Journal of Machine Learning Research 27 (2026) 1-56 Submitted 6/23; Revised 8/25; Published 2/26

Efficient frequent directions algorithms for approximate decomposition of matrices and higher-order tensors

This work is devoted to the 80th birthday of Professor Liqun Qi.

Maolin Che CHNCML@OUTLOOK.COM & MLCHE@GZU.EDU.CN
School of Mathematics and Statistics and State Key Laboratory of Public Big Data
Guizhou University
Guiyang, 550025, Guizhou, P. R. of China

Yimin Wei YMWEI@FUDAN.EDU.CN
School of Mathematical Sciences and Key Laboratory of Mathematics for Nonlinear Sciences
Fudan University
Shanghai, 200433, P. R. of China

Hong Yan H.YAN@CITYU.EDU.HK
Department of Electrical Engineering
City University of Hong Kong
83 Tat Chee Avenue, Kowloon, Hong Kong

Editor: Animashree Anandkumar

Abstract

In the framework of the FD (frequent directions) algorithm, we first develop two efficient algorithms for low-rank matrix approximations under the embedding matrices composed of the product of any SpEmb (sparse embedding) matrix and any standard Gaussian matrix, or any SpEmb matrix and any SRHT (subsampled randomized Hadamard transform) matrix. The theoretical results are also achieved based on the bounds of singular values of standard Gaussian matrices and the theoretical results for SpEmb and SRHT matrices. With a given Tucker-rank, we then obtain several efficient FD-based randomized variants of T-HOSVD (the truncated high-order singular value decomposition) and ST-HOSVD (sequentially T-HOSVD), which are two common algorithms for computing the approximate Tucker decomposition of any tensor with a given Tucker-rank. We also consider efficient FD-based randomized algorithms for computing the approximate TT (tensor-train) decomposition of any tensor with a given TT-rank. Finally, we illustrate the efficiency and accuracy of these algorithms using synthetic and real-world matrix (and tensor) data.

Keywords: Low-rank approximation, approximate Tucker decomposition, approximate TT decomposition, T-HOSVD, ST-HOSVD, TT-SVD, randomized algorithms, random projection, frequent directions, sparse embedding matrices, standard Gaussian matrices, subsampled randomized Hadamard transform, classification

1. Introduction

Low-rank matrix approximation is an essential tool in many applications, such as machine learning, signal processing, data compression, principal component analysis, and natural language processing. The optimal low-rank approximation can be obtained by matrix decomposition including singular value decomposition (SVD), rank-revealing QR factorization, and two-side orthogonal factorization (cf. Golub and Van Loan (2013)). As the dimension of the matrix increases, these methods become expensive and inefficient. For example, computing the SVD of $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ with $I_1 < I_2$ requires $\Theta(I_1^2 I_2)$ operations, which is prohibitive for larger I_1 and I_2.

To overcome this issue, several randomized algorithms for the low-rank approximation have been designed. Comparison with the truncated SVD, randomized algorithms are often faster and robust. Two common randomization techniques are employed to obtain the low-rank approximation. Let $\mu \leq I_1$ be the rank parameter. For the random projection technique (cf. Woolfe et al. (2008); Ailon and Chazelle (2009); Halko et al. (2011); Tropp (2011); Boutsidis and Gittens (2013); Meng and Mahoney (2013); Nelson and Nguyên (2013); Musco and Musco (2015)), a matrix $\mathbf{B} \in \mathbb{R}^{I_1 \times (\mu + K)}$ is obtained by using a random projection matrix to project $\mathbf{A}$, where $K > 0$ is any oversampling parameter. For the random sampling technique (cf. Deshpande et al. (2006); Drineas et al. (2006); Wang and Zhang (2013); Holodnak and Ipsen (2015)), we form a matrix $\mathbf{B}$ by choosing a set of columns from $\mathbf{A}$. Then, the rank-μ approximation $\widetilde{\mathbf{A}}_\mu$ to $\mathbf{A}$ is the best rank-μ approximation to the projection $\mathbf{A}$ on the column range of $\mathbf{B}$. One of the algorithms for computing $\widetilde{\mathbf{A}}_\mu$ can be found in (Boutsidis et al. (2014)).

With the exception of randomized algorithms, another proposed algorithm for the low-rank approximation is the FD algorithm (cf. Ghashami and Phillips (2014); Liberty (2013)), which is motivated by the idea of frequent items in item frequency approximation problem (cf. Misra and Gries (1982)). It is worthy noting that it needs $\Theta(I_1 I_2 L)$ operations to obtain $\mathbf{B} \in \mathbb{R}^{I_1 \times L}$ by applying the FD algorithm to $\mathbf{A}$. Meanwhile, an exact full SVD is implemented in each iteration of the FD algorithm. Furthermore, several new variants for the FD algorithm are proposed in (Desai et al. (2016)), which maintain error guarantees and significantly improve performance.

1.1 Related works and main contributions

Comparison to the FD algorithm, to maintain high accuracy and achieve low computational cost, several randomized variants of the FD algorithm are proposed based on different randomization techniques. Let L, q, I_1 and I_2 be positive integers with $L < I_1$ and $I_2/q > L$ being any integer, each randomized variant of the FD algorithm can be divided into three stages: (a) dividing all the columns into q parts, that is, $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_q]$ with $\mathbf{A}_i = \mathbf{A}(:, (i-1)I_2/q + 1, iI_2/q) \in \mathbb{R}^{I_1 \times (I_2/q)}$; (b) for each $i = 1, 2, \dots, q$, multiplying $\mathbf{A}_i$ by any random projection and/or sampling matrix $\mathbf{\Omega}_i \in \mathbb{R}^{I_2/q \times L}$ to obtain a matrix $\mathbf{C}_i \in \mathbb{R}^{I_1 \times L}$ as $\mathbf{C}_i = \mathbf{A}_i \mathbf{\Omega}_i$; and (c) applying the FD algorithm to $\mathbf{C} = [\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_q]$ to obtain $\mathbf{B} \in \mathbb{R}^{I_1 \times L}$.

Based on the choice of each $\mathbf{\Omega}_i$, there exist several types for randomized variants of the FD algorithm, such as the sparse FD (SFD) algorithm (cf. (Ghashami et al., 2016a, Algorithm 2)), the Faster FD (FFD) algorithm (cf. (Chen et al., 2017, Algorithm 3)), the sparse subspace embedding FD (SpFD) algorithm (cf. (Teng and Chu, 2018, Algorithm 3)), and the block Krylov iteration FD (BKIFD) algorithm (cf. (Wang et al., 2023, Algorithm 4)). Note that in the SFD algorithm, random block power method techniques (cf. Halko et al. (2011)) are used to obtain each matrix $\mathbf{\Omega}_i$. Hence, both the SFD and BKIFD algorithms are suitable for the case that for each i, the singular values of $\mathbf{A}_i$ decay slowly. Meanwhile, the SpFD algorithm can also be improved by combining the random block power method technique, whose accuracy is illustrated via several examples in Section 5.

Due to useful properties of standard Gaussian matrices, Gaussian projection is a common random projection technique for the dense matrix case. Standard Gaussian matrices are often used in theoretical analysis due to the availability of excellent probability theory and concentration of measure arguments. Hence, when setting each $\mathbf{\Omega}_i$ as any standard Gaussian matrix, **our first goal** is to derive an efficient FD algorithm for finding a much smaller matrix $\mathbf{B} \in \mathbb{R}^{I_1 \times L}$ from a dense matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$. It is worthy noting that our algorithm is different from the SpFD algorithm because each $\mathbf{\Omega}_i$ in the later is any SpEmb matrix. In Section 3, we consider details for the SFD, FFD and BKIFD algorithms and the difference among our algorithm and these algorithms. Their theoretical analysis is based on the work in (Ghashami et al. (2016a); Liberty (2013)), the bound for singular values of standard Gaussian matrices and the spectral error of approximate matrix multiplication (cf. Cohen et al. (2016)).

By using the random embedding matrix obtained by multiplying a SpEmb matrix and a SRHT matrix with appropriate dimensions, a sparse randomized SVD algorithm for low-rank matrix approximation is presented in (Clarkson and Woodruff (2017)). By using a SpEmb matrix combined with FFT-based random projections, Aizenbud et al. (2016) presented a fast randomized algorithm for approximating a low-rank LU decomposition. Numerical examples illustrate that this algorithm is faster than the sparse randomized SVD algorithm and SRHT-based Randomized LU (see Shabat et al. (2018)). A sub-Gaussian-based Randomized SVD algorithm is proposed by Aizenbud and Averbuch (2019), which utilizes the product of a standard Gaussian matrix and a sub-Gaussian matrix to project the original matrix. Although the proven error bound is not as tight as the state-of-the-art bound, experiments show that the proposed algorithm is faster in practice, while getting the same error rates as the sparse randomized SVD algorithm and FFT-based randomized SVD (see Woolfe et al. (2008)). Hence, **our second goal** is to use two mixed random projection techniques on each data batch and to perform the FD algorithm on the smaller matrix. The first one is SpEmb combined with standard Gaussian matrices and the second one is the combination of SpEmb and SRHT. When the number of blocks q is equal to 1, Algorithm 2 is the same as the randomized SVD algorithms based on some random projection/sampling matrices. The standard randomized SVD algorithms are suitable for the case in which the singular values of any matrix decay fast. When the singular values of the matrix slowly decay, these randomized SVD algorithms will be modified by combining the strategy of subspace iterations to achieve a suitable accuracy.

As a higher-order generalization of vectors and matrices, tensors are used in a wide range of applications in signal processing, data mining, machine learning, latent semantic indexing, and other fields. Tensor decomposition is important for processing and compressing tensors. Common types of tensor decomposition include CANDECOMP/PARAFAC (CP) decomposition, Tucker decomposition, Hierarchical Tucker decomposition and TT decomposition. The interested readers can refer to (Cichocki et al. (2016, 2017); Grasedyck et al. (2013); Kolda and Bader (2009)) for details. In particular, Tucker decomposition can be applied in handwritten digit classification (cf. Savas and Eldén (2007)), multilinear subspace analysis (cf. Vasilescu and Terzopoulos (2002, 2003)), item recommendation (cf. Symeonidis (2015)) and low Tucker-rank tensor completion and/or recovery (cf. Kressner et al. (2014a); Rauhut et al. (2017)). One of the advantages for TT decomposition is that its storage cost increases linearly with the order of tensors. The approximate TT decomposition can be used in many practical applications, such as the system of linear equations (cf. Oseledets and Dolgov (2012)), eigenvalue and singular value problems (cf. Dolgov et al. (2014); Kressner et al. (2014b)), tensor completion (cf. Kressner et al. (2014a)), supervised tensor learning (cf. Chen et al. (2022); Kour et al. (2023)). In this paper, we focus on the approximation of Tucker decomposition with a given Tucker-rank and TT decomposition with a given TT-rank.

