \frac{1}{C}\sum_{\{i_j\}_{j>m,j\neq n}} \tilde{\ell}_i(\boldsymbol{G}_{[2]}^{\neq n}) \\
&= \frac{1}{C}\sum_{\{i_j\}_{j>m,j\neq n}} \sum_{\substack{r_{n-1},r_n \\ k_{n-1},k_n}} \boldsymbol{\Phi}(\overline{r_{n-1}r_n}, \overline{k_{n-1}k_n})\Big(\sum_{\{r_j\}_{j\neq n-1,n}} \prod_{j\neq n} \boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{r_j r_{j-1}})\Big)\Big(\sum_{\{k_j\}_{j\neq n-1,n}} \prod_{j\neq n} \boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{k_j k_{j-1}})\Big) \\
&= \frac{1}{C}\sum_{\substack{r_1,\dots,r_N \\ k_1,\dots,k_N}} \boldsymbol{\Phi}(\overline{r_{n-1}r_n}, \overline{k_{n-1}k_n})\Big(\prod_{\substack{j\leq m \\ j\neq n}} \boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{r_j r_{j-1}})\boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{k_j k_{j-1}})\Big)\Big(\prod_{\substack{j>m \\ j\neq n}} (\boldsymbol{G}_{[2]}^{(j)\top}\boldsymbol{G}_{[2]}^{(j)})(\overline{r_j r_{j-1}}, \overline{k_j k_{j-1}})\Big),
\end{aligned} \tag{98}$$

which proves (38).

C. Detailed Complexity Analysis

C.1. CP-ALS-ES: Proposed Sampling Scheme for CP Decomposition

In this section we derive the computational complexity of the scheme proposed in Section 4.1.

Computing $\boldsymbol{\Psi}\boldsymbol{A}^{\neq n}$ First, we consider the costs of computing $\boldsymbol{\Psi}\boldsymbol{A}^{\neq n}$ as described in Section 4.1.1:

- Computing $\boldsymbol{Y}_j^{(0)}$ for all $j \in [2^q]$: Each $\boldsymbol{C}_j\boldsymbol{A}^{(\boldsymbol{v}(j))}$ costs at most $O(I_{\boldsymbol{v}(j)}R)$ to compute, and each $\boldsymbol{C}_j(\boldsymbol{e}_1\boldsymbol{1}_{1\times R})$ costs $O(R)$ to compute. Since $2^q \leq 2N$, the total cost for this step is therefore $O(R\sum_{j\neq n} I_j)$.

- Computing $\boldsymbol{Y}_j^{(m)}$ for all $m \in [q]$ and all $j \in [2^{q-m}]$: A single $J_1 \times J_1^2$ TensorSketch costs $O(RJ_1 \log J_1)$ to apply to a matrix of the form $\boldsymbol{Y}_{2j-1}^{(m-1)} \odot \boldsymbol{Y}_{2j}^{(m-1)}$. Such a TensorSketch is applied a total of $\sum_{m=1}^{q} 2^{q-m} = 2^q - 1 = O(N)$ times, so the total cost of this whole step is therefore $O(NRJ_1 \log J_1)$.

The cost for computing $\boldsymbol{\Psi}\boldsymbol{A}^{\neq n}$ is therefore

$$O\Big(R\Big(NJ_1 \log J_1 + \sum_{j\neq n} I_j\Big)\Big). \tag{99}$$

Drawing J_2 Samples Second, we consider the cost of drawing J_2 samples in $[\prod_{j\neq n} I_j]$ from the distribution $\boldsymbol{q}$ as described in Section 4.1.2:

- One-time costs: Computing the SVD of $\boldsymbol{\Psi}\boldsymbol{A}^{\neq n}$ costs $O(J_1R^2)$. Computing $\boldsymbol{\Phi} = \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1})^\top$ costs $O(R^3)$. Moreover, we can compute all products $\boldsymbol{A}^{(j)\top}\boldsymbol{A}^{(j)}$ for $j \neq n$ upfront for a cost of $O(R^2\sum_{j\neq n} I_j)$. The sum of these one-time costs is $O(R^2(J_1 + R + \sum_{j\neq n} I_j))$.

- Cost of sampling J_2 indices: Since each $\boldsymbol{A}^{(j)\top}\boldsymbol{A}^{(j)}$ for $j \neq n$ has already been computed, the cost of computing the probability $\mathbb{P}(i_m \mid (i_j)_{j<m,j\neq n})$ for a single set $(i_j)_{j\leq m,j\neq n}$ via (28) and (29) is $O(R^2N)$. The total cost for computing the whole distribution for $i_m \in [I_m]$, for all $m \in [N]\setminus\{n\}$, is therefore $O(R^2N\sum_{j\neq n} I_j)$. Since the main cost of sampling an index $i = \overline{i_1\cdots i_{n-1}i_{n+1}\cdots i_N}$ is computing the distribution for each subindex, and we need to sample a total of J_2 samples, it follows that the total cost of drawing J_2 samples is $O(J_2R^2N\sum_{j\neq n} I_j)$.

In total, when including both one-time and per-sample costs, we get a cost for drawing J_2 samples from $\boldsymbol{q}$ of

$$O\Big(R^2\Big(J_1 + R + J_2N\sum_{j\neq n} I_j\Big)\Big). \tag{100}$$

Sampled Least Squares Problem Finally, we consider the cost of constructing and solving the sampled least squares problem once the J_2 samples in $[\prod_{j\neq n} I_j]$ have been drawn:

- Once the J_2 samples in $[\prod_{j\neq n} I_j]$ are drawn, it costs $O(J_2RN)$ to form $\boldsymbol{S}\boldsymbol{A}^{\neq n}$, and $O(J_2 I_n)$ to form $\boldsymbol{S}\boldsymbol{X}_{(n)}^\top$. This can be done implicitly without forming the matrices $\boldsymbol{S}$, $\boldsymbol{A}^{\neq n}$, and $\boldsymbol{X}_{(n)}^\top$.
- The cost of computing the solution $\tilde{\boldsymbol{A}}^\top = (\boldsymbol{S}\boldsymbol{A}^{\neq n})^\dagger \boldsymbol{S}\boldsymbol{X}_{(n)}^\top$ using a standard method (e.g., via QR decomposition) is $O(J_2R^2 + J_2RI_n)$; see Section 5.3.3 in Golub & Van Loan (2013) for details.

