A Randomized Tensor Singular Value Decomposition based on the t-product

4 authors, including:

Arvind Saibaba
North Carolina State University
40 PUBLICATIONS   **265** CITATIONS

SEE PROFILE

Shuchin Aeron
Tufts University
92 PUBLICATIONS   **1,225** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Optimal Design of Experiments View project

Nonlinear Model Reduction View project

A Randomized Tensor Singular Value Decomposition based on the t-product*

Jiani Zhang†   Arvind K. Saibaba‡   Misha E. Kilmer†   Shuchin Aeron§

September 23, 2016

Abstract

The tensor Singular Value Decomposition (t-SVD) for third order tensors that was proposed by Kilmer and Martin [30] has been applied successfully in many fields, such as computed tomography, facial recognition, and video completion. In this paper, we propose a method that extends a well-known randomized matrix method to the t-SVD. This method can produce a factorization with similar properties to the t-SVD, but is more computationally efficient on very large datasets. We present details of the algorithm, theoretical results, and provide numerical results that show the promise of our approach for compressing and analyzing datasets. We also present an improved analysis of the randomized subspace iteration for matrices, which may be of independent interest to the scientific community.

Keywords: truncated SVD, randomized SVD, singular value decomposition, tensor, t-product

1   Introduction

In this era of "big data," it is not uncommon for the size of a matrix operator, or a dataset, to reach the scale of petabytes or even exabytes. By 2013, for example, Facebook was already claiming to use 1.5 petabytes to store about 10 billion photos, and Netflix claimed to use 3.14 petabytes to store available shows and movies [44]. As another example, the size of the matrix operator in quantum chromodynamics is on the order of several millions, or even billions [17]. On the one hand, there still seems to be a push to obtain ever more information by collecting more data since the storage capability exists, and for modeling very fine scale phenomena. On the other hand, current data analysis and scientific computing methods are continually challenged by the expanding sizes of the models and datasets.

Almost all of the methods in data analysis and scientific computing rely on matrix algorithms [46]. In particular, the low-rank matrix approximation,

$$\mathbf{A}_{m \times n} \approx \mathbf{B}_{m \times k}\mathbf{C}_{k \times n}, \tag{1}$$

where $k < \min\{m, n\}$, is used often, because it allows us to store or analyze the matrix $\mathbf{A}$ by the factor matrices $\mathbf{B}$ and $\mathbf{C}$ instead of the full matrix, which is more efficient when $k \ll \min\{m, n\}$.

†Department of Mathematics, Tufts University, Medford, MA 02155 (jiani.zhang@tufts.edu, misha.kilmer@tufts.edu).

‡Department of Mathematics, North Carolina State University, Raleigh, NC 27695 (asaibab@ncsu.edu).

§Department of Electrical and Computer Engineering, Tufts University, Medford, MA 02155 (shuchin@ece.tufts.edu).

Moreover, these smaller matrices have often been shown to provide specific structure that help analyze the data with better results, see [5, 8, 43].

It is well known that truncating the matrix singular value decomposition to k terms provides the optimal rank-k approximation to a matrix in both the matrix 2-norm and Frobenius norm, and the algorithms for computing the approximation are numerically robust. Therefore, it is not surprising that the truncated SVD has been proposed for use in many applications, included but not limited to image processing [35, 22], statistics [25, 40] and Partial Differential Equations (PDEs) [12]. However, the cost of accurately computing the truncated matrix SVD can be prohibitively expensive, making it unsuitable for very large scale applications [38].

Therefore, much work has been devoted to generating low-rank approximations which have similar rank-revealing properties to the SVD but which are cheaper to compute. As a trade-off, one gives up the optimality property that is the signature feature of the SVD. In recent years, much work has been devoted to the development of randomized algorithms for computing low-rank matrix approximations. They are particularly appealing because although the cannot give the optimal low-rank approximation, they can be shown to give nearly optimal results.

Randomized algorithms have been recently developed for accurate low-rank representations, see [16, 3, 10, 2, 4, 21, 20, 9] and several others. Randomized matrix methods are powerful because they are numerically robust, computationally efficient, and suitable for implementation on a variety of architectures, including high-performance computing. They usually come in two different flavors – based on random sampling of columns and rows of the matrix, or by random projection onto a lower dimensional subspace. More details of these two different approaches can be found in the review paper by [33].

All the previous work referenced above involves generating near optimal low-rank **matrix** approximations from randomized techniques. However, many data sets and matrix operators are inherently multidimensional in nature. Consider, for example, a collection of hyperspectral images. One can choose to scan the image at each wavelength as a vector, resulting in a matrix representation of all the hyperspectral data. But one might also store each 2D image as a slice of a 3-way array, resulting in a third order tensor representation of the data. As another example, each frame of a color image is technically a 3D array, and so the time sequence of a video can be stored as a 4D array with time as the last index – the resulting data structure is called a fourth order tensor. The question of interest to us in this paper is, if we choose to keep the multi-way structure inherent in the data, can we then generalize the concept of a best (in some norm) "low-rank" approximation to tensor data, and if so, how do we move from randomized low-rank matrix techniques to randomized tensor factorizations.

First, one must decide on method of tensor decomposition, and with it, the notion of best "low-rank". The well-known CP decomposition, originally proposed by Hitchcock in 1927 [26], is a decomposition as a sum of multiway outer products of vectors. Tensor rank is then defined in terms of the minimal sum of these outer products necessary to construct the tensor [31]. The difficulty is a best rank-k approximation need not exist without extra assumptions, and computing the rank-k approximation is also highly non-trivial even when it does exist. The Tucker decomposition [42], developed by Tucker in 1963, is an alternative decomposition. For a third-order tensor, the Tucker3 decomposition requires 3 factor matrices and a core tensor. A Tucker3 factorization always exists, and can be generated such that factor matrices can have orthonormal columns – the HOSVD [11] is such a decomposition. One can specify the rank of the factor matrices, and thus obtain a so-called best rank-(R_1, R_2, R_3) Tucker3 approximation. However, unlike the matrix case, this best approximation cannot be obtained by truncating the full HOSVD.

A more recent alternative approach to factoring tensors was introduced by Kilmer and Martin in 2011 [30]. In their work, the authors present the concept of a tensor-tensor product with suitable algebraic structure such that classical matrix-like factorizations are possible. In particular, they give the definition of the tensor SVD (t-SVD) over this new product, and show that truncating that expansion does give a compressed result that is optimal in the Frobenius norm.

Applications of all three types of these tensor-based decompositions can be found in the literature: see for example [45, 39, 14, 48, 23, 15, 36]. Though the decompositions are different, the common theme among the results is evidence that tensor-based decompositions of the data/operators

provide considerable improvement over the matrix-based counterparts. However, all these tensor-based decompositions are deterministic, so it is natural to explore randomization of these tensor decompositions as well.

To the best of our knowledge, the authors in [13] appear to have pioneered the generalization of random sampling methods to tensors. Specifically, they extended random sampling methods to the Tucker decomposition, and provided a guide to the theoretical analysis for the tensor-based decomposition via random sampling methods. In [41], the authors provide numerical examples of Tucker decomposition with the random sampling method. A literature search also reveals attempts to extend the random sampling approach to tensors-based on CP decomposition and Tucker decomposition, [6, 37].

In this paper, we extend a well-known matrix-based random projection method, the randomized SVD (r-SVD) [21], to third-order tensors through use of the algebra induced by the t-product and the t-SVD [30]. The motivation for focusing our efforts on the randomization of the t-SVD is the theoretical and computational advantages provided by the t-product, as well as the use of the t-SVD in applications. For example, as mentioned above, a best k-term expansion can be obtained from truncation of the t-SVD. Further, the t-SVD computations are readily parallelizable. Under the t-product, there are well defined concepts of orthogonality, identity and orthogonal projections, QR factorizations, and the like [29, 18, 34]. Moreover, in some applications, such as compression and facial recognition, the t-SVD has been shown to have superior compression characteristics [24] relative to the Tucker decomposition. When they use some storage, the t-SVD has better performance in term of recognition rate.

Contributions   We develop randomized algorithms for low-rank decompositions of tensors, based on the t-product. The first algorithm applies the randomized SVD to the frontal slices of the tensor (in the Fourier domain), where as the second algorithm applies the randomized power method to the same slices. Efficient implementations of the above algorithms are also provided in the spatial, as well as Fourier domains. We develop a framework for error analysis and derive expressions for expected behavior of the error in the low-rank representation, as well as probabilistic bounds for deviation from expectation. Our analysis for randomized power method is novel even for the matrix case. Application to facial recognition, including a parallel implementation, underscores the benefits of the proposed methods.

This paper is organized as follows. In section 2, we review some relevant mathematical concepts including the matrix r-SVD, basic definitions and theorems of tensors, and the t-SVD based on the t-product. In section 3, we give the basic randomized t-SVD (rt-SVD) method and extend this to the rt-SVD with subspace iteration. There, we also provide the analysis of error expectations. In section 4, we compare the errors of our algorithms with theoretical minimal errors on a real dataset, apply the algorithms in the application of facial recognition, and compute them in parallel on a cluster to show the improvement of computation efficiency. Conclusions and future work are provided in section 5. Where noted in the body of the paper, some of the proof details are provided in the Appendix A.

2   Preliminaries

In this section, we introduce some definitions and algorithms which are used throughout the paper. We begin by noting that boldface lowercase letters indicate vectors, e.g. $\mathbf{a}$. Boldface uppercase letters indicate matrices, e.g. $\mathbf{A}$. Boldface Euler script letters indicate tensors, e.g. $\mathcal{A}$, and unless otherwise specified, the tensors are third order.

We will assume that $\mathbf{A}$ is of rank r, and $\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^{\mathrm{H}}$ is the singular value decomposition of $\mathbf{A}$, with $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_r > 0$.

2.1 The r-SVD

The randomized Singular Value Decomposition (referred to as r-SVD), was proposed in a series of papers published over the last decade (see e.g., [32, 47]) and popularized by the review paper [21]. The first step in computing the r-SVD is generating several Gaussian random vectors $\boldsymbol{W} \in \mathbb{R}^{n \times (k+p)}$ that are, with high probability, linearly independent. Here k is the desired target truncation term of the approximation, and p is a non-negative integer oversampling parameter. The matrix $\boldsymbol{Y} := \boldsymbol{AW} \in \mathbb{C}^{m \times (k+p)}$ thus contains random linear combinations of the columns of $\boldsymbol{A}$.

A thin QR of $\boldsymbol{Y}$ is computed, so that range($\boldsymbol{Y}$) = range($\boldsymbol{Q}$).

The idea is if $\mathbf{A}$ has rapidly decaying singular values, so that the dominant part of the range of $\mathbf{A}$ is marked by the first k or so left singular vectors, i.e., $\mathbf{A} \approx \mathbf{QQ^H A}$.

Thus, one computes $\mathbf{B} := \mathbf{Q^H A}$ followed by the compact SVD of $\mathbf{B}$, $\mathbf{B} = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^H$. The estimated desired singular values of $\mathbf{A}$ are the diagonals of $\tilde{\mathbf{S}}$, while $\mathbf{Q}\tilde{\mathbf{U}}$ gives the estimated right singular vectors of $\mathbf{A}$. Algorithm 1 summarizes the procedure described above.