For a given Tucker-rank (its definition is given in Section 4), several common algorithms for Tucker decomposition are the alternating least squares (ALS) method (Tucker-ALS, see De Lathauwer et al. (2000b)), the truncated higher-order singular value decomposition (T-HOSVD, see (De Lathauwer et al. (2000b); Kolda and Bader (2009))) and the sequentially T-HOSVD (ST-HOSVD, see (Vannieuwenhoven et al. (2012))). The ALS method, called the higher-order orthogonal iteration (HOOI), updates one of the factor matrices along with the core tensor at a time, and is bottlenecked by the operation called the tensor times matrix-chain (TTMc). Several randomized variants for the ALS method are discussed in (Caiafa and Cichocki (2010); Malik and Becker (2018); Ma and Solomonik (2021); Tsourakakis (2010)). In the framework of T-HOSVD and/or ST-HOSVD, for each n, the main calculation for T-HOSVD and ST-HOSVD is to compute an exact SVD of the mode-n unfolding of the corresponding tensor. By using randomized SVD with the random projection and/or sampling techniques, different randomized variants of T-HOSVD and ST-HOSVD are presented recently, for example, see (Ahmadi-Asl et al. (2021); Che and Wei (2019); Che et al. (2020, 2021a, 2025a, 2021b); Minster et al. (2020, 2024); Sun et al. (2020); Zhou et al. (2014)). TT-SVD (cf. Oseledets and Tyrtyshnikov (2010)) and TT-cross (cf. Savostyanov and Oseledets (2011)) are two common algorithms for computing the approximate TT decomposition. For a given TT-rank, many scholars focus on the randomized variants of TT-SVD (see Che et al. (2026); Huber et al. (2017); Li et al. (2022); Shi et al. (2023)). Several adaptive randomized algorithms for the approximate TT decomposition are studied in (Alger et al. (2020); Che and Wei (2019); Che et al. (2026)).

So far, there is no literature based on the FD algorithm and its randomized variants for finding an approximation of Tucker (resp. TT) decomposition with a given Tucker (resp. TT) rank. Hence, with a given Tucker/TT rank, **our third goal** is to design efficient FD-based randomized algorithms for obtaining the approximate Tucker/TT decomposition of any tensor, based on the randomized variants of the FD algorithm for the low-rank matrix approximation. Numerical examples illustrate that the proposed algorithms are faster in practice, while getting the same error rates as the existing algorithms.

1.2 Notations and definitions

We use lower case letters (e.g. x, u, v) for scalars, lower case bold letters (e.g. $\mathbf{x}, \mathbf{u}, \mathbf{v}$) for vectors, capital letters (e.g., I, J, K) for positive integers, bold capital letters (e.g. $\mathbf{A}, \mathbf{B}, \mathbf{C}$) for matrices, and calligraphic letters (e.g. $\mathcal{A}, \mathcal{B}, \mathcal{C}$) for tensors. This notation is consistently used for the lower-order parts of a given structure. For example, the entry with row index i and column index j in a matrix $\mathbf{A}$, i.e., $(\mathbf{A})_{ij}$, is represented as a_{ij} (also $(\mathbf{x})_i = x_i$ and $(\mathcal{A})_{i_1 i_2 \dots i_N} = a_{i_1 i_2 \dots i_N}$). An N-th order real-valued tensor is denoted by $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$

such that $a_{i_1 i_2 \dots i_N} \in \mathbb{R}$ with $i_n = 1, 2, \dots, I_n$ and $n = 1, 2, \dots, N$. For specific, $\mathcal{A}$ is any number in $\mathbb{R}$ with $N = 0$, $\mathcal{A}$ is any vector in $\mathbb{R}^{I_1}$ with $N = 1$, and $\mathcal{A}$ is any matrix in $\mathbb{R}^{I_1 \times I_2}$ with $N = 2$.

The symbols $\mathbf{A}^\top$, $\|\mathbf{A}\|_F$, and $\|\mathbf{A}\|_2$, respectively, denote the transpose, the Frobenius norm and the spectral norm of $\mathbf{A} \in \mathbb{R}^{I \times J}$. We use $\mathbf{I}_J$ to denote the identity matrix in $\mathbb{R}^{J \times J}$. An orthogonal projection $\mathbf{P} \in \mathbb{R}^{I \times I}$ is defined as $\mathbf{P}^2 = \mathbf{P}$ and $\mathbf{P}^\top = \mathbf{P}$. A matrix $\mathbf{Q} \in \mathbb{R}^{I \times K}$ with $I > K$ is orthonormal if $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}_K$. We use the interpretation of $\Theta(\cdot)$ to refer to the class of functions the growth of which is bounded above and below up to a constant. The term "for each n" means $n = 1, 2, \dots, N$, where $N > 1$ is a given integer. The term "for a given n" means the integer n satisfies $1 \leq n \leq N$. For two matrices $\mathbf{A} \in \mathbb{R}^{I_1 \times J_1}$ and $\mathbf{B} \in \mathbb{R}^{I_2 \times J_2}$, the Kronecker product $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{I_1 I_2 \times J_1 J_2}$ is given by

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1J_1}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{2J_1}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I_1 1}\mathbf{B} & a_{I_1 2}\mathbf{B} & \dots & a_{I_1 J_1}\mathbf{B} \end{pmatrix}.$$

A standard Gaussian matrix $\mathbf{G} \in \mathbb{R}^{I \times L}$ (see Definition 2.1 in Tropp et al. (2017)) is defined if its entries form an independent family of standard normal variables (i.e., Gaussian with mean one and variance one). Let $I = 2^p$ for some integer $p > 0$. A SRHT matrix $\mathbf{\Omega} \in \mathbb{R}^{I \times L}$ (see Boutsidis and Gittens (2013); Chen et al. (2017)) is a random matrix with the structure $\mathbf{\Omega} = \sqrt{I/L}\mathbf{DHS}$, where $\mathbf{D}$ is an $I \times I$ diagonal matrix whose entries are independent and identically distributed (i.i.d.) random variables drawn from a uniform distribution on $\{\pm 1\}$, $\mathbf{H}$ is an $I \times I$ Walsh-Hadamard matrix, scaled by $1/\sqrt{I}$ so it is an orthogonal matrix and $\mathbf{S}$ is an $I \times L$ matrix that restricts an I-dimensional vectors to L coordinates, chosen uniformly at random.

Remark 1 *In the definition of a SRHT matrix, when we do not restrict that I is a power of 2 and let the matrix $\mathbf{H}$ as a normalized discrete cosine transform matrix, then we call $\mathbf{\Omega} = \sqrt{I/L}\mathbf{DHS}$ as a SRDCT (subsampled randomized discrete cosine transform) matrix.*

Finally, we introduce the definition of a sparse subspace embedding (SpEmb) matrix (cf. Aizenbud et al. (2016); Clarkson and Woodruff (2017); Meng and Mahoney (2013); Nelson and Nguyên (2013)), that has received much attention due to its efficient time complexity. Consider the random linear map $\mathbf{S} = \mathbf{D\Phi} \in \mathbb{R}^{I \times L}$, such that for $h : \{1, 2, \dots, I\} \to \{1, 2, \dots, L\}$, a random map such that for each $i \in \{1, 2, \dots, I\}$, $h(i) = l$ for $l \in \{1, 2, \dots, L\}$ with probability $1/L$, we have

(a) the matrix $\mathbf{\Phi} \in \{0, 1\}^{I \times L}$ is a binary matrix with nonzero entries $\mathbf{\Phi}_{ih(i)} = 1$ and all the remaining entries are equal to 0;

(b) the matrix $\mathbf{D}$ is an $I \times I$ random diagonal matrix where each diagonal entry is independently chosen to be either $+1$ or -1 with equal probability.

By the definition of any SpEmb matrix, for a given matrix $\mathbf{A} \in \mathbb{R}^{J \times I}$ and a SpEmb matrix $\mathbf{S} \in \mathbb{R}^{I \times L}$, the cost to get $\mathbf{AS}$ is $\Theta(\mathrm{nnz}(\mathbf{A}))$ operations, which is superior to the costs of $\Theta(IJL)$ for standard Gaussian matrices and $\Theta(IJ\log(L))$ for SRHT matrices (cf. Boutsidis and Gittens (2013)), where $\mathrm{nnz}(\mathbf{A})$ is the number of nonzero elements in $\mathbf{A}$. Thus it is well suitable for sparse matrices.

1.3 Organizations

The rest of the paper is organized as follows. In Section 2, we introduce the general form and randomized variants of the FD algorithm. In Section 3, we propose our randomized variants of the FD algorithm based on the mixed random projection techniques and give their theoretical analysis. In Section 4, two efficient algorithms for approximate Tucker decomposition are presented by combining the proposed algorithms with T-HOSVD and ST-HOSVD. We illustrate our algorithms via numerical examples in Section 5 and study the capabilities of our algorithms for classification tasks in Section 6. We conclude this paper in Section 7.

2. Frequent directions and the randomized variants

We now review the standard FD algorithm, which extends the idea of frequent items for item frequency approximation problem to a general matrix. It is worthy noting that the standard FD algorithm is deterministic, space efficient and produces more accurate estimates given a fixed sketch size. Suppose that $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ with $I_1 < I_2$. For a given parameter $L \leq I_1$, at first, the algorithm considers the first $2L$ columns in $\mathbf{A}$ and shrinks its top L left orthogonal vectors by the same amount to obtain an $I_1 \times L$ matrix; then combines them with the next L columns in $\mathbf{A}$ for the next iteration, and repeats the procedure. We summarize the detail process as in Algorithm 1.

Algorithm 1 Frequent directions (FD) algorithm for low-rank matrix approximation (cf. Ghashami et al. (2016b); Teng and Chu (2018))

```
Input: A matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$, the desired rank parameter $\mu$ and the sketch size $L$ with $\mu < L$.
Output: $\widetilde{\mathbf{U}}_\mu \in \mathbb{R}^{I_1 \times \mu}$, $\widetilde{\mathbf{\Sigma}}_\mu \in \mathbb{R}^{\mu \times \mu}$, $\widetilde{\mathbf{V}}_\mu \in \mathbb{R}^{I_2 \times \mu}$ such that $\mathbf{A} \approx \widetilde{\mathbf{A}}_\mu = \widetilde{\mathbf{U}}_\mu \widetilde{\mathbf{\Sigma}}_\mu \widetilde{\mathbf{V}}_\mu^\top$.
1: Let $\mathbf{B} \in \mathbb{R}^{I_1 \times 2L}$ be the zero matrix and set $\mathbf{B}(:, 1:L) = \mathbf{A}(:, 1:L)$.
2: Append zero columns to $\mathbf{A}$ such that the number of the columns in $\mathbf{A}$ is a multiple of $L$.
3: for $i = 1, 2, \ldots, \lceil I_2/L \rceil - 1$ do
4:     Let $\mathbf{B}(:, L+1:2L) = \mathbf{A}(:, iL+1:(i+1)L)$.
5:     Compute the SVD of $\mathbf{B}$ as $\mathbf{B} = \mathbf{U_B}\mathbf{\Sigma_B}\mathbf{V_B}^\top$ with $\mathbf{U_B} \in \mathbb{R}^{I_1 \times 2L}$, $\mathbf{\Sigma_B} \in \mathbb{R}^{2L \times 2L}$ and $\mathbf{V_B} \in \mathbb{R}^{2L \times 2L}$.
6:     Set $\delta_i = \sigma_{L+1}(\mathbf{B})^2$ and $\mathbf{B} = \mathbf{U_B}\sqrt{\max(\mathbf{\Sigma_B}^2 - \delta_i \mathbf{I}_{2L}, \mathbf{0}_{2L \times 2L})}$.
7: end for
8: Set $\mathbf{B} = \mathbf{B}(:, 1:L)$ and $\mathbf{U_B} = \mathbf{U_B}(:, 1:L)$.
9: Compute the SVD of $\mathbf{U_B}^\top\mathbf{A}$ as $\mathbf{U_B}^\top\mathbf{A} = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^\top$ with $\mathbf{U}_1 \in \mathbb{R}^{L \times L}$, $\mathbf{\Sigma}_1 \in \mathbb{R}^{L \times L}$ and $\mathbf{V}_1 \in \mathbb{R}^{I_2 \times L}$.
10: Form $\widetilde{\mathbf{A}}_\mu = \widetilde{\mathbf{U}}_\mu \widetilde{\mathbf{\Sigma}}_\mu \widetilde{\mathbf{V}}_\mu^\top$, where $\widetilde{\mathbf{U}}_\mu = \mathbf{U_B}\mathbf{U}_1(:, 1:\mu)$, $\widetilde{\mathbf{\Sigma}}_\mu = \mathbf{\Sigma}_1(1:\mu, 1:\mu)$ and $\widetilde{\mathbf{V}}_\mu = \mathbf{V}_1(:, 1:\mu)$.
```

Remark 2 Ghashami et al. (2016b) *and* Liberty (2013) *presented the original form for the FD algorithm, which is different from Algorithm* 1. *However, this is to maintain a sketch*

$\mathbf{B} \in \mathbb{R}^{I_1 \times L}$ *and to acquire a matrix* $\mathbf{U_B}$ *whose columns form an orthonormal basis for the column space of* $\mathbf{B}$.