In total, the costs of constructing and solving the least squares problem is therefore

$$O(J_2R(N + R + I_n)). \tag{101}$$

Total Per-Iteration Cost for CP-ALS-ES Recall that for each iteration of CP-ALS, we need to solve $N-1$ least squares problems. Consequently, adding the costs in (99), (100), (101) and multiplying by $N-1$, we get a total cost per iteration of

$$O\Big(RN^2 J_1 \log J_1 + R^2 N\Big(J_1 + R + J_2 N \sum_{j\neq n} I_j\Big) + J_2 RNI_n\Big). \tag{102}$$

If the sketch rates J_1 and J_2 are chosen according to (22) and (23), this per-iteration cost becomes

$$O\Big(\frac{R^3N^3}{\delta}\log\Big(\frac{R^2N}{\delta}\Big) + \frac{R^4N^2}{\delta} + \Big(R^3N^2\sum_{j\neq n} I_j + R^2NI_n\Big)\max\Big(\log\Big(\frac{R}{\delta}\Big), \frac{1}{\varepsilon\delta}\Big)\Big). \tag{103}$$

C.2. TR-ALS-ES: Proposed Sampling Scheme for Tensor Ring Decomposition

In this section we derive the computational complexity of the scheme proposed in Section 4.2.

Computing $\boldsymbol{\Psi}\boldsymbol{G}_{[2]}^{\neq n}$ First, we consider the computation of $\boldsymbol{\Psi}\boldsymbol{G}_{[2]}^{\neq n}$ described in Section 4.2.1:

- Computing $\boldsymbol{Y}_j^{(0)}$ for all $j \in [2^q]$: Recall that computing $\boldsymbol{C}_j\boldsymbol{H}^{(j)}$ costs $\operatorname{nnz}(\boldsymbol{H}^{(j)})$. Consequently, the cost of computing all $\boldsymbol{Y}_j^{(0)}$ for $N \leq j \leq 2^q$ is just $O(1)$. The total cost for this step is therefore $O(\sum_{j=1}^N I_j R_{j-1} R_j)$.
- Computing $\boldsymbol{Y}_j^{(m)}$ for all $m \in [q]$ and all $j \in [2^{q-m}]$: Computing $\boldsymbol{T}_j^{(m)}(\boldsymbol{Y}_{2j-1}^{(m-1)}(:, \overline{k_1k_2}) \otimes \boldsymbol{Y}_{2j}^{(m-1)}(:, \overline{k_2k_3}))$ requires applying a $J_1 \times J_1^2$ TensorSketch to the Kronecker product of two vectors, which costs $O(J_1 \log J_1)$. This needs to be done for each $k_2 \in [K_{2j}^{(m-1)}]$ when computing the sum in (35). This sum, in turn, needs to be computed for all $k_1 \in [K_{2j-1}^{(m-1)}]$, $k_3 \in [K_{2j+1}^{(m-1)}]$ and $j \in [2^{q-m}]$. Doing this for each $m \in [q]$ brings the total cost of this step to

$$O\Big(\sum_{m=1}^{q}\sum_{j=1}^{2^{q-m}}\sum_{k_1=1}^{K_{2j-1}^{(m-1)}}\sum_{k_2=1}^{K_{2j}^{(m-1)}}\sum_{k_3=1}^{K_{2j+1}^{(m-1)}} J_1 \log J_1\Big). \tag{104}$$

  As we will see further down, this expression simplifies considerably if all R_i are assumed to be equal.

Adding up the per-column costs above and multiplying them by the number of columns $R_{n-1}R_n$, we get that the cost for computing $\boldsymbol{\Psi}\boldsymbol{G}_{[2]}^{\neq n}$ is

$$O\Big(R_{n-1}R_n\Big(\sum_{j=1}^{N} I_j R_{j-1} R_j + \sum_{m=1}^{q}\sum_{j=1}^{2^{q-m}}\sum_{k_1=1}^{K_{2j-1}^{(m-1)}}\sum_{k_2=1}^{K_{2j}^{(m-1)}}\sum_{k_3=1}^{K_{2j+1}^{(m-1)}} J_1 \log J_1\Big)\Big). \tag{105}$$

Drawing J_2 Samples Second, we consider the cost of drawing J_2 samples in $[\prod_{j\neq n} I_j]$ from the distribution $\boldsymbol{q}$ as described in Section 4.2.2:

- One-time costs: Computing the SVD of $\boldsymbol{\Psi}\boldsymbol{G}_{[2]}^{\neq n}$ costs $O(J_1(R_{n-1}R_n)^2)$. Computing $\boldsymbol{\Phi} = \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1})^\top$ costs $O((R_{n-1}R_n)^3)$. Moreover, we can compute all products $\boldsymbol{G}_{[2]}^{(j)\top}\boldsymbol{G}_{[2]}^{(j)}$ for $j \neq n$ upfront for a cost of $O(\sum_{j\neq n}(R_{j-1}R_j)^2 I_j)$. The sum of these one-time costs is $O(J_1(R_{n-1}R_n)^2 + (R_{n-1}R_n)^3 + \sum_{j\neq n}(R_{j-1}R_j)^2 I_j)$.
- Cost of sampling J_2 indices: The main cost of drawing the samples is computing the sampling distributions. Even though the number of terms in the sum of (38) is exponential in N, the joint probability distribution can be computed efficiently. We discuss how to do this in Remark 20. The cost of doing this for one set of indices $(i_j)_{j\leq m, j\neq n}$ is given in (119). Repeating this for all $i_j \in [I_j]$, which is required to get the distribution for the jth index, brings the cost to

$$O\Big(I_j R_N^2 \sum_{d=1}^{N-1} R_d^2 R_{d+1}^2\Big). \tag{106}$$

  When this is repeated for all N indices, and a total of J_2 times to get all samples, this brings the cost to

$$O\bigg(J_2\Big(\sum_{j=1}^{N} I_j\Big) R_N^2 \sum_{d=1}^{N-1} R_d^2 R_{d+1}^2\bigg). \tag{107}$$