Algorithm 1: r-SVD method [21]

Input : $\mathbf{A} \in \mathbb{C}^{m \times n}$, target truncation term k, and oversampling parameter p
Output: $\mathbf{U}_k \in \mathbb{C}^{n \times k}$, $\mathbf{S}_k \in \mathbb{C}^{k \times k}$, and $\mathbf{V}_k \in \mathbb{C}^{n \times k}$

1 Generate a Gaussian random matrix $\mathbf{W} \in \mathbb{R}^{n \times (k+p)}$;
2 Form a matrix $\mathbf{Y} = \mathbf{AW}$;
3 Construct matrix $\mathbf{Q} \in \mathbb{C}^{n \times (k+p)}$ which is orthogonal column basis for $\mathbf{Y}$;
4 Form $\mathbf{B} \in \mathbb{C}^{(k+p) \times n}$, $\mathbf{B} = \mathbf{Q^* A}$;
5 Compute $\mathbf{B} = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^H$,;
6 Set $\mathbf{U} = \tilde{\mathbf{U}}(:, 1{:}k)$, $\mathbf{S}_k = \tilde{\mathbf{S}}(1{:}k, 1{:}k)$, $\mathbf{V}_k = \tilde{\mathbf{V}}(:, 1{:}k)$;
7 Form $\mathbf{U}_k = \mathbf{Q}_k \mathbf{U}$.

When $\mathbf{A}$ is dense and of size $n \times n$, this algorithm can take $\mathcal{O}(kn^2)$ flops. For more details, see [47]. The expected error in the low-rank approximation measured using the Frobenius norm can be bounded, as the result below shows. This result was first proved in [21, Theorem 10.5], but is stated here in a slightly different form. The original result derived the error bound for $\mathbb{E}\left\|\mathbf{A} - \mathbf{QQ^H A}\right\|_{\mathrm{F}}$, whereas, in the next section we require the error bound for $\mathbb{E}\left\|\mathbf{A} - \mathbf{QQ^H A}\right\|_{\mathrm{F}}^2$.

Theorem 1. *Given a matrix* $\mathbf{A} \in \mathbb{C}^{m \times n}$ *and a Gaussian random matrix* $\mathbf{W} \in \mathbb{R}^{n \times (k+p)}$, *let* $p \geq 2$ *be a pre-specified integer. Suppose that* $\boldsymbol{Q}$ *is computed as in Algorithm 1, then the expected approximation error is*

$$\mathbb{E}\left\|\mathbf{A} - \mathbf{QQ^H A}\right\|_F^2 \leq \left(1 + \frac{k}{p-1}\right)\left(\sum_{j=k+1}^{\min\{m,n\}} \sigma_j^2\right) \tag{2}$$

where σ_j *is the* j^{th} *singular value of* $\mathbf{A}$.

Proof. The proof follows readily from [21, Theorem 10.5]. $\square$

From the inequality (2), the value of $\mathbb{E}\|\mathbf{A} - \mathbf{QQ^H A}\|_{\mathrm{F}}^2$ depends on $\sum_{j>k} \sigma_j^2$. When the singular values of $\mathbf{A}$ decay gradually, $\sum_{j>k} \sigma_j^2$ can be large, and therefore the low-rank approximation as computed above may not be sufficiently accurate. In this situation, Algorithm 2, which is based on

subspace iteration, may be preferred.

Algorithm 2: r-SVD method with subspace iteration [21]

Input : $\mathbf{A} \in \mathbb{C}^{m \times n}$, target truncation term k, a parameter q, and an oversampling parameter p

Output: An orthogonal column basis $\mathbf{Q}$ of $\mathbf{Y}$

1 Generate a Gaussian random matrix $\mathbf{W} \in \mathbb{R}^{n \times (k+p)}$;
2 Form a matrix $\mathbf{Y_0} = \mathbf{AW}$ and compute the QR factorization of $\mathbf{Y}_0 = \mathbf{Q}_0 \mathbf{R}_0$;
3 **for** $i \leftarrow 1$ **to** q **do**
4 $\quad$ Form $\tilde{\mathbf{Y}}_i = \mathbf{A}^{\mathrm{H}} \mathbf{Q}_{i-1}$ and compute the QR factorization of $\tilde{\mathbf{Y}}_i = \tilde{\mathbf{Q}}_i \tilde{\mathbf{R}}_i$;
5 $\quad$ Form $\mathbf{Y}_i = \mathbf{A} \tilde{\mathbf{Q}}_i$ and compute the QR factorization of $\mathbf{Y}_i = \mathbf{Q}_i \mathbf{R}_i$;
6 **end**
7 Form a matrix $\mathbf{Q} = \mathbf{Q}_q$;

The error analysis for Algorithm 2 is developed in [21] for the spectral norm, but no analysis was presented in the Frobenius norm. We present the following result that characterizes the error due to Algorithm 2 in the Frobenius norm. We assume that k is the target truncation term, and define the singular value gap $\tau_k \equiv \frac{\sigma_{k+1}}{\sigma_k}$. We further assume that $\tau_k \ll 1$, i.e., there is a gap between singular values k and $k+1$.

Theorem 2 (Average Frobenius error for Algorithm 2). *Let* $\mathbf{A} \in \mathbb{C}^{m \times n}$ *and let* $\mathbf{W} \in \mathbb{R}^{n \times (k+p)}$ *be a Gaussian random matrix with* $p \geq 2$ *being the oversampling parameter. Suppose* $\mathbf{Q}$ *is obtained from Algorithm 2, then*

$$\mathbb{E} \left\| \mathbf{A} - \mathbf{Q}\mathbf{Q}^{\mathrm{H}}\mathbf{A} \right\|_F^2 \leq \left(1 + \frac{k}{p-1} \tau_k^{4q} \right) \left(\sum_{j=k+1}^{\min\{m,n\}} \sigma_j^2 \right),$$

where k *is a target truncation term,* q *is the number of iterations,* σ_j *is the* j^{th} *singular value of* $\mathbf{A}$, *and* $\tau_k = \sigma_{k+1}/\sigma_k \ll 1$ *is the singular value gap.*

Proof. See Appendix A. $\qquad \square$

The error due to the randomized subspace iteration is similar to Theorem 2, except for the term τ_k^{4q}. As the number of subspace iterations q increases, the effect of the residual term $k\tau_k^{4q}/(p-1)$ decreases, and the subspace iteration achieves the optimal error of the SVD. Also note that for $q = 0$, we exactly obtain the result in Theorem 2. A similar result was presented in [19, Theorem 5.7], but our analysis is sharper, see discussion in Remark 1.
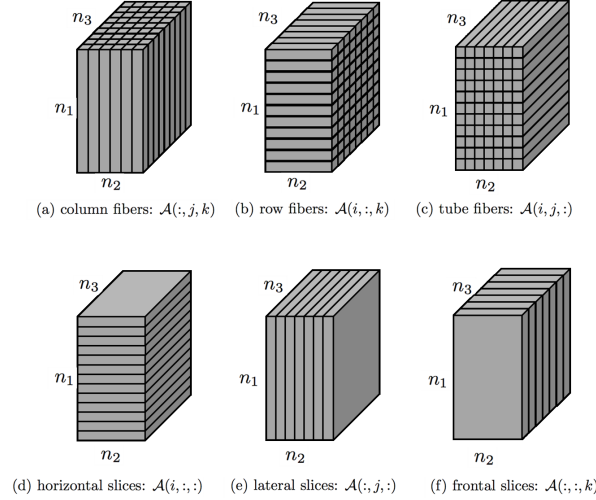
2.2 Tensors

A tensor is a multi-dimensional array, and the order of the tensor is the number of dimensions of this array. In this paper, we focus on the third order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. Each entry of the tensor $\mathcal{A}$ is denoted by Matlab indexing notation, i.e., $\mathcal{A}(i, j, k)$.

A fiber of tensor $\mathcal{A}$ is a one-dimensional array defined by fixing two indices. $\mathcal{A}(:, j, k)$ is the $(j, k)^{th}$ column fiber, $\mathcal{A}(i, :, k)$ is the $(i, k)^{th}$ row fiber, and $\mathcal{A}(i, j, :)$ is the $(j, k)^{th}$ tube fiber. A slice of tensor $\mathcal{A}$ is a two-dimensional array defined by fixing one index. $\mathcal{A}(i, :, :)$ is the ith horizontal slice, $\mathcal{A}(:, j, :)$ is the jth lateral slice, and $\mathcal{A}(:, :, k)$ is the kth frontal slice. For convenience, $\mathcal{A}(:, :, k)$ is written as $\mathcal{A}^{(k)}$. A third order tensor $\mathcal{A}$ can be seen as an $n_1 \times n_2$ array of tube fibers, each of size $1 \times 1 \times n_3$. The t-product of two tube fibers is defined as their the circular convolution, so the t-product between two tensors can be defined as in Definition 1. Next, we review several definitions from [30] that will be necessary for the rest of this paper.

Definition 1 (t-product). *Let* $\mathcal{A}$ *be an* $n_1 \times n_2 \times n_3$ *tensor and* $\mathcal{B}$ *be an* $n_2 \times n_4 \times n_3$ *tensor. The t-product of* $\mathcal{A}$ *and* $\mathcal{B}$, $\mathcal{C} = \mathcal{A} * \mathcal{B}$, *is an* $n_1 \times n_4 \times n_3$ *tensor*

$$\mathcal{C}(i, j, :) = \sum_{k=1}^{n_2} \mathcal{A}(i, k, :) * \mathcal{B}(k, j, :) = \sum_{k=1}^{n_2} \mathcal{A}(i, k, :) \circ \mathcal{B}(k, j, :)$$

(a) column fibers: $\mathcal{A}(:,j,k)$ (b) row fibers: $\mathcal{A}(i,:,k)$ (c) tube fibers: $\mathcal{A}(i,j,:)$

(d) horizontal slices: $\mathcal{A}(i,:,:)$ (e) lateral slices: $\mathcal{A}(:,j,:)$ (f) frontal slices: $\mathcal{A}(:,:,k)$

Figure 2.2.1: Fibers and slices of an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$

.

where the notation $\circ$ denotes the circular convolution.

Because the circular convolution of two tube fibers can be computed by discrete Fourier transform, the t-product can be alternatively computed in the Fourier domain[1], as shown in Algorithm 3.

Algorithm 3: t-product computation in Fourier domain [30]

Input : $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$
Output: an $n_1 \times n_4 \times n_3$ tensor $\mathcal{C}$, $\mathcal{C} = \mathcal{A} * \mathcal{B}$

1 $\hat{\mathcal{A}} \leftarrow \mathtt{fft}(\mathcal{A}, [\,], 3)$;
2 $\hat{\mathcal{B}} \leftarrow \mathtt{fft}(\mathcal{B}, [\,], 3)$;
3 **for** $i \leftarrow 1$ **to** n_3 **do**
4 $\quad | \quad \hat{\mathcal{C}}^{(i)} = \hat{\mathcal{A}}^{(i)} \hat{\mathcal{B}}^{(i)}$;
5 **end**
6 $\mathcal{C} \leftarrow \mathtt{ifft}(\hat{\mathcal{C}}, [\,], 3)$

Definition 2 (Identity tensor). *The $n_1 \times n_2 \times n_3$ identity tensor $\mathcal{I}$ is the tensor whose first frontal slice is the $n_1 \times n_2$ identity matrix, and whose other frontal slices are all zeros.*

Definition 3 (Transpose). *If $\mathcal{A}$ is an $n_1 \times n_2 \times n_3$ tensor, then $\mathcal{A}^{\mathrm{T}}$ is an $n_2 \times n_1 \times n_3$ tensor obtained by transposing each of the frontal slices and then reversing the order of transposed frontal slices 2 through n_3, see Fig. 2.2.2.*

Definition 4 (Orthogonality). *An $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$ is called orthogonal if $n_1 = n_2$, if the t-product of $\mathcal{A}^{\mathrm{T}}$ and $\mathcal{A}$ is equal to the identity tensor, i.e.,*

$$\mathcal{A}^{\mathrm{T}} * \mathcal{A} = \mathcal{I}_{n_2 n_2 n_3}.$$
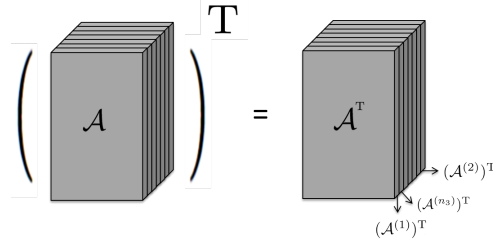
If the above equation holds but $n_1 > n_2$, then the tensor is said to be partially orthogonal.