Similar to the work in (Teng and Chu (2018)), we overlook the computational complexity related to the diagonal matrices $\mathbf{\Sigma_B}$ and $\widetilde{\mathbf{\Sigma}}_\mu$. Hence, when $\mathbf{A}$ is sparse, the Algorithm 1 needs $24I_1I_2L + 2\mathrm{nnz}(\mathbf{A})L + 166I_2L^2 + 2I_1(I_2+L)\mu - (24I_2 + 140L)L^2$ operations, where $\mathrm{nnz}(\mathbf{A})$ is the number of nonzero entries in $\mathbf{A}$; when $\mathbf{A}$ is dense, Algorithm 1 requires $26I_1I_2L + 166I_2L^2 + 2I_1(I_2+L)\mu - (24I_2 + 140L)L^2$ operations. It is worthy noting that Algorithm 1 is not easily parallelizable because its resulting sketch matrices constitute mergeable summaries.

There have $(\lceil I_2/L \rceil - 1)$ iterations in Algorithm 1 and each iteration needs $\Theta(I_1L^2)$ running time to compute the SVD of $\mathbf{B}$[1]. When I_2 is extremely large, it is difficult to use Algorithm 1 for computing the low-rank matrix approximation. To reduce the iterations of Algorithm 1, there exist several randomized variants for the FD algorithm for the low-rank approximation of a matrix, and their framework is summarized in Algorithm 2.

Algorithm 2 The framework for the fast FD algorithm

```
    Input: A matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$, the desired rank parameter $\mu$, the sketch size $L$ and the number of blocks $q$ (assume $I_2$ is a multiple of $q$) with $\mu < L$.
    Output: $\widetilde{\mathbf{U}}_\mu \in \mathbb{R}^{I_1 \times \mu}$, $\widetilde{\mathbf{\Sigma}}_\mu \in \mathbb{R}^{\mu \times \mu}$, $\widetilde{\mathbf{V}}_\mu \in \mathbb{R}^{I_2 \times \mu}$ such that $\mathbf{A} \approx \widetilde{\mathbf{A}}_\mu = \widetilde{\mathbf{U}}_\mu \widetilde{\mathbf{\Sigma}}_\mu \widetilde{\mathbf{V}}_\mu^\top$.
 1: Update the matrix $\mathbf{A}$ as $\mathbf{A} = \mathbf{AP}$, where $\mathbf{P} \in \mathbb{R}^{I_2 \times I_2}$ is a random permutation matrix.
 2: Let $\mathbf{B} \in \mathbb{R}^{I_1 \times 2L}$ be the zero matrix.
 3: Choose a matrix $\mathbf{G}_1 \in \mathbb{R}^{I_2/q \times L}$ and compute $\mathbf{B}(:, 1:L) = \mathbf{A}(:, 1:I_2/q)\mathbf{G}_1$.
 4: for $i = 1, 2, \dots, q-1$ do
 5:     Choose a matrix $\mathbf{G}_{i+1} \in \mathbb{R}^{I_2/q \times L}$ and compute $\mathbf{B}(:, L+1:2L) = \mathbf{A}(:, iI_2/q + 1 : (i+1)I_2/q)\mathbf{G}_{i+1}$.
 6:     Compute the SVD of $\mathbf{B}$ as $\mathbf{B} = \mathbf{U_B\Sigma_BV_B^\top}$ with $\mathbf{U_B} \in \mathbb{R}^{I_1 \times 2L}$, $\mathbf{\Sigma_B} \in \mathbb{R}^{2L \times 2L}$ and $\mathbf{V_B} \in \mathbb{R}^{2L \times 2L}$.
 7:     Set $\delta_i = \sigma_{L+1}(\mathbf{B})^2$ and $\mathbf{B} = \mathbf{U_B}\sqrt{\max(\mathbf{\Sigma_B^2} - \delta_i\mathbf{I}_{2L}, \mathbf{0}_{2L \times 2L})}$.
 8: end for
 9: Set $\mathbf{B} = \mathbf{B}(:, 1:L)$ and $\mathbf{U_B} = \mathbf{U_B}(:, 1:L)$.
10: Compute the SVD of $\mathbf{U_B^\top A}$ as $\mathbf{U_B^\top A} = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^\top$ with $\mathbf{U}_1 \in \mathbb{R}^{L \times L}$, $\mathbf{\Sigma}_1 \in \mathbb{R}^{L \times L}$ and $\mathbf{V}_1 \in \mathbb{R}^{I_2 \times L}$.
11: Form $\widetilde{\mathbf{A}}_\mu = \widetilde{\mathbf{U}}_\mu \widetilde{\mathbf{\Sigma}}_\mu \widetilde{\mathbf{V}}_\mu^\top$, where $\widetilde{\mathbf{U}}_\mu = \mathbf{U_B}\mathbf{U}_1(:, 1:\mu)$, $\widetilde{\mathbf{\Sigma}}_\mu = \mathbf{\Sigma}_1(1:\mu, 1:\mu)$ and $\widetilde{\mathbf{V}}_\mu = \mathbf{V}_1(:, 1:\mu)$.
```

The choice of $\{\mathbf{G}_1, \mathbf{G}_2, \dots, \mathbf{G}_q\}$ in Algorithm 2 leads to the arithmetic costs for sketching and updates, and the upper bound for $\|\mathbf{A} - \widetilde{\mathbf{A}}_\mu\|_F$. In general, each $\mathbf{G}_i$ can be a standard Gaussian matrix, a SpEmb matrix (Teng and Chu (2018)), a SRHT matrix (Chen et al. (2017)) or a SRFT (Subsampled Random Fourier Transform) matrix (Woolfe et al. (2008)). We also list two sophisticated ways to generate the matrix $\mathbf{G}_i$, given as follows:

(a) As shown in (Ghashami et al. (2016a)), each $\mathbf{G}_i$ can be obtained based on the following three steps:

1. The symbol $\lceil x \rceil$ is the smallest integer that is larger than or equal to any $x \in \mathbb{R}$.

1) to generate a standard Gaussian matrix $\mathbf{\Omega} \in \mathbb{R}^{L\times I_1}$;
2) to compute the matrix $\mathbf{Y} = \mathbf{\Omega}\mathbf{A}_i(\mathbf{A}_i^\top\mathbf{A}_i)^s$, where $\mathbf{A}_i = \mathbf{A}(:, iI_2/q{+}1 : (i{+}1)I_2/q)$ and $s > 0$ is an integer; and
3) to construct the matrix $\mathbf{G}_i$ whose columns form an orthogonal basis for the row space of $\mathbf{Y}$.

(b) From (Wang et al. (2023)), each $\mathbf{G}_i$ is computed as:

i) to generate a standard Gaussian matrix or a SpEmb matrix $\boldsymbol{G}' \in \mathbb{R}^{L\times I_1}$;
ii) to compute the matrix $\mathbf{Y} = [\boldsymbol{G}'\mathbf{A}_i; \boldsymbol{G}'\mathbf{A}_i(\mathbf{A}_i^\top\mathbf{A}_i); \dots; \boldsymbol{G}'\mathbf{A}_i(\mathbf{A}_i^\top\mathbf{A}_i)^s]$, where $\mathbf{A}_i = \mathbf{A}(:, iI_2/q + 1 : (i + 1)I_2/q)$ and $s > 0$ is an integer;
iii) to orthonormalize the rows of $\mathbf{Y}$ to obtain $\mathbf{Q} \in \mathbb{R}^{I_2/q\times(s+1)L}$;
iv) to compute the matrix $\mathbf{M} = \mathbf{Q}^\top\mathbf{A}_i^\top\mathbf{A}_i\mathbf{Q} \in \mathbb{R}^{(s+1)L\times(s+1)L}$;
v) to set $\mathbf{P}_L$ to the top L singular vectors of $\mathbf{M}$; and
vi) to form $\mathbf{G}_i = \mathbf{Q}\mathbf{P}_L$.

We now consider the computational complexity of Algorithm 2 based on several choices of each $\mathbf{G}_i$. According to (Teng and Chu (2018)), when the matrices $\mathbf{G}_i$ are SpEmb matrices, Algorithm 2 needs $\Theta(I_2 + \mathrm{nnz}(\mathbf{A})) + 2\mathrm{nnz}(\mathbf{A})L + 6I_2L^2 + 2I_1(I_2 + L)\mu + 24I_1(q - 1)L^2 + (160q - 140)L^3$ operations to calculate the rank-μ approximation of $\mathbf{A}$. When the matrices $\mathbf{G}_i$ are standard Gaussian matrices, Algorithm 2 costs $\Theta(I_2L) + 4I_1I_2L + 6I_2L^2 + 2I_1(I_2 + L)\mu + 24I_1(q-1)L^2 + (160q-140)L^3$ operations. For the case of SRHT matrices, Algorithm 2 requires $\Theta(I_1I_2 \log L) + 2I_1I_2L + 6I_2L^2 + 2I_1(I_2 + L)\mu + 24I_1(q - 1)L^2 + (160q - 140)L^3$ operations.

Remark 3 *To ensure that Steps* 6 *and* 7 *in Algorithm* 2 *are meaningful, we need to assume that* $2L \leq I_1$. *Let* $\widetilde{\mathbf{A}}_\mu$ *be obtained by applying Algorithm* 2 *to the matrix* $\mathbf{A}$ *with* μ, L *and* q. *For the upper bound of* $\|\mathbf{A} - \widetilde{\mathbf{A}}_\mu\|_2$, *the case of all the matrices* $\mathbf{G}_i$ *being SpEmb matrices is considered in* (Teng and Chu (2018)) *and the case of all the matrices* $\mathbf{G}_i$ *being SRHT matrices is studied in* (Chen et al. (2017)). *We will derive the theoretical result for the case of all the matrices* $\mathbf{G}_i$ *being standard Gaussian matrices, as shown in Theorem* 10.