Adding the one-time costs and the costs associated to computing the distributions, we get the following total cost for drawing J_2 samples:

$$O\bigg(J_1(R_{n-1}R_n)^2 + (R_{n-1}R_n)^3 + J_2\Big(\sum_{j=1}^{N} I_j\Big) R_N^2 \sum_{d=1}^{N-1} R_d^2 R_{d+1}^2\bigg). \tag{108}$$

Sampled Least Squares Problem Finally, we consider the cost of constructing and solving the sampled least squares problem once the J_2 samples in $[\prod_{j\neq n} I_j]$ have been drawn:

- Once J_2 samples in $[\prod_{j\neq n} I_j]$ are drawn, the sketched design matrix $\boldsymbol{S}\boldsymbol{G}_{[2]}^{\neq n}$ can be computed efficiently without having to form the full matrix $\boldsymbol{G}_{[2]}^{\neq n}$. We provide further details in Remark 21. With this approach, the cost of forming $\boldsymbol{S}\boldsymbol{G}_{[2]}^{\neq n}$ is

$$O\Big(J_2 R_n \sum_{j\in[N]\setminus\{n,n+1\}} R_{j-1}R_j\Big). \tag{109}$$

  Forming $\boldsymbol{S}\boldsymbol{X}_{[n]}^\top$ by sampling the appropriate rows costs $O(J_2 I_n)$.

- The cost of computing the solution $\tilde{\boldsymbol{G}}^\top = (\boldsymbol{S}\boldsymbol{G}_{[2]}^{\neq n})^\dagger \boldsymbol{S}\boldsymbol{X}_{[n]}^\top$ using a standard method (e.g., via QR decomposition) is $O(J_2(R_{n-1}R_n)^2 + J_2 R_{n-1}R_n I_n)$; see Section 5.3.3 in Golub & Van Loan (2013) for details.

In total, the cost of constructing and solving the least squares problem is therefore

$$O\bigg(J_2\Big(R_n \sum_{j\in[N]\setminus\{n,n+1\}} R_{j-1}R_j + (R_{n-1}R_n)^2 + R_{n-1}R_n I_n\Big)\bigg). \tag{110}$$

Total Per-Iteration Cost for TR-ALS-ES Recall that for each iteration of TR-ALS, we need to solve $N-1$ least squares problems. Consequently, adding the costs in (105), (108), (110) and multiplying by $N-1$, we get the total per-iteration cost. If we assume that $R_j = R$ and $I_j = I$ for all $j \in [N]$, the expression simplifies considerably and we get a total per-iteration cost of

$$O(N^2 R^5 J_1 \log J_1 + N^3 I R^6 J_2). \tag{111}$$

If the sketch rates J_1 and J_2 are chosen according to (30) and (31), this per-iteration cost becomes

$$O\bigg(\frac{N^3R^9}{\delta}\log\Big(\frac{NR^4}{\delta}\Big) + N^3 I R^8 \cdot \max\Big(\log\Big(\frac{R^2}{\delta}\Big), \frac{1}{\varepsilon\delta}\Big)\bigg). \tag{112}$$

Remark 20. At first sight, the joint probability computation in (38) looks expensive since the number of terms in the sum is exponential in N. However, since not all summation indices r_j and k_j appear in every term, the summation can be done more efficiently. In fact, the computation (38) can be viewed as the evaluation of a tensor ring, which can be done efficiently by contracting core tensors pairwise. To see this, define core tensors $\mathcal{C}^{(j)}$ for $j \in [N]$ as follows:

- For $j \leq m$ and $j \neq n$, let $\mathcal{C}^{(j)} \in \mathbb{R}^{R_{j-1}^2 \times I_j \times R_j^2}$ be defined elementwise via

$$\mathcal{C}^{(j)}(\overline{r_{j-1}k_{j-1}}, i_j, \overline{r_j k_j}) \stackrel{\text{def}}{=} \boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{r_j r_{j-1}}) \boldsymbol{G}_{[2]}^{(j)}(i_j, \overline{k_j k_{j-1}}). \tag{113}$$

- For $m < j \leq N$ and $j \neq n$, let $\mathcal{C}^{(j)} \in \mathbb{R}^{R_{j-1}^2 \times 1 \times R_j^2}$ be defined elementwise via

$$\mathcal{C}^{(j)}(\overline{r_{j-1}k_{j-1}}, 1, \overline{r_j k_j}) \stackrel{\text{def}}{=} \big(\boldsymbol{G}_{[2]}^{(j)\top} \boldsymbol{G}_{[2]}^{(j)}\big)(\overline{r_j r_{j-1}}, \overline{k_j k_{j-1}}). \tag{114}$$

- For $j = n$, let $\mathcal{C}^{(j)} = \mathcal{C}^{(n)} \in \mathbb{R}^{R_{n-1}^2 \times 1 \times R_n^2}$ be defined elementwise via

$$\mathcal{C}^{(n)}(\overline{r_{n-1}k_{n-1}}, 1, \overline{r_n k_n}) \stackrel{\text{def}}{=} \frac{1}{C} \boldsymbol{\Phi}(\overline{r_{n-1}r_n}, \overline{k_{n-1}k_n}). \tag{115}$$

We can now rewrite the expression in (38) as

$$\mathbb{P}((i_j)_{j \leq m, j \neq n}) = \mathrm{TR}(\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(N)})_{\xi_1, \ldots, \xi_N}, \tag{116}$$

where

$$\xi_j \stackrel{\text{def}}{=} \begin{cases} i_j & \text{if } j \leq m, j \neq n, \\ 1 & \text{otherwise.} \end{cases} \tag{117}$$