Definition 5 (f-Diagonal). *An $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$ is called f-diagonal, if each frontal face of $\mathcal{A}$ is diagonal.*

---

[1]All quantities in the spatial domain are real and all quantities in the Fourier domain may be complex.

Figure 2.2.2: Transpose of an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$

.

Definition 6 (t-QR factorization). *Given an $n_1 \times n_2 \times n_3$ tensor* **A**, *the t-QR factorization of $\mathcal{A}$ is*

$$\mathcal{A} = \mathcal{Q} * \mathcal{R}$$

where $\mathcal{Q}$ is partially orthogonal.

We also introduce for the first time the notion of Gaussian random tensors. Our definition for Gaussian random tensors is motivated by the need to generate as few random numbers, or samples, as possible, while still being able to use the conclusions of literature on Gaussian random matrices.

Definition 7 (Gaussian random tensor). *An $n_1 \times n_2 \times n_3$ tensor $\mathcal{W}$ is called a Gaussian random tensor, if the elements of $\mathcal{W}^{(1)}$ satisfy the standard normal distribution, and other frontal slices are all zeros.*

The Fourier transform of $\mathcal{W}$ along the 3rd dimension is denoted as $\hat{\mathcal{W}}$ such that every frontal slice is an identical copy of the first slice $\mathcal{W}^{(1)}$, see Figure 6.
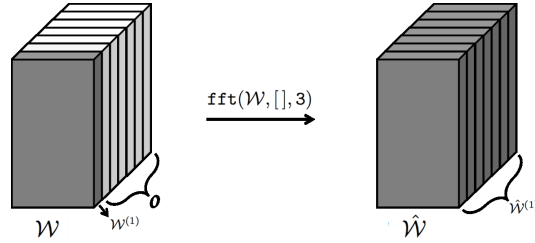
Figure 2.2.3: A third order Gaussian random tensor and its Fourier transform.

2.3 t-SVD

We now present the t-SVD and truncated t-SVD, which builds on the operations of tensors introduced in Section 2.2. These were first developed in [30].

Definition 8. *[30] Let $\mathcal{A}$ be an $n_1 \times n_2 \times n_3$ tensor. The t-SVD of $\mathcal{A}$ is*

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^{\text{T}}$$

where $\mathcal{U} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$, $\mathcal{V} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ are partially orthogonal tensors, and $\mathcal{S}_k \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is a f-diagonal tensor.

Definition 9. *[30] Given a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, define the truncated t-SVD of $\mathcal{A}$ as*

$$\mathcal{A}_k = \mathcal{U}_k * \mathcal{S}_k * \mathcal{V}_k^{\text{T}}$$

where k is a target truncation term, $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$ are partially orthogonal, and $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$ is a f-diagonal tensor.

Figure 2.3.1: The t-SVD of an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$.

The optimality of the error in the truncated t-SVD is presented below, which is a generalization of the well-known result of optimality of the truncated SVD [27, Section 7.4.2]. Note, however, that truncation of the t-SVD to k terms is not the same as computing a rank-k tensor approximation (i.e., a tensor approximation as the sum of k outer products of vectors), 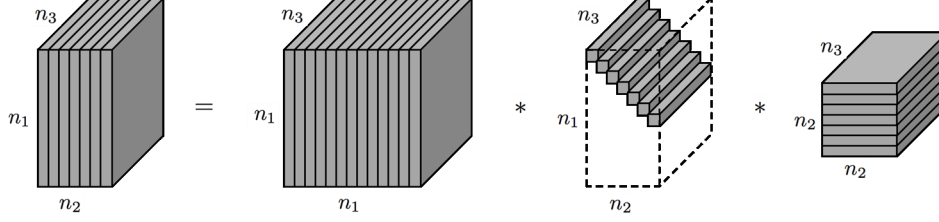nor is it equivalent to a best rank-(R_1, R_2, R_3) Tucker3 approximation. What we show is that truncating the t-SVD gives the best "tubal-rank k" approximation [29].

Here, and henceforth, we will use the short-hand notation $\sum_{j>k}$ to represent $\sum_{j=k+1}^{\min\{n_1,n_2\}}$ for clarity.

Theorem 3. *[30] Given a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{A}_k = \arg\min_{\tilde{\mathcal{A}} \in \mathcal{M}} \|\mathcal{A} - \tilde{\mathcal{A}}\|_F$, where $\mathcal{M} = \{\mathcal{C} = \mathcal{X} * \mathcal{Y} \mid \mathcal{X} \in \mathbb{R}^{n_1 \times k \times n_3}, \mathcal{Y} \in \mathbb{R}^{k \times n_2 \times n_3}\}$. Therefore, $\|\mathcal{A} - \mathcal{A}_k\|_F$ is the theoretical minimal error, given by*

$$\|\mathcal{A} - \mathcal{A}_k\|_F = \left(\frac{1}{n_3} \sum_{i=1}^{n_3} \sum_{j>k} (\hat{\sigma}_j^{(i)})^2 \right)^{1/2}, \tag{3}$$

where $\hat{\sigma}_j^{(i)} \equiv \hat{\mathcal{S}}(j,j,i)$ is the jth singular value corresponding to the ith frontal face (in the Fourier domain).

The relevance of the Fourier transform is perhaps not immediately obvious, until one examines how the truncated t-SVD is actually computed. The procedure is given in Algorithm 4.

Algorithm 4: k-term truncated t-SVD [30]

Input : $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and target truncation term k
Output: $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$, and $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$

1 $\hat{\mathcal{A}} \leftarrow \texttt{fft}(\mathcal{A}, [\,], 3)$;
2 **for** $i \leftarrow 1$ **to** n_3 **do**
3 $\quad$ $[\mathbf{U}, \mathbf{S}, \mathbf{V}] = svd(\mathcal{A}^{(i)})$;
4 $\quad$ Form $\mathbf{U}_k$, $\mathbf{S}_k$, and $\mathbf{V}_k$ by truncating $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$ with target truncation term k;
5 $\quad$ $\hat{\mathcal{U}}_k^{(i)} = \mathbf{U}_k$; $\hat{\mathcal{S}}_k^{(i)} = \mathbf{S}_k$; $\hat{\mathcal{V}}_k^{(i)} = \mathbf{V}_k$;
6 **end**
7 $\mathcal{U}_k \leftarrow \texttt{ifft}(\hat{\mathcal{U}}_k, [\,], 3)$; $\mathcal{S}_k \leftarrow \texttt{ifft}(\hat{\mathcal{S}}_k, [\,], 3)$; $\mathcal{V}_k \leftarrow \texttt{ifft}(\hat{\mathcal{V}}_k, [\,], 3)$;

For a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, computing a k-term truncated t-SVD using Algorithm 4 takes $\mathcal{O}(n_1 n_2 n_3 k)$ flops. For a matrix $\mathbf{A} \in \mathbb{R}^{n_1 n_2 \times n_3}$, computing a k-term truncated SVD takes $\mathcal{O}(n_1 n_2 n_3 k)$ flops as well. However, Algorithm 4 can be computed in parallel over the frontal slices on a cluster, whereas typical algorithms used for the truncated SVD of a matrix cannot be computed in parallel.

3 Randomized Tensor SVD

In this section, we propose the rt-SVD method, which extends the matrix r-SVD method to the t-SVD. The goal of the rt-SVD (randomized tensor SVD) method is to find a good approximate factorization of tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{U}_k * \mathcal{S}_k * \mathcal{V}_k^{\mathrm{T}}$. There are two main steps which is summarized

in Algorithm 5. The first step is to find a tensor $\mathcal{Q}$ such that

$$\mathcal{A} \approx \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A},$$

in a manner that will be made precise, and the second step is to connect this low-tubal-rank representation to a rt-SVD factorization.

Algorithm 5: rt-SVD, spatial domain version

Input : $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, target truncation term k, and oversampling parameter p
Output: $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$, and $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$

1 Generate a Gaussian random tensor $\mathcal{W} \in \mathbb{R}^{n_2 \times (k+p) \times n_3}$;
2 Form a random projection of tensor $\mathcal{A}$ as $\mathcal{Y} = \mathcal{A} * \mathcal{W}$;
3 Construct the tensor $\mathcal{Q}$ by using t-QR factorization;
4 Form a tensor $\mathcal{B} = \mathcal{Q}^{\mathrm{T}} * \mathcal{A}$, whose size is $(k + p) \times n_2 \times n_3$;
5 Compute t-SVD of $\mathcal{B}$, truncate it with target truncation term k, and obtain $\mathcal{U}$, $\mathcal{S}_k$, and $\mathcal{V}_k$;
6 Form the rt-SVD of $\mathcal{A}$, $\mathcal{A} \approx (\mathcal{Q} * \mathcal{U}) * \mathcal{S}_k * \mathcal{V}_k^{\mathrm{T}} = \mathcal{U}_k * \mathcal{S}_k * \mathcal{V}_k^{\mathrm{T}}$.

For the convenience of error analysis , we present an implementation of Algorithm 5 in the Fourier domain. This allows us to apply results from the matrix r-SVD independently to each frontal slice.

Algorithm 6: rt-SVD, Fourier domain version

Input : $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, target truncation term k, and parameter p
Output: $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$, and $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$

1 Generate a Gaussian random tensor $\mathcal{W} \in \mathbb{R}^{n_2 \times (k+p) \times n_3}$;
2 $\hat{\mathcal{A}} \leftarrow \mathtt{fft}(\mathcal{A}, [\,], \mathtt{3})$ and $\hat{\mathcal{W}} \leftarrow \mathtt{fft}(\mathcal{W}, [\,], \mathtt{3})$;
3 **for** $i \leftarrow 1$ **to** n_3 **do**
4 $\quad \hat{\mathcal{Y}}^{(i)} = \hat{\mathcal{A}}^{(i)} \hat{\mathcal{W}}^{(i)}$;
5 $\quad [\hat{\mathcal{Q}}^{(i)}, \hat{\mathcal{R}}^{(i)}] = \mathtt{qr}(\hat{\mathcal{Y}}^{(\mathtt{i})}, \mathtt{0})$;
6 $\quad \hat{\mathcal{B}}^{(i)} = (\hat{\mathcal{Q}}^{(i)})^{\mathrm{H}} \hat{\mathcal{A}}^{(i)2}$;
7 $\quad [\hat{\mathcal{U}}^{(i)}, \hat{\mathcal{S}}^{(i)}, \hat{\mathcal{V}}^{(i)}] = \mathtt{svd}(\hat{\mathcal{B}}^{(\mathtt{i})})$;
8 $\quad \hat{\mathcal{U}}_k^{(i)} = \hat{\mathcal{Q}}^{(i)} \hat{\mathcal{U}}^{(i)}$; $\hat{\mathcal{S}}_k^{(i)} = \hat{\mathcal{S}}^{(i)}(1:k, 1:k)$; $\hat{\mathcal{V}}_k^{(i)} = \hat{\mathcal{V}}^{(i)}(:, 1:k)$.
9 **end**
10 $\mathcal{U}_k \leftarrow \mathtt{ifft}(\hat{\mathcal{U}}_{\mathtt{k}}, [\,], \mathtt{3})$; $\mathcal{S}_k \leftarrow \mathtt{ifft}(\hat{\mathcal{S}}_{\mathtt{k}}, [\,], \mathtt{3})$; $\mathcal{V}_k \leftarrow \mathtt{ifft}(\hat{\mathcal{V}}_{\mathtt{k}}, [\,], \mathtt{3})$.

We now present a theorem that gives the expected error of $\|\mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A}\|_{\mathrm{F}}$ where the tensor $\mathcal{Q}$ is computed using Algorithm 5.