Remark 4 *Note that Algorithm* 2 *works well for matrices whose singular values exhibit some decay, but may produce a poor basis when the input matrix has a flat singular spectrum. To overcome this issue, we can introduce the power scheme into this algorithm. In detail, we compute* $\mathbf{B}(:, 1 : L) = \mathbf{A}_1(\mathbf{A}_1^\top\mathbf{A}_1)^s\mathbf{G}_1$ *and* $\mathbf{B}(:, L + 1 : 2L) = \mathbf{A}_i(\mathbf{A}_i^\top\mathbf{A}_i)^s\mathbf{G}_{i+1}$, *where* $\mathbf{A}_i = \mathbf{A}(:, iI_2/q + 1 : (i + 1)I_2/q)$ *and* $s > 0$ *is a given integer.*

3. The proposed work

In the framework of the FD algorithm, by combining different random projection strategies, we derive another efficient randomized algorithm for low-rank matrix approximations. We also consider their computational complexity and theoretical results based on different combinations.

3.1 Framework of the algorithm

Suppose that $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ with $I_1 < I_2$. Let L and q be positive integers with $L \leq I_1$ and I_2 being a multiple of q. We list the process of Algorithm 3 as follows,

(1) we update the matrix $\mathbf{A}$ as $\mathbf{A} = \mathbf{AP}$, where $\mathbf{P} \in \mathbb{R}^{I_2 \times I_2}$ is a random permutation matrix;

(2) we use the product of a SpEmb matrix and a random matrix to project the matrix $\mathbf{A}(:, 1 : I_2/q)$ and then form the first L columns of the matrix $\mathbf{B} \in \mathbb{R}^{I_1 \times 2L}$;

(3) for the first iteration, we use the product of a SpEmb matrix and a random matrix to project the matrix $\mathbf{A}(:, I_2/q + 1 : 2I_2/q)$, form the last L columns of $\mathbf{B}$ of size $I_1 \times 2L$ and update the matrix $\mathbf{B}$ as $\mathbf{B} = \mathbf{U_B}\sqrt{\max(\mathbf{\Sigma}_\mathbf{B}^2 - \delta_1 \mathbf{I}_{2L}, \mathbf{0}_{2L \times 2L})}$ with $\mathbf{\Sigma_B} = \mathrm{diag}(\sigma_1(\mathbf{B}), \dots, \sigma_{2L}(\mathbf{B}))$ and $\delta_1 = \sigma_{L+1}(\mathbf{B})$;

(4) for the next iteration, the last L columns of $\mathbf{B}$ is obtained by applying the product of a SpEmb matrix and a random matrix to project the matrix $\mathbf{A}(:, iI_2/q+1 : (i+1)I_2/q)$, and we repeat the above procedure.

In Algorithm 3, the random permutation matrix $\mathbf{P} \in \mathbb{R}^{I_2 \times I_2}$ is generated by the function "randperm" in MATLAB and constructed via Fisher-Yates-Knuth shuffle: for each $\mathbf{e}_i$, i.e., the ith column of the identity matrix $\mathbf{I}_{I_2}$, with $i = 1, 2, \dots, I_2$, the probability of $\mathbf{Pe}_i = \mathbf{e}_j$ is $1/I_2$ with $j = 1, 2, \dots, I_2$.

Similar to Remark 4, when the input matrix has a flat singular spectrum, we introduce the power scheme into this algorithm: we compute $\mathbf{B}(:, 1 : L) = \mathbf{A}_1(\mathbf{A}_1^\top \mathbf{A}_1)^s \mathbf{S}_1 \mathbf{G}_1$ and $\mathbf{B}(:, L+1 : 2L) = \mathbf{A}_i(\mathbf{A}_i^\top \mathbf{A}_i)^s \mathbf{S}_{i+1} \mathbf{G}_{i+1}$ with $i = 1, 2, \dots, q-1$, where $\mathbf{A}_i = \mathbf{A}(:, iI_2/q + 1 : (i+1)I_2/q)$ and $s > 0$ is a given integer. We now analyze the computational complexity of Algorithm 3. When $\mathbf{A}$ is dense, for each step in Algorithm 3, we have

(1) to generate all the matrices $\mathbf{S}_i$ and $\mathbf{G}_i$ needs $q \cdot \Theta(I_2/q + L'L) = \Theta(I_2 + qL'L)$ operations;

(2) it costs $q \cdot \Theta(I_1 I_2/q) + q \cdot 2I_1 L'L = \Theta(I_1 I_2) + 2qI_1 L'L$ operations to compute $\mathbf{B}(:, 1 : L) = \mathbf{A}_1 \mathbf{S}_1 \mathbf{G}_1$ and to update $\mathbf{B}(:, L+1 : 2L) = \mathbf{A}_i \mathbf{S}_{i+1} \mathbf{G}_{i+1}$ with $i = 1, 2, \dots, q-1$;

(3) to form the SVD of $\mathbf{B}$ requires $(q-1)(24I_1 L^2 + 160L^3)$ operations;

(4) it demands $2I_1 I_2 L + 6I_2 L^2 + 20L^3 + 2I_1(I_2 + L)\mu$ operations to obtain the matrix $\widetilde{\mathbf{A}}_\mu$.

Then, Algorithm 3 with SpEmb+Gaussian needs

$$\begin{aligned}&\Theta(I_2 + qL'L + I_1 I_2) + 2qI_1 L'L + 2I_1 I_2 L \\ &+ 6I_2 L^2 + 2I_1(I_2 + L)\mu + 24I_1(q-1)L^2 + (160q - 140)L^3\end{aligned}$$

operations to obtain a rank-μ approximation $\widetilde{\mathbf{A}}_\mu$ to $\mathbf{A}$.

Remark 5 *When* $\mathbf{A}$ *is sparse, it costs* $\Theta(\mathrm{nnz}(\mathbf{A})) + 2qI_1 L'L$ *operations to compute* $\mathbf{B}(:, 1 : L) = \mathbf{A}(:, 1 : I_2/q)\mathbf{S}_1 \mathbf{G}_1$ *and* $\mathbf{B}(:, L+1 : 2L) = \mathbf{A}(:, iI_2/q+1 : (i+1)I_2/q)\mathbf{S}_{i+1}\mathbf{G}_{i+1}$ *with* $i =$

Algorithm 3 Another efficient FD algorithm with the mixed strategy

```
    Input: A matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$, the desired rank $\mu$, two positive integers $L$ and $L'$ with $L' > L$, and the number of blocks $q$ (assume $I_2$ is a multiple of $q$).
    Output: $\widetilde{\mathbf{U}}_\mu \in \mathbb{R}^{I_1 \times \mu}$, $\widetilde{\mathbf{\Sigma}}_\mu \in \mathbb{R}^{\mu \times \mu}$, $\widetilde{\mathbf{V}}_\mu \in \mathbb{R}^{I_2 \times \mu}$ such that $\mathbf{A} \approx \widetilde{\mathbf{A}}_\mu = \widetilde{\mathbf{U}}_\mu \widetilde{\mathbf{\Sigma}}_\mu \widetilde{\mathbf{V}}_\mu^\top$.
 1: Update the matrix $\mathbf{A}$ as $\mathbf{A} = \mathbf{AP}$, where $\mathbf{P} \in \mathbb{R}^{I_2 \times I_2}$ is a random permutation matrix.
 2: Let $\mathbf{B} \in \mathbb{R}^{I_1 \times 2L}$ be the zero matrix.
 3: Choose a SpEmb matrix $\mathbf{S}_1 \in \mathbb{R}^{I_2/q \times L'}$ and a standard Gaussian matrix $\mathbf{G}_1 \in \mathbb{R}^{L' \times L}$.
 4: Compute $\mathbf{B}(:, 1:L) = \mathbf{A}(:, 1:I_2/q)\mathbf{S}_1\mathbf{G}_1$.
 5: for $i = 1, 2, \dots, q-1$ do
 6:     Choose a SpEmb matrix $\mathbf{S}_{i+1} \in \mathbb{R}^{I_2/q \times L'}$ and a standard Gaussian matrix $\mathbf{G}_{i+1} \in \mathbb{R}^{L' \times L}$.
 7:     Compute $\mathbf{B}(:, L+1:2L) = \mathbf{A}(:, iI_2/q+1:(i+1)I_2/q)\mathbf{S}_{i+1}\mathbf{G}_{i+1}$.
 8:     Compute the SVD of $\mathbf{B}$ as $\mathbf{B} = \mathbf{U_B \Sigma_B V_B^\top}$ with $\mathbf{U_B} \in \mathbb{R}^{I_1 \times 2L}$, $\mathbf{\Sigma_B} \in \mathbb{R}^{2L \times 2L}$ and $\mathbf{V_B} \in \mathbb{R}^{2L \times 2L}$.
 9:     Set $\delta_i = \sigma_{L+1}(\mathbf{B})^2$ and $\mathbf{B} = \mathbf{U_B}\sqrt{\max(\mathbf{\Sigma_B^2} - \delta_i \mathbf{I}_{2L}, \mathbf{0}_{2L \times 2L})}$.
10: end for
11: Set $\mathbf{B} = \mathbf{B}(:, 1:L)$ and $\mathbf{U_B} = \mathbf{U_B}(:, 1:L)$.
12: Compute the SVD of $\mathbf{U_B^\top A}$ as $\mathbf{U_B^\top A} = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^\top$ with $\mathbf{U}_1 \in \mathbb{R}^{L \times L}$, $\mathbf{\Sigma}_1 \in \mathbb{R}^{L \times L}$ and $\mathbf{V}_1 \in \mathbb{R}^{I_2 \times L}$.
13: Form $\widetilde{\mathbf{A}}_\mu = \widetilde{\mathbf{U}}_\mu \widetilde{\mathbf{\Sigma}}_\mu \widetilde{\mathbf{V}}_\mu^\top$, where $\widetilde{\mathbf{U}}_\mu = \mathbf{U_B U}_1(:, 1:\mu)$, $\widetilde{\mathbf{\Sigma}}_\mu = \mathbf{\Sigma}_1(1:\mu, 1:\mu)$ and $\widetilde{\mathbf{V}}_\mu = \mathbf{V}_1(:, 1:\mu)$.
```

$1, 2, \dots, q-1$*; and it to obtain the matrix* $\widetilde{\mathbf{A}}_\mu$ *needs to* $2\mathrm{nnz}(\mathbf{A})L + 6I_2L^2 + 20L^3 + 2I_1(I_2+L)\mu$ *operations. Then for a sparse matrix* $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$*, Algorithm* 3 *with SpEmb+Gaussian needs*

$$\Theta(I_2 + qL'L + \mathrm{nnz}(\mathbf{A})) + 2qI_1L'L + 2\mathrm{nnz}(\mathbf{A})L \\ + 6I_2L^2 + 2I_1(I_2+L)\mu + 24I_1(q-1)L^2 + (160q-140)L^3$$

operations.