As discussed in Zhao et al. (2016), the value of an entry in a tensor ring can be computed via a sequence of matrix-matrix products follows by taking the matrix trace:

$$\mathrm{TR}(\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(N)})_{\xi_1, \ldots, \xi_N} = \mathrm{trace}\left(\mathcal{C}^{(1)}(:, \xi_1, :) \cdot \mathcal{C}^{(2)}(:, \xi_2, :) \cdots \mathcal{C}^{(N)}(:, \xi_N, :)\right), \tag{118}$$

where each $\mathcal{C}^{(j)}(:, \xi_j, :)$ is treated as a $R_{j-1}^2 \times R_j^2$ matrix. If the matrix product in (118) is done left to right, evaluating the right hand side costs

$$O\Big(R_N^2 \sum_{j=1}^{N-1} R_j^2 R_{j+1}^2\Big). \tag{119}$$

Remark 21. As described by Malik & Becker (2021), it is possible to construct the sketched design matrix $\boldsymbol{S}\boldsymbol{G}_{[2]}^{\neq n}$ efficiently without first forming the full matrix $\boldsymbol{G}_{[2]}^{\neq n}$. To see how, note that each row $\boldsymbol{G}_{[2]}^{\neq n}(i, :)$ is the vectorization of the tensor slice $\mathcal{G}^{\neq n}(:, i, :)$ due to Definition 1. From Definition 2, the tensor slice $\mathcal{G}^{\neq n}(:, i, :)$ is given by

$$\mathcal{G}^{\neq n}(:, \overline{i_{n+1} \cdots i_N i_1 \cdots i_{n-1}}, :) = \mathcal{G}^{(n+1)}(:, i_{n+1}, :) \cdots \mathcal{G}^{(N)}(:, i_N, :) \cdot \mathcal{G}^{(1)}(:, i_1, :) \cdots \mathcal{G}^{(n-1)}(:, i_{n-1}, :). \tag{120}$$

Suppose $\boldsymbol{v} \in [\prod_{j \neq n} I_j]^{J_2}$ contains the J_2 sampled indices corresponding to the sketch $\boldsymbol{S}$. Let $\tilde{\mathcal{G}}^{\neq n} \in \mathbb{R}^{R_n \times J_2 \times R_{n-1}}$ be a tensor which we define as follows: For each $j \in [J_2]$, let $i = \overline{i_{n+1} \cdots i_N i_1 \cdots i_{n-1}} \stackrel{\text{def}}{=} \boldsymbol{v}(j)$ and define

$$\tilde{\mathcal{G}}^{\neq n}(:, j, :) \stackrel{\text{def}}{=} \frac{1}{\sqrt{J_2 \boldsymbol{q}(i)}} \mathcal{G}^{(n+1)}(:, i_{n+1}, :) \cdots \mathcal{G}^{(N)}(:, i_N, :) \cdot \mathcal{G}^{(1)}(:, i_1, :) \cdots \mathcal{G}^{(n-1)}(:, i_{n-1}, :). \tag{121}$$

We now have $\boldsymbol{S}\boldsymbol{G}_{[2]}^{\neq n} = \tilde{\boldsymbol{G}}_{[2]}^{\neq n}$. If the matrix product in (121) is computed from left to right, it costs $O(R_n \sum_{j \in [N] \setminus \{n, n+1\}} R_{j-1} R_j)$. Since this needs to be computed for each $j \in [J_2]$, the total cost for computing $\boldsymbol{S}\boldsymbol{G}_{[2]}^{\neq n}$ via this scheme is

$$O\Big(J_2 R_n \sum_{j \in [N] \setminus \{n, n+1\}} R_{j-1} R_j\Big). \tag{122}$$

We refer the reader to Malik & Becker (2021) for further details.

C.3. Complexity Analysis of Competing Methods

In this section we provide a few notes on how we computed the computational complexity of the other methods we compare with in Tables 1 and 2.

C.3.1. CP-ALS

The standard way to implement CP-ALS is given in Figure 3.3 in Kolda & Bader (2009). The leading order cost per least squares solve for that algorithm is

$$O(NIR^2 + R^3 + NI^{N-1}R + I^N R). \tag{123}$$

Since N such least squares problems need to be solved each iteration, the per-iteration cost is

$$O(N^2IR^2 + NR^3 + N^2I^{N-1}R + NI^N R). \tag{124}$$

When N is large, this becomes $O(N(N+I)I^{N-1}R)$ which is what we report in Table 1.

C.3.2. SPALS

Cheng et al. (2016) only give the sampling complexity for the case when $N = 3$ in their paper. For arbitrary N, and without any assumptions on the rank of the factor matrices or the Khatri–Rao product design matrix, their scheme requires $J \gtrsim R^N \log(I_n/\delta)/\varepsilon^2$ samples when solving for the nth factor matrix in order to achieve the additive error guarantees in Theorem 4.1 of their paper.[7]

SPALS requires a one-time upfront cost of $\operatorname{nnz}(\mathcal{X})$ in order to compute the second term in Equation (5) in Cheng et al. (2016). In SPALS, the nth factor is updated via

$$\boldsymbol{A}^{(n)} = \boldsymbol{X}_{(n)}\boldsymbol{S}^\top \boldsymbol{S}\Big(\bigodot_{\substack{j=N\\ j\neq n}}^{1} \boldsymbol{A}^{(j)}\Big)\Big(\circledast_{\substack{j=1\\ j\neq n}}^{N} \boldsymbol{A}^{(j)\top}\boldsymbol{A}^{(j)}\Big)^{-1}, \tag{125}$$

where $\boldsymbol{S}$ is a sampling matrix and $\circledast$ denotes elementwise (Hadamard) product. When this is computed in the appropriate order, and if $\log$ factors are ignored and we assume that $I_n = I$ for all $n \in [N]$, then the cost of computing $\boldsymbol{A}^{(n)}$ is

$$\tilde{O}(NIR^2 + (N+I)R^{N+1}/\varepsilon^2). \tag{126}$$

Notice that the cost of computing the sampling distribution is dominated by the cost above. Since N factor matrices need to be updated per iteration, the total per-iteration cost is

$$\tilde{O}(N^2IR^2 + N(N+I)R^{N+1}/\varepsilon^2). \tag{127}$$

When N is large, this becomes $\tilde{O}(N(N+I)R^{N+1}/\varepsilon^2)$, which is what we report in Table 1.