Theorem 4. *Given an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$ and an $n_2 \times (k + p) \times n_3$ Gaussian random tensor $\mathcal{W}$, if $\mathcal{Q}$ is obtained from t-QR of $\mathcal{Y} = \mathcal{A} * \mathcal{W}$, then*

$$\mathbb{E}\|\mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A}\|_{\mathrm{F}} \le \sqrt{1 + \frac{k}{p-1}} \left(\frac{1}{n_3} \sum_{i=1}^{n_3} \sum_{j>k} (\hat{\sigma}_j^{(i)})^2 \right)^{1/2}.$$

where k is a target truncation term, $p \ge 2$ is the oversampling parameter, and $\hat{\sigma}_j^{(i)}$ is the i_{th} component of $\mathtt{fft}(\mathcal{S}(\mathtt{j}, \mathtt{j}, :), [\,], \mathtt{3})$.

Proof. See Appendix B. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

---

[2] Since slices of $\hat{\mathcal{A}}$ can be complex in general, we use the notation superscript H instead of superscript T here.

Theorem 4 is important because it shows that, in expectation, the error in the rt-SVD algorithm is within a factor $\sqrt{1 + \dfrac{k}{p-1}}$ of the optimal result in Theorem 3. Note that this is the same optimality factor that one obtains in the matrix case, see Theorem 2.

Theorem 4 also shows how the error of the low-tubal-rank approximation $\mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A}$ relies on the decay of singular values of the frontal slices $\hat{\mathcal{A}}^{(i)}$. If the singular values decay rapidly, we can approximate $(\sum_{j>k} \hat{\sigma}_j^{(i)})^{1/2}$ by $\hat{\sigma}_{k+1}^{(i)}$ and therefore

$$\mathbb{E}\, \|\mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A}\|_{\mathrm{F}} \leq \sqrt{1 + \frac{k}{p-1}} \max_{1 \leq i \leq n_3} \hat{\sigma}_{k+1}^{(i)}.$$

If the singular values of $\hat{\mathcal{S}}^{(i)}$ decay gradually, we can instead use the following approximation $\sum_{j>k}(\hat{\sigma}_j^{(i)})^2 \leq (m-k)(\hat{\sigma}_{k+1}^{(i)})^2$ where $\hat{\sigma}_{k+1}^{(i)}$ is the $(k+1)^{th}$ singular value of the ith frontal slice in the Fourier domain, and $m = \min\{n_1, n_2\}$. However, $\hat{\sigma}_{k+1}^{(i)}$ can be large, in which case the accuracy of the rt-SVD may be poor. We present a new algorithm (Algorithm 7) for tensor low-tubal-rank representation based on the t-product that applies the randomized subspace iteration to each frontal slice in the Fourier domain.

Algorithm 7: rt-SVD with subspace iteration, spatial domain version

Input : $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, target truncation term k, oversampling parameter p, the number of iterations q

Output: $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$, and $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$

1 Generate a Gaussian random tensor $\mathcal{W} \in \mathbb{R}^{n_2 \times (k+p) \times n_3}$;
2 Form a tensor $\mathcal{Y}_0 = \mathcal{A} * \mathcal{W}$ and compute the t-QR factorization $\mathcal{Y}_0 = \mathcal{Q}_0 * \mathcal{R}_0$;
3 **for** $i \leftarrow 1$ **to** q **do**
4 $\quad$ $\tilde{\mathcal{Y}}_i = \mathcal{A}^{\mathrm{T}} * \mathcal{Q}_{i-1}$ and compute the t-QR factorization $\tilde{\mathcal{Y}}_i = \tilde{\mathcal{Q}}_i * \tilde{\mathcal{R}}_i$;
5 $\quad$ $\mathcal{Y}_i = \mathcal{A} * \tilde{\mathcal{Q}}_i$ and compute the t-QR factorization $\mathcal{Y}_i = \mathcal{Q}_i * \mathcal{R}_i$;
6 **end**
7 Form a tensor $\mathcal{Q} = \mathcal{Q}_q$;
8 Form a tensor $\mathcal{B} = \mathcal{Q}^{\mathrm{T}} * \mathcal{A}$, the size of $\mathcal{B}$ is $(k+p) \times n_2 \times n_3$ which is smaller than tensor $\mathcal{A}$;
9 Compute t-SVD of $\mathcal{B}$, truncate it, and obtain $\mathcal{U}, \mathcal{S}_k, \mathcal{V}_k$;
10 Form the rt-SVD of $\mathcal{A}$, $\mathcal{A} \approx (\mathcal{Q} * \mathcal{U}) * \mathcal{S}_k * \mathcal{V}_k^{\mathrm{T}} = \mathcal{U}_k * \mathcal{S}_k * \mathcal{V}_k^{\mathrm{T}}$.

Algorithm 7 works efficiently when the singular values, given by the diagonals of $\hat{\mathcal{S}}^{(i)}$, decay at the same rate across each frontal slice of $\widehat{\mathcal{A}}$. However, when the singular values decay gradually only for some slices, Algorithm 7 may be wasteful in terms of computational costs, since some iterations can be stopped earlier than others. This can be avoided if a different number of iterations q_i is used for each frontal slice. Let us define the iteration vector as $\mathbf{q} = (q_1, q_2, \dots, q_{n_3})^{\top}$. We present an algorithm (Algorithm 8) that employs different iteration count in each frontal slice.

The expected error of the probabilistic part of Algorithm 8 is given in Theorem 5.

Theorem 5. *Given an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$ and an $n_2 \times (k+p) \times n_3$ tensor $\mathcal{W}$, if $\mathcal{Q}$ is obtained from Algorithm 8, then*

$$\mathbb{E}\, \left\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A} \right\|_F \leq \left(\frac{1}{n_3} \sum_{i=1}^{n_3} \left(1 + \frac{k}{p-1}(\tau_k^{(i)})^{4q_i} \right) \left(\sum_{j>k}(\hat{\sigma}_j^{(i)})^2 \right) \right)^{1/2},$$

where k is a target truncation term, $p \geq 2$ is the oversampling parameter, $\mathbf{q}$ is the iterations count vector, $\hat{\sigma}_j^{(i)}$ is the ith component of $\mathtt{fft}(\mathcal{S}(\mathtt{j},\mathtt{j},:),[\,],\mathtt{3})$, and the singular value gap $\hat{\tau}_k^{(i)} = \dfrac{\hat{\sigma}_{k+1}^{(i)}}{\hat{\sigma}_j^{(i)}} \ll 1$.

Proof. See Appendix B. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad$ $\square$

Algorithm 8: rt-SVD with subspace iterations, Fourier domain version

---

Input : $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, target truncation term k, parameter p, and the iterations vector $\mathbf{q}$
Output: $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$, and $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$

1 Generate a Gaussian random tensor $\mathcal{W} \in \mathbb{R}^{n_2 \times (k+p) \times n_3}$;

2 $\hat{\mathcal{A}} \leftarrow \texttt{fft}(\mathcal{A}, [], 3)$ and $\hat{\mathcal{W}} \leftarrow \texttt{fft}(\mathcal{W}, [], 3)$;

3 for $i \leftarrow 1$ **to** n_3 **do**

4 $\quad \hat{\mathcal{Y}}^{(i)} = \hat{\mathcal{A}}^{(i)} \hat{\mathcal{W}}^{(i)}$;

5 $\quad [\hat{\mathcal{Q}}_{j-1}^{(i)}, \sim] = \texttt{qr}(\hat{\mathcal{Y}}^{(\texttt{i})}, 0)$;

6 $\quad$ **for** $j \leftarrow 1$ **to** q_i **do**

7 $\quad\quad \hat{\mathcal{Z}}_j^{(i)} = (\hat{\mathcal{A}}^{(i)})^{\text{H}} \hat{\mathcal{Q}}_{j-1}^{(i)}$;

8 $\quad\quad [\hat{\mathcal{G}}_j^{(i)}, \sim] = \texttt{qr}(\hat{\mathcal{Z}}^{(\texttt{i})}, 0)$;

9 $\quad\quad \hat{\mathcal{Y}}_j^{(i)} = \hat{\mathcal{A}}^{(i)} \hat{\mathcal{G}}_j^{(i)}$;

10 $\quad\quad [\hat{\mathcal{Q}}_j^{(i)}, \sim] = \texttt{qr}(\hat{\mathcal{Y}}^{(\texttt{i})}, 0)$;

11 $\quad$ **end**

12 $\quad$ Form $\hat{\mathcal{Q}}^{(i)}$ as $\hat{\mathcal{Q}}^{(i)} = \hat{\mathcal{Q}}_j^{(i)}$;

13 $\quad \hat{\mathcal{B}}^{(i)} = (\hat{\mathcal{Q}}^{(i)})^{\text{H}} \hat{\mathcal{A}}^{(i)}$;

14 $\quad [\hat{\mathcal{U}}^{(i)}, \hat{\mathcal{S}}^{(i)}, \hat{\mathcal{V}}^{(i)}] = \texttt{svd}(\hat{\mathcal{B}}^{(\texttt{i})})$;

15 $\quad \hat{\mathcal{U}}_k^{(i)} = \hat{\mathcal{Q}}^{(i)} \hat{\mathcal{U}}^{(i)}$; $\hat{\mathcal{S}}_k^{(i)} = \hat{\mathcal{S}}^{(i)}(1:k, 1:k)$; $\hat{\mathcal{V}}_k^{(i)} = \hat{\mathcal{V}}^{(i)}(:, 1:k)$.

16 end

17 $\mathcal{U}_k \leftarrow \texttt{ifft}(\hat{\mathcal{U}}_{\texttt{k}}, [], 3)$; $\mathcal{S}_k \leftarrow \texttt{ifft}(\hat{\mathcal{S}}_{\texttt{k}}, [], 3)$; $\mathcal{V}_k \leftarrow \texttt{ifft}(\hat{\mathcal{V}}_{\texttt{k}}, [], 3)$.

If the iteration count $q_i = q$ for all $i = 1, \ldots, n_3$, then we can use the following simpler bound

$$\mathbb{E} \left\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\text{T}} * \mathcal{A} \right\|_{\text{F}} \leq \sqrt{1 + \frac{k}{p-1}(\tau_k^{\max})^{4q}} \left(\frac{1}{n_3} \sum_{i=1}^{n_3} \sum_{j>k} (\hat{\sigma}_j^{(i)})^2 \right)^{1/2}, \tag{4}$$

where $\tau_k^{\max} = \max_{1 \leq i \leq n_3} \tau_k^{(i)}$ is the largest singular value gap. In particular, $q = 0$ gives the same result as Theorem 4.

Theorem 5 suggests an effective strategy to pick the iteration count q_i. Suppose we are given a tolerance parameter $0 < \epsilon < 1$. Then we choose

$$q_i = \left\lceil \frac{1}{4} \log \frac{\epsilon(p-1)}{k} \middle/ \log \tau_k^{(i)} \right\rceil, \tag{5}$$

which ensures that the error in $\mathbb{E} \left\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\text{T}} * \mathcal{A} \right\|_{\text{F}}$ is at most $\sqrt{1 + \epsilon}$ of the optimal result in Theorem 3.

Theorems 4 and 5 provide insight into the average behavior of the error. This next result provides the tail bounds of the probabilistic error.

Theorem 6. *With the assumptions of Theorem 5, let $0 < \delta < 1$ be the failure probability and define the constant*

$$C_\delta = \frac{e\sqrt{k+p}}{p+1} \left(\frac{2}{\delta} \right)^{\frac{1}{p+1}} \left(\sqrt{n_2 - k} + \sqrt{k+p} + \sqrt{2 \log \frac{2}{\delta}} \right).$$

Then with probability at most δ,

$$\left\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\text{T}} * \mathcal{A} \right\|_F^2 \leq \frac{1}{n_3} \sum_{i=1}^{n_3} \left(1 + C_\delta^2 (\tau_k^{(i)})^{4q_i} \right) \left(\sum_{j>k} (\hat{\sigma}_j^{(i)})^2 \right).$$

Proof. See Appendix B. □

Theorem 6 shows that although the result holds with high probability, there is an arbitrary small chance that the upper bound may not hold. For the sample values $n_2 = 100$, $k = 30$, $p = 20$ and $\delta = 10^{-16}$, we obtain $C_\delta \approx 43$.