3.2 Theoretical analysis

Let $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ and $\mathbf{B} \in \mathbb{R}^{I_1 \times L}$ with $L < I_1 < I_2$. For a given positive integer $\mu < L$, we use $\pi_{\mathbf{B}}^{\mu}(\mathbf{A})$ to denote the projection of $\mathbf{A}$ on the top-μ singular vectors of $\mathbf{B}$, i.e., $\pi_{\mathbf{B}}^{\mu}(\mathbf{A}) = \mathbf{UU}^\top\mathbf{A}$, where the columns of $\mathbf{U}$ are the top-μ left singular vectors of $\mathbf{B}$. For any $0 < \alpha < 1$ and integer $0 \leq \mu \leq \mathrm{rank}(\mathbf{A})$, if $\|\mathbf{AA}^\top - \mathbf{BB}^\top\|_2 \leq \alpha\|\mathbf{A} - \mathbf{A}_\mu\|_F$, we call $\mathbf{B}$ as an (α, μ)-cov-sketch (Huang (2019)) of $\mathbf{A}$.

Lemma 6 ((Huang, 2019, Lemma 3)) *Let* $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ *and* $\mathbf{A} \in \mathbb{R}^{I_1 \times L}$ *with* $L < \min\{I_1, I_2\}$*. For a given positive integer* $\mu < L$*, the projection error of* $\mathbf{B}$ *with respect to* $\mathbf{A}$ *is defined as* $\|\mathbf{A} - \pi_{\mathbf{B}}^{\mu}(\mathbf{A})\|_F$*. Then, one has*

$$\|\mathbf{A} - \pi_{\mathbf{B}}^{\mu}(\mathbf{A})\|_F^2 \leq \|\mathbf{A} - \mathbf{A}_\mu\|_F^2 + 2\mu\|\mathbf{AA}^\top - \mathbf{BB}^\top\|_2,$$

where $\mathbf{A}_\mu$ *is the best rank-*μ *approximation of* $\mathbf{A}$*.*

Let $\widetilde{\mathbf{A}}_\mu$ and $\mathbf{B}$ be obtained from Algorithm 1, 2 or 3. It is clear to see that $\|\mathbf{A}-\widetilde{\mathbf{A}}_\mu\|_F = \|\mathbf{A}-\pi_{\mathbf{B}}^{\mu}(\mathbf{A})\|_F$. Hence, we have

$$\|\mathbf{A}-\widetilde{\mathbf{A}}_\mu\|_F^2 \leq \|\mathbf{A}-\mathbf{A}_\mu\|_F^2 + 2\mu\|\mathbf{A}\mathbf{A}^\top - \mathbf{B}\mathbf{B}^\top\|_2. \tag{1}$$

From (1), when we consider the upper bound of $\|\mathbf{A}-\widetilde{\mathbf{A}}_\mu\|_F^2$, we need to give the bounds for $\|\mathbf{A}\mathbf{A}^\top - \mathbf{B}\mathbf{B}^\top\|_2$.

Theorem 7 *For any $\epsilon,\delta,\tau \in (0,1/2)$, let q, I_1, I_2, L' and L be positive integers such that $L < L' < I_2$, $I_1 < I_2$ and*

$$qL' \geq \frac{12(I_1^2+I_1)}{\epsilon^2\delta}, \quad L' = \left\lceil I_1 + \frac{3I_1}{\epsilon\delta} \right\rceil, \quad L = \Omega((\tau+\log(1/\delta))/\epsilon^2).$$

Assume that I_2 is a multiple of q. For each $i = 1,2,\ldots,q$, suppose that all the $\mathbf{S}_i$ are SpEmb matrices and all the $\mathbf{G}_i$ are standard Gaussian matrices. Let $\mathbf{P} \in \mathbb{R}^{I_2\times I_2}$ be a random permutation matrix. When the matrix $\mathbf{B} \in \mathbb{R}^{I_1\times L}$ is obtained by applying Algorithm 3 with SpEmb+Gaussian to $\mathbf{A} \in \mathbb{R}^{I_1\times I_2}$, then with probability at least $1-\delta-\exp(-\epsilon^2/2)$, one has

$$\|\mathbf{A}\mathbf{A}^\top - \mathbf{B}\mathbf{B}^\top\|_2 \leq \left(\epsilon + \epsilon\left(I_2 + \frac{I_2}{\tau}\right) + \frac{2I_1(\sqrt{L'}+\sqrt{L}+\epsilon)^2}{qL}\right)\|\mathbf{A}\|_F^2.$$

Remark 8 *Note that the condition for qL' in Theorem 7 can be replaced by*

$$qL' \geq \frac{12(I_1'^2+I_1')}{\epsilon^2\delta}, \quad L' = \left\lceil I_1' + \frac{3I_1'}{\epsilon\delta} \right\rceil,$$

where $I_1' \leq I_1$ is the rank of the matrix $\mathbf{A}$.

When $\widetilde{\mathbf{A}}_\mu$ be obtained by applying Algorithm 3 with SpEmb+Gaussian, the upper bound for $\|\mathbf{A}-\widetilde{\mathbf{A}}_\mu\|_F^2$ is obtained by combining (1) and Theorem 7, which is summarized in the following corollary.

Corollary 9 *Let all assumptions be the same as Theorem 7. For any positive integer $\mu < L$, let $\widetilde{\mathbf{A}}_\mu$ be obtained by applying Algorithm 3 with SpEmb+Gaussian to $\mathbf{A} \in \mathbb{R}^{I_1\times I_2}$, then with probability at least $1-\delta-\exp(-\epsilon^2/2)$, we have*

$$\|\mathbf{A}-\widetilde{\mathbf{A}}_\mu\|_F^2 \leq \|\mathbf{A}-\mathbf{A}_\mu\|_F^2 + 2\mu\left(\epsilon\left(1 + I_2 + \frac{I_2}{\tau}\right) + \frac{2I_1(\sqrt{L'}+\sqrt{L}+\epsilon)^2}{qL}\right)\|\mathbf{A}\|_F^2.$$

Finally, we derive the upper bound for $\|\mathbf{A}-\widetilde{\mathbf{A}}_\mu\|_F^2$, where $\widetilde{\mathbf{A}}_\mu$ is obtained by applying Algorithm 2 with standard Gaussian matrices.

Theorem 10 *Suppose that $\epsilon,\delta,\tau \in (0,1/2)$. Let I_1, I_2, L, μ and q satisfy that $\mu < L < \min\{I_1, I_2/q\}$ and $L = \Omega((\tau+\log(1/\delta))/\epsilon^2)$, where I_2 is a multiple of q. Let $\widetilde{\mathbf{A}}_\mu$ be obtained by applying Algorithm 2 with Gaussian to $\mathbf{A} \in \mathbb{R}^{I_1\times I_2}$, then with probability at least $1-\delta-\exp(-\epsilon^2/2)$, one has*

$$\|\mathbf{A}-\widetilde{\mathbf{A}}_\mu\|_F^2 \leq \|\mathbf{A}-\mathbf{A}_\mu\|_F^2 + 2\mu\left(\epsilon\left(1+\frac{1}{\tau}\right) + \frac{2(\sqrt{L'}+\sqrt{L}+\epsilon)^2}{L}\right)\|\mathbf{A}\|_F^2.$$

3.3 Revisiting Algorithm 3 based on SRHT matrices

We now consider Algorithm 3 by replacing each $\mathbf{G}_i$ as a SRHT matrix, with is denoted by Algorithm 3 with SpEmb+SRHT.

Lemma 11 (Ailon and Liberty, 2008, Theorem 2.1) *Given $\mathbf{x} \in \mathbb{R}^I$ and $L < I$, one can construct $\mathbf{\Omega} \in \mathbb{R}^{I\times L}$ and compute $\mathbf{\Omega}^\top \mathbf{x}$ in $\Theta(I\log(L))$ operations.*

Suppose that J is a power of two. Let $\mathbf{\Omega}$ be the $J \times L$ SRHT matrix with $J > L$. From Lemma 11, for any $\mathbf{A} \in \mathbb{R}^{I\times J}$, Then $\mathbf{A\Omega}$ can be computed in $\Theta(IJ\log(L))$ operations.

Remark 12 *It is worth noting that the SRHT is defined only when the matrix dimension is a power of two. An alternative option is to use other structured orthonormal randomized transforms such as discrete cosine transform (DCT) or the discrete Hartley transform (DHT)* (Rokhlin and Tygert (2008); Woolfe et al. (2008))*. The discrete Fourier matrix is the real analog of the Walsh-Hadamard matrix. When we consider the case of complex-valued tensors, we use the subsampled random Fourier transform* (Rokhlin and Tygert (2008); Woolfe et al. (2008)) *to replace the subsampled random Hadarmard transform.*

For clarity, we assume that I_2 is a multiplier of q and L' is a power of two. We now count the computational complexity of Algorithm 3 with all the matrices $\mathbf{G}_i$ being SRHT matrices, that is, $\mathbf{G}_i = \mathbf{\Omega}_i$. Comparison to standard Gaussian matrices, we list the different point as follows,

(2') it costs $q\cdot\Theta(I_1I_2/q+I_1L'\log(L)) = \Theta(I_1I_2+qI_1L'\log(L))$ operations to compute $\mathbf{B}(:,1:L) = \mathbf{A}(:,1:I_2/q)\mathbf{S}_1\mathbf{\Omega}_1$ and $\mathbf{B}(:,L+1:2L) = \mathbf{A}(:,iI_2/q+1:(i+1)I_2/q)\mathbf{S}_{i+1}\mathbf{\Omega}_{i+1}$ with $i = 1,2,\dots,q-1$.

Then, Algorithm 3 with SpEmb+SRHT requires

$$\begin{aligned}&\Theta(I_2 + qL'L + I_1I_2) + \Theta(qI_1L'\log(L)) + 2I_1I_2L\\ &+ 6I_2L^2 + 2I_1(I_2+L)\mu + 24I_1(q-1)L^2 + (160q-140)L^3\end{aligned}$$

operations to obtain a rank-μ approximation $\widetilde{\mathbf{A}}_\mu$ to $\mathbf{A}$. Finally, we include the computational complexity of the FD algorithm and its randomized variants in Table 1 for a clearer comparison. According to Table 1, the main difference among the FD algorithm and its randomized variants is the way to form $\mathbf{U_B}$.

When $\widetilde{\mathbf{A}}_\mu$ be obtained by applying Algorithm 3 with SpEmb+SRHT, the upper bound for $\|\mathbf{A}-\widetilde{\mathbf{A}}_\mu\|_F^2$ is given the following theorem.

Theorem 13 *For any $\epsilon,\delta \in (0,1/2)$, let q, I_1, I_2, L' and L be positive integers such that $L < L' < I_2$, $I_1 < I_2$ and*

$$qL' \geq \frac{12(I_1^2+I_1)}{\epsilon^2\delta},\quad L' = \left\lceil I_1 + \frac{3I_1}{\epsilon\delta}\right\rceil.$$

Assume that assume that I_2 is a multiplier of q, L' is a power of two with $L \leq \min\{I_1, L'\}$ and $L' < I_2/q$, and $0 < \beta < 1$ satisfies $0 < I_2\beta/q < 1$. For each $i = 1,2,\dots,q$, suppose that all the $\mathbf{S}_i$ are SpEmb matrices and all the $\mathbf{G}_i$ are SRHT matrices.