C.3.3. CP-ARLS-LEV

From Theorem 8 in Larsen & Kolda (2020), the sampling complexity for CP-ARLS-LEV required to achieve relative error guarantees when solving for the nth factor matrix is $J \gtrsim R^{N-1}\max(\log(R/\delta), 1/(\delta\varepsilon))$. Solving the sampled least squares problem, which has a design matrix of size $J \times R$ and I_n right hand sides via e.g. QR decomposition (see Section 5.3.3 in Golub & Van Loan (2013)) will therefore cost $O((R+I_n)R^N \max(\log(R/\delta), 1/(\delta\varepsilon)))$. Each iteration requires solving N such least squares problems. If we assume that $I_n = I$ for all $n \in [N]$ and ignore $\log$ factors, the per-iteration cost becomes

$$\tilde{O}\big(N(R+I)R^N/(\delta\varepsilon)\big), \tag{128}$$

which is what we report in Table 1.

Consider the least squares problem in (6) with the design matrix $\boldsymbol{A}^{\neq n}$ defined in as in (7). When the leverage score sampling distribution is estimated as in CP-ARLS-LEV, the exponential dependence on N in the sampling complexity cannot be improved. The following example provides a concrete example when the exponential dependence is required.

[7] If the Khatri–Rao product design matrix is full rank, which happens if all factor matrices are full rank, then $J \gtrsim R^{N-1}\log(I_n/\delta)/\varepsilon^2$ samples will suffice.

Example 22. Without loss of generality, consider the case $n = N$ in which case the least squares design matrix in (7) is

$$\boldsymbol{A}^{\neq N} = \boldsymbol{A}^{(N-1)} \odot \cdots \odot \boldsymbol{A}^{(1)}. \tag{129}$$

Suppose all $\boldsymbol{A}^{(j)}$ for $j \in [J-1]$ are of size $R \times R$ and defined as

$$\boldsymbol{A}^{(j)} \stackrel{\text{def}}{=} \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Omega}^{(j)} \end{bmatrix} \tag{130}$$

where each $\boldsymbol{\Omega}^{(j)} \in \mathbb{R}^{(R-1)\times(R-1)}$ has i.i.d. standard Gaussian entries. We assume the matrices $\boldsymbol{\Omega}^{(j)}$ are all full-rank, which is true almost surely. The first column and row of $\boldsymbol{A}^{\neq N}$ are $\boldsymbol{e}_1$ and $\boldsymbol{e}_1^\top$, respectively. For $r \geq 2$ we have

$$\boldsymbol{A}^{\neq N}(:, r) = \begin{bmatrix} 0 \\ \boldsymbol{\Omega}^{(N-1)}(:, r) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 0 \\ \boldsymbol{\Omega}^{(1)}(:, r) \end{bmatrix}. \tag{131}$$

Since all the matrices $\boldsymbol{\Omega}^{(j)}$ are full-rank it follows that their Kronecker product is full-rank (this follows from Theorem 4.2.15 in Horn & Johnson (1994)). Since the columns $\boldsymbol{A}^{\neq N}(:, r)$ for $r \geq 2$ are equal to columns of $\boldsymbol{\Omega}^{(N-1)} \otimes \cdots \otimes \boldsymbol{\Omega}^{(1)}$ with some added zeros, it follows that they are linearly independent, and therefore the submatrix $\boldsymbol{\Gamma} \stackrel{\text{def}}{=} (\boldsymbol{A}^{\neq N}(i, j))_{i\geq 2, j \geq 2}$ is full-rank. We may write

$$\boldsymbol{A}^{\neq N} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Gamma} \end{bmatrix} \in \mathbb{R}^{R^{N-1} \times R}. \tag{132}$$

If a sampling matrix $\boldsymbol{S}$ does not sample the first row of $\boldsymbol{A}^{\neq N}$, then $\boldsymbol{S}\boldsymbol{A}^{\neq N}$ will be rank-deficient and relative error guarantees therefore unachievable. Since all $\boldsymbol{A}^{(j)}$ are square and full-rank, the sampling procedure used in CP-ARLS-LEV will sample rows of $\boldsymbol{A}^{\neq N}$ uniformly. In order to sample the first row of $\boldsymbol{A}^{\neq N}$ with probability at least 0.5 with uniform sampling, we clearly need to sample at least half of all rows of $\boldsymbol{A}^{\neq N}$, i.e., we need $J \geq R^{N-1}/2$.

C.3.4. Methods for Tensor Ring Decomposition

The complexities we report in Table 2 for other methods where taken directly from Table 1 in Malik & Becker (2021).

D. Additional Experiment Details

D.1. Details on Algorithm Implementations

Our implementation of CP-ARLS-LEV is based on Algorithm 3 in Larsen & Kolda (2020). We do not use any hybrid-deterministic sampling, but we do combine repeated rows. Some key functionality required for our CP-ALS-ES is written in C and incorporated into Matlab via the MEX interface. Our own TR-ALS-ES is implemented by appropriately modifying the Matlab code for TR-ALS-Sampled by Malik & Becker (2021).

D.2. Datasets

The photo used for the sampling distribution comparison was taken by Sebastian Müller on Unsplash and is available at `https://unsplash.com/photos/l54ZALpH2_I`. We converted this figure to gray scale by averaging the three color channels. We also cropped the image slightly to make the width and height a power of 2. The tensorization is done following the ideas for visual data tensorization discussed in Yuan et al. (2019b). Please see our code for precise details.

The COIL-100 dataset was created by Nene et al. (1996) and is available for download at `https://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php`.