We now discuss the computational cost of Algorithm 8. Recall that the t-SVD requires $\mathcal{O}\left(n_1 n_2 n_3 \log n_3\right)$ flops to transform to the Fourier domain and an additional $\mathcal{O}\left(n_1 n_2 n_3 \min\{n_1, n_2\}\right)$ flops for the decomposition. On the other hand, the rt-SVD method only requires $\mathcal{O}\left(k n_1 n_2 n_3\right)$. The rt-SVD can be advantageous when $k \ll \min\{n_1, n_2\}$.

4   Numerical Results

In this section, we provide some numerical results on the accuracy of the proposed low-rank representations, as well as on the computation time of the proposed methods. The proposed algorithms are demonstrated on an application to facial recognition. The datasets for the experiments are a subset of the Cropped Extended Yale Face Dataset B [1] (abbreviated as Cropped Yale B dataset) and the dataset of faces maintained at AT&T Laboratories Cambridge [7] (abbreviated as AT&T dataset). The Cropped Yale B dataset has 1140 images that contains the first 30 possible illuminations of 38 different people. Each image has 192×168 pixels in a grayscale range. This is collected into a $192 \times 1140 \times 168$ tensor, and this tensor is denoted by $\mathcal{B}$. The AT&T dataset has 400 images that contains 10 different poses of 40 people. Each image has 112×92 pixels in a grayscale which is collected into a $112 \times 400 \times 92$ tensor, denoted as $\mathcal{E}$. The experiments were run on a laptop with 2.3 GHz Intel Core i7 and 8 GB memory.

4.1   Error Analysis

In Section 3, we derived theoretical results for the expected approximation errors of rt-SVD and rt-SVD with subspace iteration. Here, we provide some numerical results to demonstrate their comparative performance. We compare the relative errors obtained by using rt-SVD, rt-SVD with subspace iteration and the relative theoretically minimal errors on the dataset $\mathcal{B}$. The target truncation term k is allowed to vary between 50 and 180. We define the relative errors obtained by using the rt-SVD with subspace iteration as e_k^q,

$$e_k^q \;=\; \frac{\|(\mathcal{I} - \mathcal{Q}\mathcal{Q}^\top)\mathcal{A}\|_{\mathrm{F}}}{\|\mathcal{A}\|_F}, \tag{6}$$

where q represents the number of iterations (see Algorithm 7) and k denotes the target truncation term. Because the rt-SVD is a specific case of rt-SVD with subspace iteration with $q = 0$, we will use e_k^0 to denote the relative errors obtained by using rt-SVD.

Theorem 3, gives us the best possible relative error e_k, as a function of the target truncation term k.

$$e_k \;\equiv\; \frac{\|\mathcal{A} - \mathcal{A}_k\|_{\mathrm{F}}}{\|\mathcal{A}\|_F} \;=\; \frac{\|\hat{\mathcal{S}}(k+1:n, k+1:n, :)\|_{\mathrm{F}}}{\|\hat{\mathcal{S}}\|_F} \tag{7}$$

Figure 4.1 shows that the errors e_k^q, with different number of iterations, have the similar convergence trajectories and are quite close to the best possible theoretical error e_k. In other words, rt-SVD and rt-SVD with subspace iteration are both comparable in accuracy with the truncated t-SVD. Moreover, Figure 4.1 shows that e_k^q approaches e_k when the number of iterations q increases, yielding a more accurate approximation.

4.2   Facial Recognition

In this section, we apply the rt-SVD and rt-SVD with subspace iterations on the Cropped Yale B dataset and AT&T dataset. The images from each database is split into a training and test
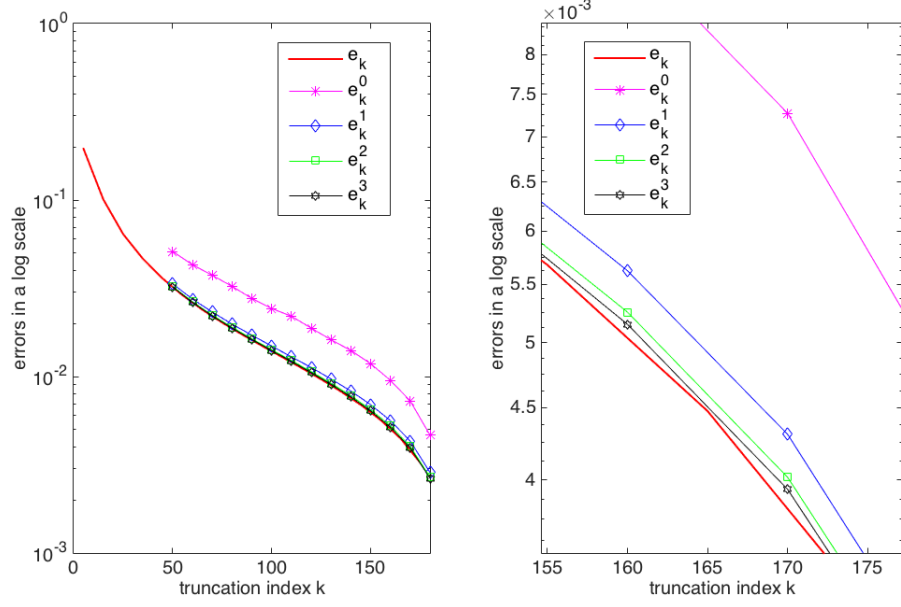
Figure 4.1.1: (left) The comparison of the theoretically minimal errors and the errors of rt-SVD with subspace iterations. (right) The zoomed in version of the left panel.

datasets. The tensors are constructed in such a way that the various images stored as lateral slices of the tensor. We process the training dataset and store a projector tensor $\mathcal{U}_k$ and a coefficient tensor $\mathcal{C}$ of smaller dimensions as described in Table 1. For each image in the test dataset, we obtain a tensor coefficient by projecting onto the training dataset, and the face is recognized as the lateral slice with the closest distance to the tensor coefficient. The procedure for processing the training and test datasets is shown in Table 1.

Table 1: The procedure of facial recognition based on t-SVD method

Facial Recognition Procedure	
For the training Dataset:	For the new image in the test dataset:
1. Form the training dataset into a third order tensor and calculate the mean lateral slice across the second dimension;	1. Form the new image as a tensor with only one lateral slice and subtract the mean lateral slice from it;
2. Calculate standard mean-shifted tensor and denote it as tensor $\mathcal{A}$;	2. Compute the standard mean-shifted lateral slice and denote it as $\mathcal{T}$;
3. Compute the truncated t-SVD of $\mathcal{A}$ or the approximated truncated t-SVD of $\mathcal{A}$ with target truncation term k;	3. Compute the coefficient tensor $\mathcal{C}_t = \mathcal{U}_k^\mathrm{T} * \mathcal{T}$;
4. Compute the coefficient tensor $\mathcal{C} = \mathcal{U}_k^\mathrm{T} * \mathcal{A}$, and store it with projector tensor $\mathcal{U}_k$.	4. Find the smallest distance of $\mathcal{C}_t$ with each lateral slices of $\mathcal{C}$.

To measure the performance more rigorously, we use 10-fold cross-validation. In 10-fold cross-validation, the dataset is randomly partitioned into 10 equal-size subsets. In the kth trial, the k subset (also referred to as a fold) is used as the test dataset, whereas the other 9 subsets are

simultaneously used as training dataset. Therefore, the algorithm will be tested 10 times with 10 different combinations of the same dataset. The randomized algorithms are run 20 times for each fold to compute the mean, maximum, and minimum of recognition rates. The recognition rate here is defined as

$$r = \frac{\texttt{the number of images recognized correctly}}{\texttt{the number of test images}}.$$

4.2.1   Cropped Yale Face B Dataset

There are 1140 images in Cropped Yale B Dataset, so the size of training dataset is $192 \times 1026 \times 168$ and the size of test dataset is $192 \times 114 \times 168$ in each fold. A few sample images from the Cropped Yale B dataset under different illuminations are shown in Figure 4.2.1.

Figure 4.2.1: Sample images from Cropped Yale B Dataset

.

Table 2 shows the accuracy of the rt-SVD algorithm, whereas Table 3 shows the accuracy of rt-SVD with subspace iterations. The computational times are reported in Figure 4.2.2. We observe that as the truncation term gets larger, the recognition rate increases, as well as the computation time. In practice, the target truncation term k depends on the tradeoff between recognition rate and computational time. We report results for target truncation term ranging from 25 to 50.

Table 2: Recognition Rates on Cropped Yale B dataset with $k = 25$

r	fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
The t-SVD method
	0.9912	1.0000	0.9211	1.0000	1.0000	0.9912	0.9035	0.9737	0.7368	0.9825
The rt-SVD method
mean	0.9912	1.0000	0.9175	0.9943	1.0000	0.9912	0.9035	0.9737	0.7368	0.9772
min	0.9912	1.0000	0.9123	0.9912	1.0000	0.9912	0.9035	0.9737	0.7368	0.9737
max	0.9912	1.0000	0.9211	1.0000	1.0000	0.9912	0.9035	0.9737	0.7368	0.9912
The rt-SVD method with subspace iterations $q = 1$
mean	0.9912	1.0000	0.9211	1.0000	1.0000	0.9912	0.9035	0.9737	0.7368	0.9833
min	0.9912	1.0000	0.9211	1.0000	1.0000	0.9912	0.9035	0.9737	0.7368	0.9737
max	0.9912	1.0000	0.9211	1.0000	1.0000	0.9912	0.9035	0.9737	0.7368	0.9912
The rt-SVD method with subspace iterations $q = 2$
mean	0.9912	1.0000	0.9211	1.0000	1.0000	0.9912	0.9035	0.9737	0.7368	0.9882
min	0.9912	1.0000	0.9211	1.0000	1.0000	0.9912	0.9035	0.9737	0.7368	0.9825
max	0.9912	1.0000	0.9211	1.0000	1.0000	0.9912	0.9035	0.9737	0.7368	0.9912

In Table 2, we make the following observations.

- The minimum and maximum recognition rates are very close to the mean recognition rate in the series of rt-SVD methods.

Table 3: Recognition Rates on Cropped Yale B dataset with $k = 50$

r	fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
The t-SVD method										
	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
The rt-SVD method										
mean	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
min	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
max	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
The rt-SVD method with subspace iterations $q = 1$										
mean	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
min	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
max	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
The rt-SVD method with subspace iterations $q = 2$										
mean	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
min	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
max	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912

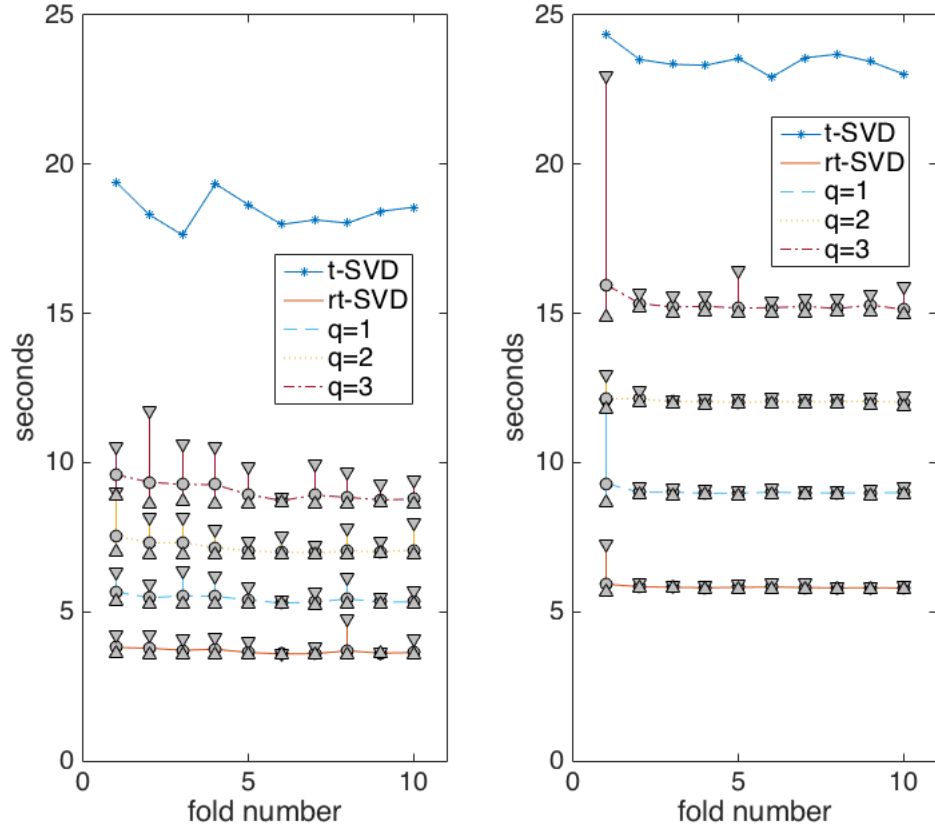Figure 4.2.2: (left) Running time to process the the training dataset of Cropped Yale B with $k = 25$. (right) Running time to process the training dataset of Cropped Yale B with $k = 50$.