Algorithms	Form $\mathbf{U_B}$	Compute the SVD of $\mathbf{U}_\mathbf{B}^\top\mathbf{A}$	Obtain $\widetilde{\mathbf{A}}_\mu$
Algorithm 1	$(24I_1L^2 + 180L^3)(I_2/L - 1)$	$2I_1I_2L + 6I_2L^2 + 20L^3$	$2I_1(I_2 + L)\mu$
Algorithm 2 with Gaussian	$\Theta(I_2L) + 2I_1I_2L + 2(q-1)(24I_1L^2 + 160L^3)$	$2I_1I_2L + 6I_2L^2 + 20L^3$	$2I_1(I_2 + L)\mu$
Algorithm 2 with SpEmb	$\Theta(I_2 + I_1I_2) + 2I_1I_2L + 2(q-1)(24I_1L^2 + 160L^3)$	$2I_1I_2L + 6I_2L^2 + 20L^3$	$2I_1(I_2 + L)\mu$
Algorithm 2 with SRHT	$\Theta(I_1I_2\log L) + (q-1)(24I_1L^2 + 160L^3)$	$2I_1I_2L + 6I_2L^2 + 20L^3$	$2I_1(I_2 + L)\mu$
Algorithm 3 with SpEmb+Gaussian	$\Theta(I_2 + qL'L + I_1I_2) + 2qI_1L'L + (q-1)(24I_1L^2 + 160L^3)$	$2I_1I_2L + 6I_2L^2 + 20L^3$	$2I_1(I_2 + L)\mu$
Algorithm 3 with SpEmb+SRHT	$\Theta(I_2 + qL'L + I_1I_2) + \Theta(qI_1L'\log(L)) + (q-1)(24I_1L^2 + 160L^3)$	$2I_1I_2L + 6I_2L^2 + 20L^3$	$2I_1(I_2 + L)\mu$

Table 1: With a dense matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$, the computational complexity of Algorithm 1 and several randomized variants.

For any positive integer $\mu < L$, let $\widetilde{\mathbf{A}}_\mu$ be obtained by applying Algorithm 3 with SpEmb+SRHT to $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$, then with probability at least

$$1 - \delta - \frac{I_2}{q}\beta - \left(2\frac{I_2}{L'} + 1\right)\delta - \frac{2I_1}{\exp(\min\{L', I_1\})},$$

we have

$$\|\mathbf{A} - \widetilde{\mathbf{A}}_\mu\|_F^2 \leq \|\mathbf{A} - \mathbf{A}_\mu\|_F^2 + 2\mu\left(\epsilon + \widetilde{O}\left(\frac{1}{L} + \Gamma\left(L, \frac{I_2}{L'}, \min\{I_1, L'\}\right)\right)\frac{I_2}{q}\right)\|\mathbf{A}\|_F^2,$$

where

$$\Gamma\left(L, \frac{I_2}{L'}, \min\{I_1, L'\}\right) = \sqrt{\frac{\min\{I_1, L'\}}{L(I_2/L')^2}} + \sqrt{\frac{1 + \sqrt{\min\{I_2/L'\}/L}}{I_1/L'}},$$

and $\widetilde{O}(\cdot)$ hides logarithmic factors on β, δ, $\min\{I_1, L'\}$, I_2 and L'.

4. Applications to approximate tensor decomposition

In this section, we first introduce the definitions of Tucker decomposition and TT decomposition. We then present well-known algorithms for these two decompositions: T-HOSVD and ST-HOSVD for Tucker decomposition and TT-SVD for TT decomposition. We finally deduce the FD-based randomized variants of T-HOSVD, ST-HOSVD and TT-SVD.

4.1 Tucker decomposition with two common algorithms: T-HOSVD and ST-HOSVD

For a given n, the mode-n product of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ by $\mathbf{B} \in \mathbb{R}^{J_n \times I_n}$ (cf. Kolda and Bader (2009)), denoted by $\mathcal{A} \times_n \mathbf{B}$, is a tensor $\mathcal{C} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N}$, where its entries are given by

$$c_{i_1 \dots i_{n-1} j i_{n+1} \dots i_m} = \sum_{i_n=1}^{I_n} a_{i_1 \dots i_{n-1} i_n i_{n+1} \dots i_N} b_{j i_n}.$$

For two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the *Frobenius norm* of a tensor $\mathcal{A}$ is given by $\|\mathcal{A}\|_F = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$ and the scalar product $\langle \mathcal{A}, \mathcal{B} \rangle$ is defined as

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1, i_2, \dots, i_N = 1}^{I_1, I_2, \dots, I_N} a_{i_1 i_2 \dots i_N} b_{i_1 i_2 \dots i_N}.$$

As shown in (Kolda and Bader (2009)), the mode-n unfolding matrix of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, denoted by $\mathbf{A}_{(n)}$, arranges the mode-n fibers into the columns of a matrix. More specifically, a tensor element $(i_1, i_2, \dots, i_N)$ maps on a matrix element (i_n, j), where

$$\begin{aligned} j =& i_1 + (i_2 - 1)I_1 + \cdots + (i_{n-1} - 1)I_1 \dots I_{n-2} + (i_{n+1} - 1)I_1 \dots I_{n-1} \\ &+ \cdots + (i_N - 1)I_1 \dots I_{n-1} I_{n+1} \dots I_{N-1}. \end{aligned}$$

For given N positive integers $\mu_n < I_n$, the Tucker decomposition (Tucker (1966)) of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is given by

$$\mathcal{A} \approx \mathcal{G} \times_1 \mathbf{Q}_1 \times_2 \mathbf{Q}_2 \cdots \times_N \mathbf{Q}_N, \tag{2}$$

where $\mathbf{Q}_n \in \mathbb{R}^{I_n \times \mu_n}$ are the factor matrices and $\mathcal{G} \in \mathbb{R}^{\mu_1 \times \mu_2 \times \cdots \times \mu_N}$ is the core tensor. By using the mode-n unfolding of a tensor, we have

$$\mathbf{A}_{(n)} \approx \mathbf{Q}_n \mathbf{G}_{(n)} (\mathbf{Q}_N \otimes \cdots \otimes \mathbf{Q}_{n+1} \otimes \mathbf{Q}_{n-1} \otimes \cdots \otimes \mathbf{Q}_1)^\top.$$

Specifically, if the columns of $\mathbf{Q}_n$ are extracted from those of $\mathbf{A}_{(n)}$, (2) is called the CUR-type decomposition of $\mathcal{A}$ (cf. Caiafa and Cichocki (2010); Drineas and Mahoney (2007); Oseledets et al. (2008); Saibaba (2016)); if the entries of $\mathcal{A}$, $\mathcal{G}$ and $\mathbf{U}_n$ are nonnegative, (2) is called the nonnegative Tucker decomposition (cf. Zhang et al. (2016); Zhou et al. (2012, 2015)). The Tucker-rank of $\mathcal{A}$ is given by

$$\{\mathrm{rank}(\mathbf{A}_{(1)}), \mathrm{rank}(\mathbf{A}_{(2)}), \dots, \mathrm{rank}(\mathbf{A}_{(N)})\},$$

where $\mathrm{rank}(\mathbf{A}_{(n)})$ is the rank of $\mathbf{A}_{(n)}$ with $n = 1, 2, \dots, N$.

When all the factor matrices are orthonormal, a classical algorithm for computing the Tucker decomposition is better known as the higher-order singular value decomposition (HOSVD) (De Lathauwer et al. (2000a)). When $\mu_n < \mathrm{rank}(\mathbf{A}_{(n)})$ for some n, the decomposition is called the truncated HOSVD (T-HOSVD). The T-HOSVD is not optimal in terms of giving the best fit as measured by the norm of the difference, but it is a good starting point for the high-order orthogonal iteration (HOOI) (De Lathauwer et al. (2000b)) that is also used for computing the Tucker decomposition with orthonormal factor matrices.

Let $\mathcal{G}$ and $\mathbf{Q}_n$ be obtained from T-HSOVD and $\widehat{\mathcal{A}} = \mathcal{G} \times_1 \mathbf{Q}_1 \times_2 \mathbf{Q}_2 \cdots \times_N \mathbf{Q}_N$. Then the upper bound of $\|\mathcal{A} - \widehat{\mathcal{A}}\|_F$ is given in the following theorem.

Theorem 14 (Vannieuwenhoven et al., 2012, Theorem 5.1) *For a given tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and N positive integers μ_n with $\mu_n < I_n$, let $\mathcal{G}$ and $\mathbf{Q}_n$ be obtained from* T-HOSVD *with a Tucker-rank $\{\mu_1, \mu_2, \dots, \mu_N\}$, then we have*

$$\|\mathcal{A} - \widehat{\mathcal{A}}\|_F^2 \leq \sum_{n=1}^N \|\mathbf{A}_{(n)} - \mathbf{A}_{(n),\mu_n}\|_F^2,$$

with $\widehat{\mathcal{A}} = \mathcal{G} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N$, where $\mathbf{A}_{(n),\mu_n}$ is the best rank-μ_n approximation of $\mathbf{A}_{(n)}$.

Vannieuwenhoven et al. (2012) presented an alternative strategy to truncate the HOSVD, that is called the sequentially truncated higher-order singular value decomposition (ST-HOSVD). In contrast to T-HOSVD that processes the modes independently, the ST-HOSVD processes the modes sequentially with a given processing order.

Let $\mathcal{G}$ and $\mathbf{Q}_n$ be obtained from ST-HOSVD with $\mathbf{p} = \{1, 2, \dots, N\}$. Then the upper bound of $\|\mathcal{A} - \mathcal{G} \times_1 \mathbf{Q}_1 \times_2 \mathbf{Q}_2 \cdots \times_N \mathbf{Q}_N\|_F$ is given in the following theorem.

Theorem 15 (Vannieuwenhoven et al., 2012, Theorem 6.5) *For a given tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and N positive integers μ_n with $\mu_n < I_n$, let $\mathcal{G}$ and $\mathbf{Q}_n$ be obtained from* ST-HOSVD *with a Tucker-rank $\{\mu_1, \mu_2, \dots, \mu_N\}$ and $\mathbf{p} = \{1, 2, \dots, N\}$, then we have*

$$\|\mathcal{A} - \widehat{\mathcal{A}}\|_F^2 \leq \sum_{n=1}^N \|\mathbf{A}_{(n)} - \mathbf{A}_{(n),\mu_n}\|_F^2,$$

with $\widehat{\mathcal{A}} = \mathcal{G} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N$.

Remark 16 *As shown in Theorems* 14 *and* 15, *when* $I_n \leq I_1 \dots I_{n-1} I_{n+1} \dots I_N$, *the term* $\|\mathbf{A}_{(n)} - \mathbf{A}_{(n),\mu_n}\|_F^2$ *is the same as*

$$\|\mathbf{A}_{(n)} - \mathbf{A}_{(n),\mu_n}\|_F^2 = \sum_{i=\mu_n+1}^{I_n} \sigma_i(\mathbf{A}_{(n)})^2,$$

where $\sigma_i(\mathbf{A}_{(n)})$ *is the ith singular value of* $\mathbf{A}_{(n)}$.