D.3. Sampling Distribution Plots and Computational Time

We have included figures below that compare the sampling distributions used by our methods with those used by the previous state-of-the-art methods in the least squares problem considered in the first experiment in Section 5. For a rank-10 CP decomposition of the tabby cat tensor, Figure 2 shows the exact leverage score distribution ($\boldsymbol{p}$ in Definition 7), the sampling distribution used by CP-ARLS-LEV, and a realization (for $J_1 = 1000$) of the distribution our CP-ALS-ES uses. Figure 3 shows the same things as Figure 2, but for a rank-20 CP decomposition. For a rank-$(3, \ldots, 3)$ tensor ring decomposition of the tabby cat tensor, Figure 4 shows the exact leverage score distribution, the sampling distribution used

by TR-ALS-Sampled, and a realization (for $J_1 = 1000$) of the distribution our TR-ALS-ES uses. Figure 5 shows the same things as Figure 4, but for a rank-$(5, \dots, 5)$ tensor ring decomposition.

Notice that the sampling distribution that our methods use follow the exact leverage score sampling distribution closely. The distributions used by CP-ARLS-LEV and TR-ALS-Sampled are less accurate. In particular, when $R > I = 16$ for the CP decomposition (Figure 3) or when $R_{n-1}R_n > I = 16$ for the tensor ring decomposition (Figure 5), CP-ARLS-LEV and TR-ALS-Sampled sample from *a uniform distribution*. This is not an anomaly, but rather a direct consequence from how those methods estimate the leverage scores. Our proposed methods, by contrast, handle those cases well.

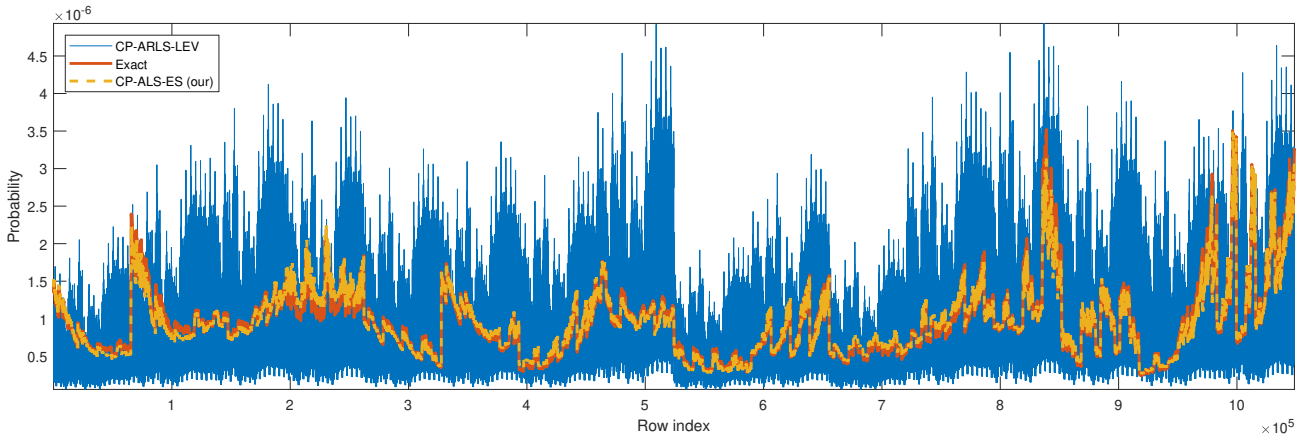

Figure 2. Comparison of the exact leverage score distribution, the sampling distribution used by CP-ARLS-LEV, and a realization (for $J_1 = 1000$) of the distribution used by our CP-ALS-ES. The least squares problem corresponds to solving for the 6th factor matrix in a rank-10 CP decomposition of the 6-way tabby cat tensor.

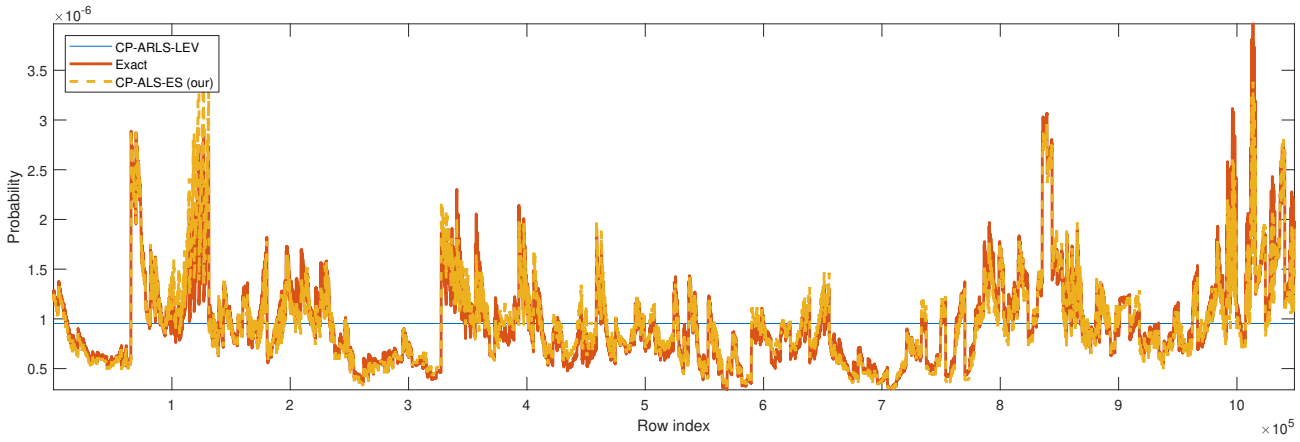

Figure 3. Same as Figure 2, but for a rank-20 decomposition.

Tables 6 and 7 report the time it took to compute the distributions used in Tables 3 and 4, respectively. Note that the different methods do not compute the full distributions the way we do in Tables 3–4 and Figures 2–5, so these numbers are not representative of actual decomposition time and are only added here for completeness.

D.4. Feature Extraction Experiments

We provide some further details on the feature extraction experiments in Section 5.2 in this section. For a rank-25 CP decomposition, the 4th factor matrix is of size 7200×25. We directly use this factor matrix as the feature matrix we feed to the k-NN method in Matlab. For the rank-$(5, \dots, 5)$ tensor ring decomposition, the 4th core tensor is of size $5 \times 7200 \times 5$. We turn this into a 7200×25 matrix via a classical mode-2 unfolding which we then use as the feature matrix in the k-NN algorithm.