- Comparing the recognition rate between t-SVD and the series of rt-SVD methods, they are identical in 7 out of 10 folds. In the other 3 folds (fold 3, fold 4, and fold 10), the difference is very slight, less than .001.

- In fold 10, the maximum recognition rates of the series of rt-SVD method are even slightly higher than the recognition rate of rt-SVD.

The randomized algorithms show almost no variation between different realizations which shows that while there is a probability of failure, however small, the accuracy of the low-rank representations concentrates about its mean value. Table 3 shows similar results as Table 2. In particular, the recognition rates are identical in each fold. The rt-SVD is about a third as expensive as the full t-SVD.

4.2.2 AT&T Dataset

For the AT&T dataset, there are 400 images, so the size of training dataset in each fold is $112 \times 360 \times 92$ and the size of test dataset in each fold is $192 \times 40 \times 168$. As compared to the Cropped Yale B dataset, the AT&T dataset has images of people with different poses, some sample faces are shown in Figure 4.2.2. As we discussed in subsection 4.2.1, we provide the result with two different target truncation terms, 15 and 25. Tables 4 and 5 show the performance of rt-SVD and rt-SVD with subspace iterations. Figure 4.2.4 shows the comparison of running times. The numerical results are consistent with numerical result on the Cropped Yale B dataset, and this demonstrates our algorithms have good performance on both the illumination-varying dataset and pose-varying dataset.

Figure 4.2.3: Sample images from AT&T dataset.

Table 4: Recognition Rates on AT&T dataset with $k = 15$

r	fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
The t-SVD method										
	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000
The rt-SVD method										
mean	0.9750	1.0000	1.0000	0.9775	0.9750	0.9537	0.9250	1.0000	0.9750	0.9013
min	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000
max	0.9750	1.0000	1.0000	1.0000	0.9750	0.9750	0.9250	1.0000	0.9750	0.9250
The rt-SVD method with subspace iterations $q = 1$										
mean	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000
min	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000
max	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000
The rt-SVD method with subspace iterations $q = 2$										
mean	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000
min	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000
max	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000

Table 5: Recognition Rates on AT&T dataset with $k = 25$

r	fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
The t-SVD method										
	0.9750	1.0000	1.0000	0.9750	0.9500	0.9500	0.9250	1.0000	0.9750	0.9000
The rt-SVD method										
mean	0.9750	1.0000	1.0000	0.9750	0.9587	0.9500	0.9250	1.0000	0.9750	0.9000
min	0.9750	1.0000	1.0000	0.9750	0.9500	0.9500	0.9250	1.0000	0.9750	0.9000
max	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000
The rt-SVD method with subspace iterations $q = 1$										
mean	0.9750	1.0000	1.0000	0.9750	0.9500	0.9500	0.9250	1.0000	0.9750	0.9000
min	0.9750	1.0000	1.0000	0.9750	0.9500	0.9500	0.9250	1.0000	0.9750	0.9000
max	0.9750	1.0000	1.0000	0.9750	0.9500	0.9500	0.9250	1.0000	0.9750	0.9000
The rt-SVD method with subspace iterations $q = 2$										
mean	0.9750	1.0000	1.0000	0.9750	0.9500	0.9500	0.9250	1.0000	0.9750	0.9000
min	0.9750	1.0000	1.0000	0.9750	0.9500	0.9500	0.9250	1.0000	0.9750	0.9000
max	0.9750	1.0000	1.0000	0.9750	0.9500	0.9500	0.9250	1.0000	0.9750	0.9000

4.3 Computation time for parallel implementation

In the facial recognition application, most of the computation time is spent on computing the compression, either exactly or approximately. In this subsection, we report the computation times of computing truncated t-SVD, rt-SVD, and rt-SVD with subspace iterations implemented in parallel on a cluster. The dataset we use, as an example, is the Cropped Yale B dataset $\mathcal{B}$, and the target truncation term (i.e., k) is 50. The experiments are run on Matlab 2015a in (Tufts cluster with Intel(R) Xeon(R) CPU X5675 running at 3.07 GHz (8 cores)), and up to 8 processors are used. The t-SVD is computed by Algorithm 4, the rt-SVD is computed using Algorithm 6, and the rt-SVD with subspace iterations is computed using Algorithm 8. The results are shown in Figure 4.3.1; as can be seen, with increasing number of processors, the runtime using the parallel computing, decreases on average.

5 Conclusion and Future Work

In this paper, we discussed the advantages and limitations of the matrix version of randomized algorithms and deterministic tensor-based algorithms. The algorithms we design combines the advantages of randomization for dimensionality reduction, and applied it to tensor decompositions based on the t-product. We extend the randomized SVD method to third order tensors and provide a basic version algorithm (rt-SVD), as well as a more general version algorithm (rt-SVD with subspace iterations). We showed, theoretically, that the expected errors of both rt-SVD and rt-SVD with subspace iterations, are comparable to the best rank-k approximation obtained using the deterministic t-SVD. Furthermore, we provide numerical support by means of application to facial recognition, on two commonly used publicly available datasets. Moreover, the randomized algorithms proposed here, can be readily parallelized and we demonstrate the benefits of parallelization on the computational time. This makes our algorithm both accurate and efficient, in practice. Our algorithms also have the added benefit that if the application is run on a distributed memory machine, the approximation factorization can be separately stored on different processors, and can be conveniently used later, without additional computation cost.

For future work, there are several potential research directions. First, the authors in [24] proposed a different method for truncated tensor SVD decomposition, called t-SVD II. This method, while deterministic, has good performance when the singular values decay at different rates across the frontal slices. One avenue of future research would be to develop a randomized version of t-SVD II, the benefits of which maybe higher recognition rate and lower computational cost. In this paper, we have used Gaussian random tensors for dimensionality reduction. The second possible direction is

to investigate other possible structured random tensors, based on subsampled Hadamard or Fourier Transform. One drawback of the current methods for compression is that, if a new image or set of images is introduced into the database, then we have to recompute the approximate factorization which can be very expensive. It would be interesting to extend updating or down-dating of matrix factorizations to tensors using the t-product. Finally, the randomized algorithms can be extended to other tensor decompositions based on invertible linear transforms (see [28] for more details).

A Randomized subspace iteration

Our goal in this section is to prove Theorem 2. Before we give the proof, we define the following quantities that will be used later.

Partition $\boldsymbol{A}$ conformally as

$$\boldsymbol{A} \;=\; \begin{bmatrix} \boldsymbol{U}_1 & \boldsymbol{U}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_1 & \\ & \boldsymbol{\Sigma}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_1^{\mathrm{H}} \\ \boldsymbol{V}_2^{\mathrm{H}} \end{bmatrix}. \tag{8}$$

Define $\boldsymbol{W}_1 \equiv \boldsymbol{V}_1^{\mathrm{H}}\boldsymbol{W}$ and $\boldsymbol{W}_2 \equiv \boldsymbol{V}_2^{\mathrm{H}}\boldsymbol{W}$. The subspace iteration (Algorithm 2) with random starting guess, can be alternatively expressed as

$$\boldsymbol{Y} \;=\; (\boldsymbol{A}\boldsymbol{A}^{\mathrm{H}})^q \boldsymbol{A}\boldsymbol{W} \;=\; \boldsymbol{U} \begin{bmatrix} \boldsymbol{\Sigma}_1^{2q+1} & \\ & \boldsymbol{\Sigma}_2^{2q+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{W}_1 \\ \boldsymbol{W}_2 \end{bmatrix}.$$

Suppose $\boldsymbol{V}$ has full column rank; we denote the projection matrix by $\boldsymbol{\mathcal{P}}_{\boldsymbol{V}} \equiv \boldsymbol{V}(\boldsymbol{V}^{\mathrm{H}}\boldsymbol{V})^{-1}\boldsymbol{V}^{\mathrm{H}}$ corresponding to $\mathrm{Range}(V)$. Additionally, because of $\boldsymbol{V}$ has orthonormal columns, then the expression for the projector simplifies to $\boldsymbol{\mathcal{P}}_{\boldsymbol{V}} = \boldsymbol{V}\boldsymbol{V}^{\mathrm{H}}$.

We present a result that characterizes the error of the low-rank matrix approximation. The result makes minimal assumptions about the sampling matrix $\boldsymbol{W}$, and is therefore, applicable to any distribution. This result will be used to prove Theorem 2.

Theorem 7. *Let $\boldsymbol{A}$ be an $m \times n$ matrix with SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{H}}$ and $k \geq 0$ be a fixed parameter. Choose a matrix $\boldsymbol{W} \in \mathbb{C}^{n \times \ell}$, define $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ as above and assume that $\boldsymbol{W}_1$ has full row rank. Compute $\boldsymbol{Q}$ using Algorithm 2. Then the approximation error satisfies*

$$\left\| (\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{H}})\boldsymbol{A} \right\|_F^2 \;\leq\; \|\boldsymbol{\Sigma}_2\|_F^2 + \tau_k^{4q} \left\| \boldsymbol{\Sigma}_2 \boldsymbol{W}_2 \boldsymbol{W}_1^\dagger \right\|_F^2. \tag{9}$$

Proof. Define matrices $\boldsymbol{Z}$ and $\boldsymbol{F}$ as

$$\boldsymbol{Z} \;=\; \boldsymbol{U}^{\mathrm{H}}\boldsymbol{Y}\boldsymbol{W}_1^\dagger \boldsymbol{\Sigma}_1^{-(2q+1)} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{F} \end{bmatrix} \qquad \boldsymbol{F} \equiv \boldsymbol{\Sigma}_2^{2q+1} \boldsymbol{W}_2 \boldsymbol{W}_1^\dagger \boldsymbol{\Sigma}_1^{-(2q+1)}.$$

The construction of $\boldsymbol{Z}$ ensures that the following results hold

$$\mathrm{Range}(\boldsymbol{Z}) \subset \mathrm{Range}(\boldsymbol{Y}) = \mathrm{Range}(\boldsymbol{U}^{\mathrm{H}}\boldsymbol{Y}) = \mathrm{Range}(\boldsymbol{U}^{\mathrm{H}}\boldsymbol{Q}). \tag{10}$$