As shown in (Vannieuwenhoven et al. (2012)), the computational complexity of T-HOVSD and ST-HOSVD are, respectively, given by

$$\begin{cases} \Theta\left(\sum_{n=1}^{N} \mu_n \prod_{j=1}^{N} I_j\right) + 2 \cdot \sum_{n=1}^{N} \prod_{i=1}^{n} \mu_i \prod_{j=n}^{N} I_j; \\ \Theta\left(\sum_{n=1}^{N} \prod_{i=1}^{n} \mu_i \prod_{j=n}^{N} I_j\right) + 2 \cdot \sum_{n=1}^{N} \prod_{i=1}^{n} \mu_i \prod_{j=n+1}^{N} I_j. \end{cases}$$

In particular, when $I_n = I$ and $\mu_n = \mu$ with $n = 1, 2, \dots, N$, then T-HOVSD (respectively ST-HOSVD) requires

$$\Theta\left(NI^N \mu\right) + 2 \cdot \sum_{n=1}^{N} \mu^n I^{N-n+1} \text{ and } \Theta\left(\sum_{n=1}^{N} \mu^n I^{N-n+1}\right) + 2 \cdot \sum_{n=1}^{N} \mu^n I^{N-n+1}$$

operations to obtain $\widehat{\mathcal{A}}$.

4.2 TT decomposition with the TT-SVD algorithm

The mode-(n, m) product (cf. Oseledets (2011)) (called *tensor-tensor product*) of two tensors $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ and $\mathcal{B} \in \mathbb{R}^{J_1 \times J_2 \times \dots \times J_M}$ with common modes $I_n = J_m$ that produces an order $(M + N - 2)$ tensor $\mathcal{C} \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times I_{n+1} \times \dots \times I_N \times J_1 \times \dots \times J_{m-1} \times J_{m+1} \times \dots \times J_M}$:

$$\mathcal{C} = \mathcal{A} \times_n^m \mathcal{B},$$

where its entries are given by

$$c_{i_1 \dots i_{n-1} i_{n+1} \dots i_N j_1 \dots j_{m-1} i_{m+1} \dots j_N} = \sum_{i_n=1}^{I_n} a_{i_1 \dots i_{n-1} i_n i_{n+1} \dots i_N} b_{j_1 \dots j_{m-1} i_n j_{m+1} \dots \dots j_M}.$$

The (i, j)-element of another unfolding matrix $\mathbf{A}_{([n])}$ of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ is defined as $\mathbf{A}_{([n])}(i, j) = a_{i_1 i_2 \dots i_N}$, where

$$\begin{aligned} i &= i_1 + (i_2 - 1) I_1 + \dots + (i_n - 1) I_1 \dots I_{n-1}, \\ j &= i_{n+1} + (i_{n+2} - 1) I_{n+1} + \dots + (i_N - 1) I_{n+1} \dots I_{N-1}. \end{aligned}$$

The TT-rank of $\mathcal{A}$ is defined as $\{\text{rank}(\mathbf{A}_{([1])}), \text{rank}(\mathbf{A}_{([2])}), \dots, \text{rank}(\mathbf{A}_{([N-1])})\}$, where $\text{rank}(\mathbf{A}_{([n])})$ is the rank of $\mathbf{A}_{([n])}$ with $n = 1, 2, \dots, N-1$. For given $(N-1)$ positive integers $\mu_n \leq \min\{I_1 \dots I_n, I_{n+1} \dots I_N\}$ with $n = 1, 2, \dots, N-1$, the TT decomposition of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ is described in the scalar form as

$$a_{i_1 i_2 \dots i_N} \approx \sum_{j_1=1}^{\mu_1} \sum_{j_2=1}^{\mu_2} \dots \sum_{j_{N-1}=1}^{\mu_{N-1}} q^{(1)}_{i_1, j_1} q^{(2)}_{j_1, i_2, j_2} \dots q^{(N)}_{j_{N-1}, i_N},$$

or equivalently by using slice representations:

$$a_{i_1 i_2 \dots i_N} \approx \mathbf{Q}^{(1)}(i_1) \mathbf{Q}^{(2)}(i_2) \dots \mathbf{Q}^{(N)}(i_N),$$

where slice matrices are defined as

$$\mathbf{Q}^{(n)}(i_n) = \mathcal{Q}^{(n)}(:, i_n, :) \in \mathbb{R}^{\mu_{n-1} \times \mu_n},$$

i.e., $\mathbf{Q}^{(n)}(i_n)$ is an i_nth lateral slice of the core $\mathcal{Q}^{(n)} \in \mathbb{R}^{\mu_{n-1} \times I_n \times \mu_n}$ for $n = 1, 2, \dots, N$ with $\tau_0 = \tau_N = 1$. By the tensor-tensor product, one has

$$\mathcal{A} \approx \mathcal{Q}^{(1)} \times_3^1 \mathcal{Q}^{(2)} \times_3^1 \dots \times_3^1 \mathcal{Q}^{(N)}.$$

Using TT-SVD with a given TT-rank $\{\mu_1, \mu_2, \dots, \mu_{N-1}\}$ on a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$, where $\mu_n \leq \min\{\mu_{n-1} I_n, I_{n+1} \dots I_N\}$ and $\mu_0 = 1$, we obtain an approximate TT decomposition of $\mathcal{A}$ as $\widetilde{\mathcal{A}} = \mathcal{Q}^{(1)} \times_3^1 \mathcal{Q}^{(2)} \times_3^1 \dots \times_3^1 \mathcal{Q}^{(N)}$, where $\mathcal{Q}^{(n)} \in \mathbb{R}^{\mu_{n-1} \times I_n \times \mu_n}$ is left orthogonal to $n = 1, 2, \dots, N-1$. The computational complexity of TT-SVD is given by

$$\Theta\left(\sum_{n=1}^{N-1} \mu_{n-1} \mu_n I_n \dots I_N\right) + 2 \cdot \sum_{n=1}^{N-1} \mu_{n-1} \mu_n I_n \dots I_N$$

with $\mu_0 = 1$. The upper bound of $\|\mathcal{A} - \widetilde{\mathcal{A}}\|_F$ is given in the following theorem.

Theorem 17 (Oseledets and Tyrtyshnikov, 2010, Theorem 2.2) *For a given tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ and $(N-1)$ positive integers $\mu_n \leq \min\{\mu_{n-1} I_n, I_{n+1} \dots I_N\}$ with $\mu_0 = 1$, let $\mathcal{Q}^{(n)} \in \mathbb{R}^{\mu_{n-1} \times I_n \times \mu_n}$ be obtain from* TT-SVD *with a TT-rank $\{\mu_1, \mu_2, \dots, \mu_{N-1}\}$, then one has*

$$\|\mathcal{A} - \widetilde{\mathcal{A}}\|_F^2 \leq \sum_{n=1}^{N-1} \|\mathbf{A}_{([n])} - \mathbf{A}_{([n]), \mu_n}\|_F^2$$

with $\widetilde{\mathcal{A}} = \mathcal{Q}^{(1)} \times_3^1 \mathcal{Q}^{(2)} \times_3^1 \dots \times_3^1 \mathcal{Q}^{(N)}$, where $\mathbf{A}_{([n]), \mu_n}$ is the best rank-μ_n approximation of the matrix $\mathbf{A}_{([n])}$.

4.3 The FD-based randomized variant for T-HOSVD

As shown in (De Lathauwer et al. (2000b); Kolda and Bader (2009)), for a desired Tucker-rank $\{\mu_1, \mu_2, \dots, \mu_N\}$, all the factor matrices $\{\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_N\}$ for Tucker decomposition of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ can be obtained by the following problem.

Problem 1 *Suppose that $\mathcal{A} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$. For given N positive integers $\mu_n < I_n$ with $n = 1, 2, \dots, N$, the objective is to find N orthonormal matrices $\mathbf{Q}_n \in \mathbb{R}^{I_n\times\mu_n}$ such that*

$$\{\mathbf{Q}_1, \dots, \mathbf{Q}_N\} = \underset{\mathbf{V}_1,\dots,\mathbf{V}_N}{\operatorname{argmin}} \left\|\mathcal{A} - \mathcal{A} \times_1 (\mathbf{V}_1\mathbf{V}_1^\top)\cdots \times_N (\mathbf{V}_N\mathbf{V}_N^\top)\right\|_F^2,$$

where all the matrices $\mathbf{V}_n \in \mathbb{R}^{I_n\times\mu_n}$ are orthonormal.

After obtaining all the matrices $\{\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_N\}$, we can compute the core tensor $\mathcal{G} \in \mathbb{R}^{\mu_1\times\mu_2\times\cdots\times\mu_N}$ as $\mathcal{G} = \mathcal{A} \times_1 \mathbf{Q}_1^\top \times_2 \mathbf{Q}_2^\top \cdots \times_N \mathbf{Q}_N^\top$. For a solution $\{\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_N\}$ for Problem 1, we have (cf. Che and Wei (2019); Vannieuwenhoven et al. (2012))

$$\begin{aligned}
&\mathcal{A} - \mathcal{A} \times_1 (\mathbf{Q}_1\mathbf{Q}_1^\top) \times_2 (\mathbf{Q}_2\mathbf{Q}_2^\top)\cdots \times_N (\mathbf{Q}_N\mathbf{Q}_N^\top)\\
&\quad = \mathcal{A} \times_1 (\mathbf{I}_{I_1} - \mathbf{Q}_1\mathbf{Q}_1^\top) + \mathcal{A} \times_1 (\mathbf{Q}_1\mathbf{Q}_1^\top) \times_2 (\mathbf{I}_{I_2} - \mathbf{Q}_2\mathbf{Q}_2^\top)\\
&\quad + \cdots + \mathcal{A} \times_1 (\mathbf{Q}_1\mathbf{Q}_1^\top)\cdots \times_{N-1} (\mathbf{Q}_{N-1}\mathbf{Q}_{N-1}^\top) \times_N (\mathbf{I}_{I_N} - \mathbf{Q}_N\mathbf{Q}_N^\top).
\end{aligned} \tag{3}$$

From (3), by using the triangular inequality and the fact that $\|\mathbf{Q}_n\mathbf{Q}_n^\top\|_2 \le 1$ with $n = 1, 2, \dots, N-1$, we have

$$\|\mathcal{A} - \mathcal{A} \times_1 (\mathbf{Q}_1\mathbf{Q}_1^\top)\cdots \times_N (\mathbf{Q}_N\mathbf{Q}_N^\top)\|_F^2 \le \sum_{n=1}^{N} \|\mathcal{A} \times_1 (\mathbf{I}_{I_n} - \mathbf{Q}_n\mathbf{Q}_n^\top)\|_F^2. \tag{4}$$

Hence in order to finding an approximate solution $\{\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_N\}$ to Problem 1, we must consider the following problem.

Problem 2 *For a given positive integer $1 \le n \le N$, let μ_n be a positive integer with $\mu_n < I_n$. Suppose that $\mathcal{A} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, the objective is to find an orthonormal matrix $\mathbf{Q}_n \in \mathbb{R}^{I_n\times\mu_n}$ such that*

$$\mathbf{Q}_n = \underset{\mathbf{V}_n}{\operatorname{argmin}} \|\mathcal{A} \times_n (\mathbf{I}_{I_n} - \mathbf{V}_n\mathbf{V}_n^\top)\|_F^2 = \underset{\mathbf{V}_n}{\operatorname{argmin}} \|(\mathbf{I}_{I_n} - \mathbf{V}_n\mathbf{V}_n^\top)\mathbf{A}_{(n)}\|_F^2,$$

where $\mathbf{V}_n \in \mathbb{R}^{I_n\times\mu_n}$ is orthonormal.