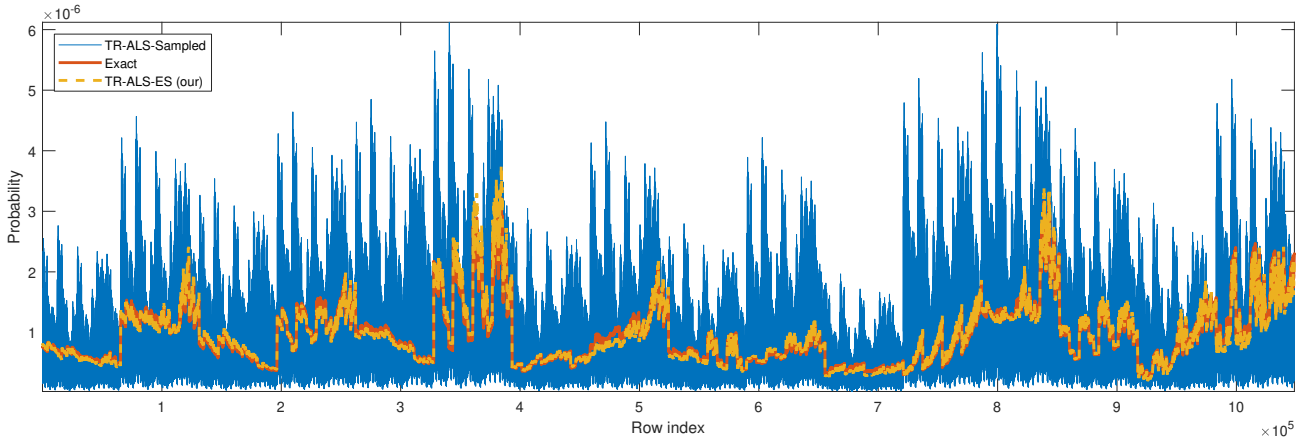

Figure 4. Comparison of the exact leverage score distribution, the sampling distribution used by TR-ALS-Sampled, and a realization (for $J_1 = 1000$) of the distribution used by our TR-ALS-ES. The least squares problem corresponds to solving for the 6th core tensor in a rank-$(3, \ldots, 3)$ tensor ring decomposition of the 6-way tabby cat tensor.

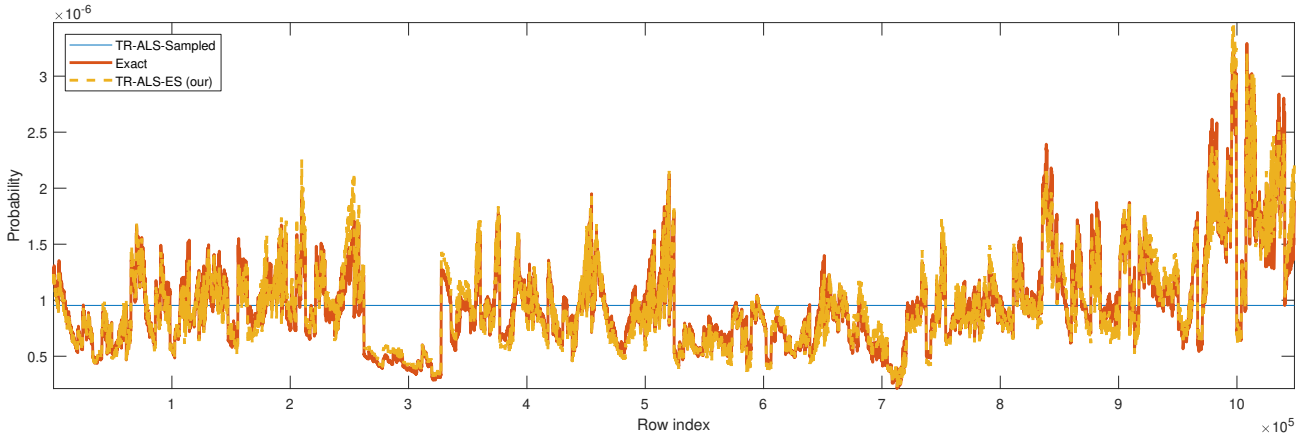

Figure 5. Same as Figure 4, but for a rank-$(5, \ldots, 5)$ decomposition.

Table 6. Time in seconds it took to compute the distributions used in Table 3.

Method	$R = 10$	$R = 20$
CP-ARLS-LEV	0.01	0.01
CP-ALS-ES ($J_1 =$ 1e+4)	0.06	0.12
CP-ALS-ES ($J_1 =$ 1e+3)	0.04	0.07
CP-ALS-ES ($J_1 =$ 1e+2)	0.03	0.07

Table 7. Time in seconds it took to compute the distributions used in Table 4.

Method	$R = 3$	$R = 5$
TR-ALS-Sampled	0.01	0.01
TR-ALS-ES ($J_1 =$ 1e+4)	0.07	0.21
TR-ALS-ES ($J_1 =$ 1e+3)	0.03	0.10
TR-ALS-ES ($J_1 =$ 1e+2)	0.03	0.10