Plugging the SVD of $\boldsymbol{A}$ into the error in the low-rank approximation

$$\left\| (\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{H}})\boldsymbol{A} \right\|_F^2 \;=\; \left\| (\boldsymbol{I} - \boldsymbol{\mathcal{P}}_{\boldsymbol{Q}})\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{H}} \right\|_F^2 = \|(\boldsymbol{I} - \boldsymbol{\mathcal{P}}_{\boldsymbol{Q}})\boldsymbol{U}\boldsymbol{\Sigma}\|_F^2,$$

where the last step follows since the Frobenius norm is unitarily invariant. Recall that [21, Proposition 8.4] implies that $\boldsymbol{U}^{\mathrm{H}}\boldsymbol{\mathcal{P}}_{\boldsymbol{Q}}\boldsymbol{U} = \boldsymbol{\mathcal{P}}_{\boldsymbol{U}^{\mathrm{H}}\boldsymbol{Q}}$, and from [21, Proposition 8.5] and (10) follows

$$\|(\boldsymbol{I} - \boldsymbol{\mathcal{P}}_{\boldsymbol{Q}})\boldsymbol{U}\boldsymbol{\Sigma}\|_F^2 = \left\| \boldsymbol{\Sigma}^{\mathrm{H}}(\boldsymbol{I} - \boldsymbol{\mathcal{P}}_{\boldsymbol{U}^{\mathrm{H}}\boldsymbol{Q}})\boldsymbol{\Sigma} \right\|_F \leq \left\| \boldsymbol{\Sigma}^{\mathrm{H}}(\boldsymbol{I} - \boldsymbol{\mathcal{P}}_{\boldsymbol{Z}})\boldsymbol{\Sigma} \right\|_F = \|(\boldsymbol{I} - \boldsymbol{\mathcal{P}}_{\boldsymbol{Z}})\boldsymbol{\Sigma}\|_F^2.$$

Following the steps of [21, Theorem 9.1] it can be shown that

$$(\boldsymbol{I} - \boldsymbol{\mathcal{P}}_{\boldsymbol{Z}})\boldsymbol{\Sigma} = \begin{bmatrix} (\boldsymbol{I} + \boldsymbol{F}^{\mathrm{H}}\boldsymbol{F})^{-1}\boldsymbol{F}^{\mathrm{H}}\boldsymbol{F}\boldsymbol{\Sigma}_1 \\ \left(\boldsymbol{I} - \boldsymbol{F}(\boldsymbol{I} + \boldsymbol{F}^{\mathrm{H}}\boldsymbol{F})^{-1}\boldsymbol{F}^{\mathrm{H}} \right)\boldsymbol{\Sigma}_2 \end{bmatrix}.$$

We also recall from [21, Theorem 9.1] the following inequality

$$I - F(I + F^{\mathrm{H}}F)^{-1}F^{\mathrm{H}} \preceq I. \tag{11}$$

We can then bound

$$\|(I - \mathcal{P}_Z)\Sigma\|_{\mathrm{F}}^2 = \left\|(I + F^{\mathrm{H}}F)^{-1}F^{\mathrm{H}}F\Sigma_1\right\|_{\mathrm{F}}^2 + \left\|\left(I - F(I + F^{\mathrm{H}}F)^{-1}F^{\mathrm{H}}\right)\Sigma_2\right\|_{\mathrm{F}}^2 \tag{12}$$

where the last step follows from (11) and [21, Proposition 8.4]. We can bound

$$\left\|(I + F^{\top}F)^{-1}F^{\mathrm{H}}F\Sigma_1\right\|_{\mathrm{F}} \leq \|F(I + F^{\mathrm{H}}F)^{-1}\|_2 \|F\Sigma_1\|_{\mathrm{F}} \leq \|F\Sigma_1\|_{\mathrm{F}},$$

where the second result follows from the sub-multiplicativity of the Frobenius norm, and the last result follows from a simple SVD argument. Together with (12) this gives

$$\|(I - \mathcal{P}_Q)A\|_{\mathrm{F}}^2 \leq \|(I - \mathcal{P}_Z)\Sigma\|_{\mathrm{F}}^2 \leq \|\Sigma_2\|_{\mathrm{F}}^2 + \|F\Sigma_1\|_{\mathrm{F}}^2. \tag{13}$$

Observe that $F\Sigma_1 = \Sigma_2^{2q}(\Sigma_2 W_2 W_1^{\dagger})\Sigma_1^{-2q}$, and together with some simple norm inequalities, it follows that

$$\|F\Sigma_1\|_{\mathrm{F}} \leq \|\Sigma_2^{2q}\|_2 \|\Sigma_1^{-2q}\|_2 \left\|\Sigma_2 W_2 W_1^{\dagger}\right\|_{\mathrm{F}} \leq \tau_k^{2q} \left\|\Sigma_2 W_2 W_1^{\dagger}\right\|_{\mathrm{F}}.$$

Combining this result with (13) gives the desired result. $\qquad\square$

We are now ready to prove Theorem 2.

Proof of Theorem 2 Apply the same arguments as [21, Theorem 10.5] to Theorem 7. $\square$

Remark 1. *The estimate in Theorem 2 is sharper than [19, Theorem 5.7] when $\ell = k + p$ and $p \geq 2$. This implies that the singular value gap is chosen to be between k and $k + 1$. To see this, we can simplify the bound in Theorem 2 to*

$$\mathbb{E}\left\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^{\mathrm{H}}\mathbf{A}\right\|_F \leq \sqrt{\sum_{j>k}\sigma_j^2 + \frac{k\tau_k^{4q}(n-k)}{p-1}\sigma_{k+1}^2}.$$

The equivalent bound from [19, Theorem 5.7] is

$$\mathbb{E}\left\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^{\mathrm{H}}\mathbf{A}\right\|_F \leq \sqrt{\sum_{j>k}\sigma_j^2 + k\tau_k^{4q}(n-k)C^2\sigma_{k+1}^2},$$

where $C = (\sqrt{n-k} + \sqrt{k+p} + 7)(\frac{4e\sqrt{k+p}}{p+1})$. Since the first terms in both expressions are the same, comparing the second terms we have

$$\frac{(\sqrt{n-k} + \sqrt{k+p} + 7)^2}{n-k}\left(\frac{16e^2(k+p)}{(p+1)^2}\right)(p-1) > 1.$$

This shows that our bound is tighter under these conditions. For any other choice of p and ℓ, [19, Theorem 5.7] maybe sharper depending on the singular value decay.

B Proofs of Section 3

We present the following Lemma, which is similar to Parseval's theorem.

Lemma 1. *Let $\mathcal{A}$ be a real $n_1 \times n_2 \times n_3$ tensor. Then*

$$\|\mathcal{A}\|_F^2 = \frac{1}{n_3}\sum_{i=1}^{n_3}\left\|\hat{\mathcal{A}}^{(i)}\right\|_F^2,$$

where $\hat{\mathcal{A}}^{(i)} \equiv \hat{\mathcal{A}}(:,:,i)$ is the ith frontal slice of the tensor in the Fourier domain.

Proof. Define the operation $\mathrm{circ}\,(\mathrm{MatVec}(\mathcal{A}))$ as

$$\mathrm{circ}\,(\mathrm{MatVec}(\mathcal{A})) = \begin{bmatrix} \mathcal{A}^{(1)} & \mathcal{A}^{(n_3)} & \cdots & \mathcal{A}^{(2)} \\ \mathcal{A}^{(2)} & \mathcal{A}^{(1)} & \cdots & \mathcal{A}^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{A}^{(n_3)} & \mathcal{A}^{(n_3-1)} & \cdots & \mathcal{A}^{(1)} \end{bmatrix}.$$

Let $\boldsymbol{F}_{n_3}$ be the Discrete Fourier Matrix. Then [30, Equation 3.1] implies that

$$(\boldsymbol{F}_{n_3} \otimes \boldsymbol{I}_{n_1})\mathrm{circ}\,(\mathrm{MatVec}(\mathcal{A}))\,(\boldsymbol{F}_{n_3}^{\mathrm{H}} \otimes \boldsymbol{I}_{n_2}) = \begin{bmatrix} \hat{\mathcal{A}}^{(1)} & & & \\ & \hat{\mathcal{A}}^{(2)} & & \\ & & \ddots & \\ & & & \hat{\mathcal{A}}^{(n_3)} \end{bmatrix}.$$

The unitary invariance of the Frobenius norm implies

$$\left\| (\boldsymbol{F}_{n_3} \otimes \boldsymbol{I}_{n_1})\mathrm{circ}\,(\mathrm{MatVec}(\mathcal{A}))\,(\boldsymbol{F}_{n_3}^{\mathrm{H}} \otimes \boldsymbol{I}_{n_2}) \right\|_{\mathrm{F}}^2 = \sum_{i=1}^{n_3} \left\| \hat{\mathcal{A}}^{(i)} \right\|_{\mathrm{F}}^2.$$

Alternatively, from its definition it follows that

$$\|\mathcal{A}\|_{\mathrm{F}}^2 = n_3 \sum_{i=1}^{n_3} \left\| \mathcal{A}^{(i)} \right\|_{\mathrm{F}}^2 = \left\| (\boldsymbol{F}_{n_3} \otimes \boldsymbol{I}_{n_1})\mathrm{circ}\,(\mathrm{MatVec}(\mathcal{A}))\,(\boldsymbol{F}_{n_3}^{\mathrm{H}} \otimes \boldsymbol{I}_{n_2}) \right\|_{\mathrm{F}}^2.$$

Equating appropriate terms gives us the desired result. $\square$

We now use this result to prove Theorems 4 and 5.

Proof of Theorem 4 Using Lemma 1 and the linearity of the expectation, we can write

$$\mathbb{E} \left\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A} \right\|_{\mathrm{F}}^2 \le \frac{1}{n_3} \left(\sum_{i=1}^{n_3} \mathbb{E} \left\| \hat{\mathcal{A}}^{(i)} - \hat{\mathcal{Q}}^{(i)}(\hat{\mathcal{Q}}^{(i)})^{\mathrm{H}} \hat{\mathcal{A}}^{(i)} \right\|_{\mathrm{F}}^2 \right). \tag{14}$$

We can bound the individual terms in the summation by applying the result from (2),

$$\mathbb{E} \left\| \hat{\mathcal{A}}^{(i)} - \hat{\mathcal{Q}}^{(i)}(\hat{\mathcal{Q}}^{(i)})^{\top} \hat{\mathcal{A}}^{(i)} \right\|_{\mathrm{F}}^2 \le \left(1 + \frac{k}{p-1} \right) \left(\sum_{j>k} (\hat{\sigma}_j^{(i)})^2 \right). \tag{15}$$

Substitute inequality (15) into inequality (14) to obtain

$$\mathbb{E} \left\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} \right\|_{\mathrm{F}}^2 \le \left(1 + \frac{k}{p-1} \right) \left(\frac{1}{n_3} \sum_{i=1}^{n_3} \sum_{j>k} (\hat{\sigma}_j^{(i)})^2 \right).$$

Finally, using Hölder's inequality

$$\mathbb{E} \left\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A} \right\|_{\mathrm{F}} \le \left(\mathbb{E} \left\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A} \right\|_{\mathrm{F}}^2 \right)^{1/2},$$

from which the desired result follows. $\square$

Proof of Theorem 5 Apply the results of Theorem 2 to each frontal slice in (14) and follow the remaining steps of Theorem 4. $\square$

We next prove Theorem 6. We will need a result from [19].

Lemma 2. *[19, Theorem 5.8] Let $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ be defined as in Appendix A. Let $0 < \delta < 1$ be the failure probability and define the constant*

$$C_\delta = \frac{e\sqrt{k+p}}{p+1} \left(\frac{2}{\delta}\right)^{\frac{1}{p+1}} \left(\sqrt{n-k} + \sqrt{k+p} + \sqrt{2\log\frac{2}{\delta}}\right).$$

Then $Prob\left\{\|\boldsymbol{W}_2\|_2\|\boldsymbol{W}_1^\dagger\|_2 \geq C_\delta\right\} \leq \delta.$

Proof of Theorem 6 Apply Lemma 1 and Theorem 7 to obtain

$$\left\|\mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A}\right\|_{\mathrm{F}}^2 \leq \frac{1}{n_3}\sum_{i=1}^{n_3}\left(\left\|\hat{\mathcal{S}}_2^{(i)}\right\|_{\mathrm{F}}^2 + (\tau_k^{(i)})^{4q}\left\|\hat{\mathcal{S}}_2^{(i)}\hat{\mathcal{W}}_2^{(i)}(\hat{\mathcal{W}}_1^{(i)})^\dagger\right\|_{\mathrm{F}}^2\right).$$

The sub-multiplicative property of the Frobenius norm implies

$$\left\|\hat{\mathcal{S}}_2^{(i)}\hat{\mathcal{W}}_2^{(i)}(\hat{\mathcal{W}}_1^{(i)})^\dagger\right\|_{\mathrm{F}} \leq \left\|\hat{\mathcal{W}}_2^{(i)}(\hat{\mathcal{W}}_1^{(i)})^\dagger\right\|_2\left\|\hat{\mathcal{S}}_2^{(i)}\right\|_{\mathrm{F}}.$$

Plugging this into the above equation gives us

$$\left\|\mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A}\right\|_{\mathrm{F}}^2 \leq \frac{1}{n_3}\sum_{i=1}^{n_3}\left(1 + (\tau_k^{(i)})^{4q}\left\|\hat{\mathcal{W}}_2^{(i)}(\hat{\mathcal{W}}_1^{(i)})^\dagger\right\|^2\right)\left(\sum_{j>k}(\hat{\sigma}_j^{(i)})^2\right).$$

Since Gaussian random matrices are invariant to rotations, $\mathcal{W}_2^{(i)}$ and $\mathcal{W}_1^{(i)}$ will be independent for each $i = 1,\ldots,n_3$. We can therefore apply Lemma 2 to obtain the desired result. $\square$

References

[1] The extended Yale Face Dataset B. `http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html`.

[2] Dimitris Achlioptas. Database-friendly random projections: Johnson–Lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

[3] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast Johnson–Lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563. ACM, 2006.

[4] Nir Ailon and Bernard Chazelle. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

[5] Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.

[6] David J Biagioni, Daniel Beylkin, and Gregory Beylkin. Randomized interpolative decomposition of separated representations. *Journal of Computational Physics*, 281:116–134, 2015.

[7] AT&T Laboratories Cambridge. The database of faces. `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`.

[8] Tony F Chan and Per Christian Hansen. Low-rank revealing QR factorizations. *Numerical Linear Algebra with Applications*, 1(1):33–44, 1994.

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

[10] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

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

[12] Mihai Dorobantu and Björn Engquist. Wavelet-based numerical homogenization. *SIAM Journal on Numerical Analysis*, 35(2):540–559, 1998.

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

[14] Daniel M Dunlavy, Tamara G Kolda, and W Philip Kegelmeyer. Multilinear algebra for analyzing data with multiple linkages. *Graph Algorithms in the Language of Linear Algebra*, pages 85–114, 2011.

[15] Gregory Ely, Shuchin Aeron, Ning Hao, Misha E Kilmer, et al. 5D and 4D pre-stack seismic data completion using tensor nuclear norm (TNN). *preprint*, 2013.

[16] Peter Frankl and Hiroshi Maehara. The Johnson–Lindenstrauss lemma and the sphericity of some graphs. *Journal of Combinatorial Theory, Series B*, 44(3):355–362, 1988.

[17] Andreas Frommer, Thomas Lippert, Björn Medeke, and Klaus Schilling. *Numerical Challenges in Lattice Quantum Chromodynamics: Joint Interdisciplinary Workshop of John Von Neumann Institute for Computing, Jülich, and Institute of Applied Computer Science, Wuppertal University, August 1999*, volume 15. Springer Science & Business Media, 2012.

[18] David F Gleich, Chen Greif, and James M Varah. The power and Arnoldi methods in an algebra of circulants. *Numerical Linear Algebra with Applications*, 20(5):809–831, 2013.

[19] Ming Gu. Subspace iteration randomization and singular value problems. *SIAM Journal on Scientific Computing*, 37(3):A1139–A1173, 2015.

[20] Nathan Halko, Per-Gunnar Martinsson, Yoel Shkolnisky, and Mark Tygert. An algorithm for the principal component analysis of large data sets. *SIAM Journal on Scientific computing*, 33(5):2580–2594, 2011.

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

[22] Per Christian Hansen, James G Nagy, and Dianne P O'leary. *Deblurring images: matrices, spectra, and filtering*, volume 3. Siam, 2006.

[23] N Hao, L Horesh, and ME Kilmer. Nonnegative tensor decomposition. In *Compressed Sensing & Sparse Filtering*, pages 123–148. Springer, 2014.

[24] Ning Hao, Misha E Kilmer, Karen Braman, and Randy C Hoover. Facial recognition using tensor–tensor decompositions. *SIAM Journal on Imaging Sciences*, 6(1):437–463, 2013.

[25] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. springer New York, 2009.

[26] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *J. Math. Phys*, 6(1):164–189, 1927.

[27] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[28] Eric Kernfeld, Misha Kilmer, and Shuchin Aeron. Tensor–tensor products with invertible linear transforms. *Linear Algebra and its Applications*, 485:545–570, 2015.

[29] Misha E Kilmer, Karen Braman, Ning Hao, and Randy C Hoover. Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging. *SIAM Journal on Matrix Analysis and Applications*, 34(1):148–172, 2013.

[30] Misha E Kilmer and Carla D Martin. Factorization strategies for third-order tensors. *Linear Algebra and its Applications*, 435(3):641–658, 2011.

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

[32] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167–20172, 2007.

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

[34] Carla D Martin, Richard Shafer, and Betsy LaRue. An order-p tensor factorization with applications in imaging. *SIAM Journal on Scientific Computing*, 35(1):A474–A490, 2013.

[35] Rowayda A Sadek. SVD based image processing applications: State of the ART, contributions and research challenges. *arXiv preprint arXiv:1211.7102*, 2012.

[36] Oguz Semerci, Ning Hao, Misha E Kilmer, and Eric L Miller. Tensor-based formulation and nuclear norm regularization for multi-energy computed tomography. *arXiv preprint arXiv:1307.5348*, 2013.

[37] Ryan Sigurdson and Carmeliza Navasca. Randomized tensor–based algorithm for image classification. In *Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on*, pages 1984–1988. IEEE, 2012.

[38] Horst D Simon and Hongyuan Zha. Low-rank matrix approximation using the lanczos bidiagonalization process with applications. *SIAM Journal on Scientific Computing*, 21(6):2257–2274, 2000.

[39] Age Smilde, Rasmus Bro, and Paul Geladi. *Multi-way analysis: applications in the chemical sciences*. John Wiley & Sons, 2005.

[40] Luis Tenorio. Statistical regularization of inverse problems. *SIAM review*, 43(2):347–366, 2001.

[41] Charalampos E Tsourakakis. MACH: Fast randomized tensor decompositions. In *SDM*, pages 689–700. SIAM, 2010.

[42] Ledyard R Tucker. Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 122137, 1963.

[43] Alle-Jan Van der Veen. A Schur method for low-rank matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, 17(1):139–160, 1996.

[44] Ashlee Vance. Netflix, Reed Hastings survive missteps to join Silicon Valley's elite. *Bloomberg Businessweek*, 2013.

[45] M Alex O Vasilescu and Demetri Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *Computer VisionECCV 2002*, pages 447–460. Springer, 2002.

[46] Rafi Witten and Emmanuel Candes. Randomized algorithms for low-rank matrix factorizations: sharp performance bounds. *Algorithmica*, pages 1–18, 2013.

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

[48] Zemin Zhang, Gregory Ely, Shuchin Aeron, Ning Hao, and Misha Kilmer. Novel methods for multilinear data completion and de-noising based on tensor-SVD. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3842–3849. IEEE, 2014.
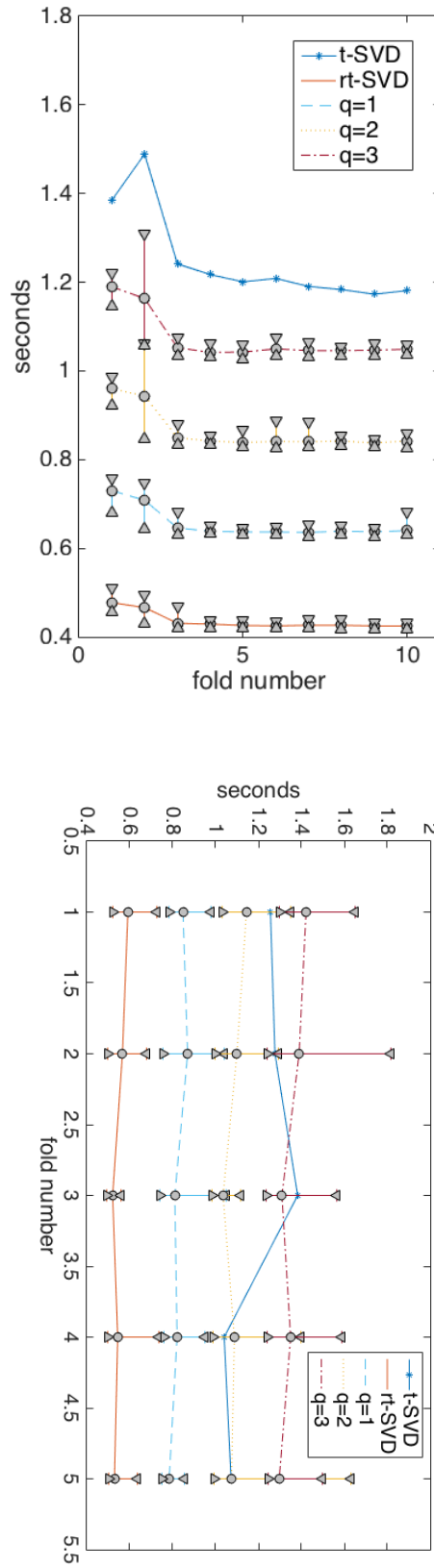
Figure 4.2.4: (up) Running time to process the *AT&T* training dataset with $k = 15$. (down) Running time to process the *AT&T* training dataset with $k = 25$.
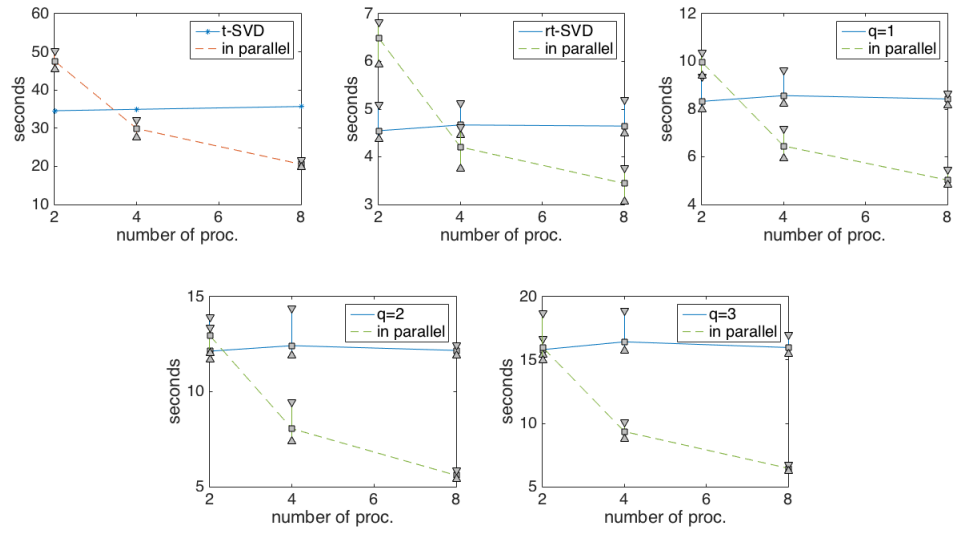
Figure 4.3.1: The computation time of t-SVD, rt-SVD, and rt-SVD with subspace iterations ($q = 1 - 3$), with and without, parallel computing.