In our feature extraction experiment we assume that both the labeled and unlabeled images are available when the tensor decomposition is computed. This is a limitation since we might want to classify new unlabeled images that arrive after the decomposition has been computed without having to recompute the decomposition. We now propose a potential approach to circumventing this limitation. Adding a new image corresponds to adding new rows to $\mathcal{X}_{(4)}$ and $\mathcal{X}_{[4]}$. The factor matrix $\boldsymbol{A}^{(4)}$ for the CP decomposition of this augmented tensor will have an additional row, while the number of rows will remain the same in the other factor matrices. Similarly, the core tensor $\mathcal{G}^{(4)}$ for the tensor ring decomposition will have an additional lateral slice, while the number of lateral slices will remain the same for the other cores. For the CP decomposition, a feature vector for the new image can therefore be computed via

$$\boldsymbol{a}^* = \arg\min_{\boldsymbol{a}} \|\boldsymbol{A}^{\neq 4}\boldsymbol{a}^\top - \boldsymbol{x}^\top\|_2, \tag{133}$$

where $\boldsymbol{a}^* \in \mathbb{R}^{1\times R}$ is the new row in $\boldsymbol{A}^{(4)}$ and $\boldsymbol{x} \in \mathbb{R}^{1\times 49152}$ is the new row in $\mathcal{X}_{(4)}$. For the tensor train decomposition, a feature vector for the new image can similarly be computed via

$$\boldsymbol{g}^* = \arg\min_{\boldsymbol{g}} \|\boldsymbol{G}_{[2]}^{\neq 4}\boldsymbol{g}^\top - \hat{\boldsymbol{x}}^\top\|_2, \tag{134}$$

where $\boldsymbol{g}^* \in \mathbb{R}^{1\times R_3 R_4}$ is a reshaped version of the new lateral slice in $\mathcal{G}^{(4)}$ and $\hat{\boldsymbol{x}} \in \mathbb{R}^{1\times 49152}$ is the new row in $\mathcal{X}_{[4]}$. The sampling techniques in Section 4 can be used to compute approximate solutions to (133) and (134) efficiently. The feature vectors $\boldsymbol{a}^*$ and $\boldsymbol{g}^*$ can now be used to classify the new image.

D.5. Demonstration of Improved Complexity for the Tensor Ring Decomposition

We construct a synthetic 10-way tensor that demonstrates the improved sampling and computational complexity of our proposed TR-ALS-ES over TR-ALS-Sampled. It is constructed via (8) from core tensors $\mathcal{G}^{(n)} \in \mathbb{R}^{3\times 6\times 3}$ for $n \in [10]$ with $\mathcal{G}^{(n)}(1,1,1) = 3$ and all other entries zero. Additionally, i.i.d. Gaussian noise with standard deviation 0.01 is added to all entries of the tensor. Both methods are run for 20 iterations with target ranks $(3,3,\ldots,3)$ and are initialized using a variant of the randomized range finding approach proposed by Larsen & Kolda (2020), Appendix F, adapted to the tensor ring decomposition. TR-ALS-Sampled fails even when as many as *half* (i.e., $J = 6^9/2 \approx 5.0$e+6) of all rows are sampled, taking 966 seconds. By contrast, our TR-ALS-ES only requires a recursive sketch size of $J_1 =$ 1e+4 and $J_2 =$ 1e+3 samples to get an accurate solution, taking 41 seconds. Our method improves the sampling complexity and compute time by 3 and 1 orders of magnitude, respectively.

D.6. Preliminary Results From Experiments on the Tensor Train Decomposition

In this section we provide some preliminary results from experiments on the tensor train (TT) decomposition. We do these experiments by running the different tensor ring decomposition algorithms with $R_0 = R_N = 1$ which makes the resulting decomposition a TT. We refer to the methods by the same names as the tensor ring decomposition methods, but with "TR" replaced by "TT."

Table 8. KL-divergence (lower is better) of the approximated sampling distribution from the exact one for a TT-ALS least squares problem with target TT-ranks $R_n = R$ for $1 \le n \le 5$. The TT-ALS least squares problem is identical to the TR-ALS problem in (11) but with the restriction $R_0 = R_N = 1$.

Method	$R = 3$	$R = 5$
TT-ALS-Sampled	0.4843	0.2469
TT-ALS-ES ($J_1 =$ 1e+4)	0.0004	0.0007
TT-ALS-ES ($J_1 =$ 1e+3)	0.0087	0.0065
TT-ALS-ES ($J_1 =$ 1e+2)	0.0425	0.0845

First, we repeat the experiments in Section 5.1 for the TT decomposition. All settings are the same as for the tensor ring decomposition except that $R_0 = R_6 = 1$. The results are shown in Table 8. The discrepancy between the approximate leverage score sampling distribution that TT-ALS-Sampled samples from and the exact one is greater than it is for the tensor ring decomposition (compare with Table 4). The discrepancy of TT-ALS-ES is similar to that of TR-ALS-ES for $J_1 =$ 1e+4 and $J_1 =$ 1e+3 and smaller for $J_1 =$ 1e+2. Since we are solving for the 6th core tensor out of 6, the theoretical bound on

Table 9. Run time, decomposition error and classification accuracy when using the TT decomposition for feature extraction.

Method	Time (s)	Error	Accuracy (%)
TT-ALS	9440.1	0.44	95.2
TT-ALS-Sampled	12.3	0.44	94.1
TT-ALS-ES (our)	31.4	0.44	94.2

J_1 in (30) becomes $J_1 \gtrsim N(R_5R_6)^2/\delta = NR^2/\delta$ since $R_6 = 1$. For the tensor ring decomposition with all $R_n = R$, the same bound is $J_1 \gtrsim NR^4/\delta$. A smaller discrepancy is therefore expected for TT-ALS-ES than for TR-ALS-ES.

Next, we repeat the feature extraction experiment in Section 5.2 for the TT decomposition. In addition to the restriction $R_0 = R_1 = 1$ we also increase the number of samples from 1000 to 3000 for both TT-ALS-Sampled and TT-ALS-ES since using fewer than 3000 samples yields poor results for both methods. We also add a small Tikhonov regularization term (with regularization constant 10^{-2}) in all least squares solves for both randomized methods in order to avoid numerical issues that otherwise appear for both. The TT-ranks are $R_n = 5$ for $1 \leq n \leq 3$. The results are reported in Table 9. The run time for TT-ALS is slightly faster than that of TR-ALS which is expected since the TT has fewer parameters than the tensor ring (compare with Table 5). The two randomized algorithms are a bit slower than they are for the tensor ring decomposition due to the larger number of samples being drawn. The decomposition error is higher and the classification accuracy lower than they are for the tensor ring decomposition. This is also expected since the TT decomposition has fewer parameters.