For each n, by applying Algorithm 3 to $\mathbf{A}_{(n)}$, we can obtain an orthonormal matrix $\mathbf{Q}_n$ such that the column space of $\mathbf{Q}_n$ is an approximation of that of $\mathbf{A}_{(n)}$. The overall process is summarized in Algorithm 4. For clarity, we assume that for each n, $I_1 \dots I_{n-1}I_{n+1}\dots I_N$ is a multiple of q_n. For the case of Algorithm 3 with SpEmb+Gaussian, the computational complexity of Algorithm 4 is given as

$$\begin{aligned}
&\sum_{n=1}^{N} \big(\Theta(I_1\dots I_{n-1}I_{n+1}\dots I_N + q_nL_n'L_n + I_1I_2\dots I_N) + 2q_nI_nL_n'L_n + 2I_1I_2\dots I_NL_n\big)\\
&+ \sum_{n=1}^{N} \big(6I_1\dots I_{n-1}I_{n+1}\dots I_NL_n^2 + 2I_n(I_{n-1}I_{n+1}\dots I_N + L_n)\mu_n\big)\\
&+ \sum_{n=1}^{N} \big(24I_n(q_n-1)L_n^2 + (160q_n - 140)L_n^3 + 2\mu_1\dots\mu_nI_n\dots I_N\big),
\end{aligned}$$

Algorithm 4 Randomized T-HOSVD with Algorithm 3

```
    Input: A tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the desired Tucker-rank $\{\mu_1, \mu_2, \dots, \mu_N\}$, two tuples of sketch sizes $\{L'_1, L'_2, \dots, L'_N\}$ and $\{L_1, L_2, \dots, L_N\}$ with $L'_n > L_n$, and the tuple of numbers of blocks $\{q_1, q_2, \dots, q_N\}$.
    Output: $N$ orthonormal matrices $\mathbf{Q}_n$ and a core tensor $\mathcal{G}$ such that $\mathcal{A} \approx \widehat{\mathcal{A}} = \mathcal{G} \times_1 \mathbf{Q}_1 \times_2 \mathbf{Q}_2 \cdots \times_N \mathbf{Q}_N$.
1:  Set a temporary tensor $\mathcal{C} = \mathcal{A}$.
2:  for $n = 1, 2, \dots, N$ do
3:      Form the mode-$n$ unfolding $\mathbf{A}_{(n)}$ of $\mathcal{A}$.
4:      Obtain $\mathbf{Q}_n$ by applying Algorithm 3 to $\mathbf{A}_{(n)}$ with $\mu_n$, $L'_n$, $L_n$ and $q_n$.
5:      Update $\mathcal{C} = \mathcal{C} \times_n \mathbf{Q}_n^\top$.
6:  end for
7:  Write the core tensor $\mathcal{G}$ as $\mathcal{G} = \mathcal{C}$.
```

which is simplified as

$$\sum_{n=1}^{N} (I_n + I_n(L_n + \mu_n) + L_n^2 + 1)\Theta(I_1 \dots I_{n-1} I_{n+1} \dots I_N) + 2\mu_1 I_1 \dots I_N$$

under the case that for each n, $\mu_n < L_n < L'_n \ll \min\{I_n, I_1 \dots I_{n-1} I_{n+1} \dots I_N\}$.

Remark 18 *For the case of Algorithm 3 with SpEmb+SRHT, the computational complexity of Algorithm 4 is given as*

$$\begin{aligned}
&\sum_{n=1}^{N} \left(\Theta(I_1 \dots I_{n-1} I_{n+1} \dots I_N + q_n L'_n L_n + I_1 I_2 \dots I_N) + \Theta(q_n I_n L'_n \log(L_n)) + 2 I_1 I_2 \dots I_N L_n\right) \\
&+ \sum_{n=1}^{N} \left(6 I_1 \dots I_{n-1} I_{n+1} \dots I_N L_n^2 + 2 I_n (I_{n-1} I_{n+1} \dots I_N + L_n)\mu_n\right) \\
&+ \sum_{n=1}^{N} \left(24 I_n (q_n - 1) L_n^2 + (160 q_n - 140) L_n^3 + 2\mu_1 \dots \mu_n I_n \dots I_N\right).
\end{aligned}$$

For the case of Algorithm 3 with SpEmb+Gaussian, when $\widehat{\mathcal{A}} = \mathcal{G} \times_1 \mathbf{Q}_1 \times_2 \mathbf{Q}_2 \cdots \times_N \mathbf{Q}_N$ is obtained by Algorithm 4, the upper bound for $\|\mathcal{A} - \widehat{\mathcal{A}}\|_F^2$ is easily obtained by using Corollary 9 to each part in the right-hand side of (4).

Theorem 19 *For each n, let μ_n, q_n, I_n, L'_n and L_n be positive integers such that $\mu_n < L_n < L'_n < \min\{I_n, (I_1 \dots I_{n-1} I_{n+1} \dots I_N)/q_n\}$, $I_1 \dots I_{n-1} I_{n+1} \dots I_N$ is a multiple of q_n and*

$$q_n L'_n \geq \frac{12(I_n^2 + I_n)}{\epsilon_n^2 \delta_n}, \quad L'_n = \left\lceil I_n + \frac{3 I_n}{\epsilon_n \delta_n} \right\rceil, \quad L_n = \Omega((\tau_n + \log(1/\delta_n))/\epsilon_n^2),$$

with $\epsilon_n, \delta_n, \tau_n \in (0, 1/2)$ and $I_n < I_1 \dots I_{n-1} I_{n+1} \dots I_N$. For the case of Algorithm 3 with SpEmb+Gaussian, when the tensor $\widehat{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is obtained by applying Algorithm 4 *to* $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$*, then with probability at least* $1 - \sum_{n=1}^{N} (\delta_n + \exp(-\epsilon_n^2/2))$*, we have*

$$\|\mathcal{A} - \widehat{\mathcal{A}}\|_F^2 \leq \sum_{n=1}^{N} \left(\sum_{i_n=\mu_n+1}^{I_n} (\sigma_{i_n}(\mathbf{A}_{(n)}))^2 + T_n \|\mathcal{A}\|_F^2 \right),$$

where for each n*, the coefficient* T_n *is given by*

$$T_n = 2\mu_n \left(\epsilon_n + \epsilon_n I_1 \dots I_{n-1} I_{n+1} \dots I_N \left(1 + \frac{1}{\tau_n} \right) \right) + \frac{4 I_n \mu_n (\sqrt{L_n'} + \sqrt{L_n} + \epsilon_n)^2}{q_n L_n}.$$

4.4 The FD-based randomized variant for ST-HOSVD

On the other hand, from (3), we can also have

$$\begin{aligned}
&\left\| \mathcal{A} - \mathcal{A} \times_1 (\mathbf{Q}_1 \mathbf{Q}_1^\top) \times_2 (\mathbf{Q}_2 \mathbf{Q}_2^\top) \cdots \times_N (\mathbf{Q}_N \mathbf{Q}_N^\top) \right\|_F^2 \\
&\leq \left\| \mathcal{A} \times_1 (\mathbf{I}_{I_1} - \mathbf{Q}_1 \mathbf{Q}_1^\top) \right\|_F^2 + \left\| \mathcal{A} \times_1 \mathbf{Q}_1^\top \times_2 (\mathbf{I}_{I_2} - \mathbf{Q}_2 \mathbf{Q}_2^\top) \right\|_F^2 \\
&\quad + \cdots + \left\| \mathcal{A} \times_1 \mathbf{Q}_1^\top \cdots \times_{N-1} \mathbf{Q}_{N-1}^\top \times_N (\mathbf{I}_{I_N} - \mathbf{Q}_N \mathbf{Q}_N^\top) \right\|_F^2. \qquad (5)
\end{aligned}$$

Hence, each $\mathbf{Q}_n$ can be found by solving the following problem.

Problem 3 *For a given* n*, let* μ_n *be a positive integer with* $\mu_n < I_n$ *and* $\mathbf{Q}_i \in \mathbb{R}^{I_i \times \mu_i}$ *be orthonormal with* $i = 1, 2, \dots, n-1$*. Suppose that* $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$*, the objective is to find an orthonormal matrix* $\mathbf{Q}_n \in \mathbb{R}^{I_n \times \mu_n}$ *such that*

$$\begin{aligned}
\mathbf{Q}_n &= \underset{\mathbf{V}_n}{\operatorname{argmin}} \|\mathcal{A} \times_1 \mathbf{Q}_1^\top \cdots \times_{n-1} \mathbf{Q}_{n-1}^\top \times_n (\mathbf{I}_{I_n} - \mathbf{V}_n \mathbf{V}_n^\top)\|_F^2 \\
&= \underset{\mathbf{V}_n}{\operatorname{argmin}} \|(\mathbf{I}_{I_n} - \mathbf{V}_n \mathbf{V}_n^\top) \mathbf{C}_{(n)}\|_F^2
\end{aligned}$$

with $\mathcal{C} = \mathcal{A} \times_1 \mathbf{Q}_1^\top \cdots \times_{n-1} \mathbf{Q}_{n-1}^\top \in \mathbb{R}^{\mu_1 \times \dots \mu_{n-1} \times I_n \times \dots \times I_N}$*, where the matrix* $\mathbf{V}_n \in \mathbb{R}^{I_n \times \mu_n}$ *is orthonormal.*

For a given Tucker-rank $\{\mu_1, \mu_2, \dots, \mu_N\}$, the basic idea of Algorithm 5 is that for each n, we compute the factor matrix $\mathbf{Q}_n$ by using Algorithm 3 to $\mathbf{C}_{(n)}$ with given L_n, L_n' and q_n, and update $\mathcal{C}$ as $\mathcal{C} = \mathcal{C} \times_n \mathbf{Q}_n^\top$. Note that the column space of $\mathbf{Q}_n$ an approximation of that of $\mathbf{C}_{(n)}$. By the relationship between $\mathcal{A}$ and $\mathcal{C}$, we have $\mathbf{C}_{(n)} = \mathbf{A}_{(n)} (\mathbf{I}_{I_N} \otimes \cdots \otimes \mathbf{I}_{I_n} \otimes \mathbf{Q}_{n-1} \otimes \cdots \otimes \mathbf{Q}_1)$, which implies that the column space of $\mathbf{Q}_n$ an approximation of that of $\mathbf{A}_{(n)}$.

For each n, we suppose that $\prod_{i=1}^{n-1} \mu_i \prod_{j=n}^{N} I_j$ is a multiple of q_n, and it is worthy noting that $\mathbf{C}_{(n)} \in \mathbb{R}^{I_n \times \prod_{i=1}^{n-1} \mu_i \prod_{j=n+1}^{N} I_j}$. Then, for each n, the application of Algorithm 3 with SpEmb+Gaussian to $\mathbf{C}_{(n)}$ with μ_n, L_n', L_n and q_n requires

$$\begin{aligned}
&\Theta(\mu_1 \dots \mu_{n-1} I_{n+1} \dots I_N + q_n L_n' L_n + \mu_1 \dots \mu_{n-1} I_n \dots I_N) \\
&+ 2 q_n I_n L_n' L_n + 2 \mu_1 \dots \mu_{n-1} I_n \dots I_N L_n \\
&+ 6 \mu_1 \dots \mu_{n-1} I_{n+1} \dots I_N L_n^2 + 2 I_n (\mu_1 \dots \mu_{n-1} I_{n+1} \dots I_N + L_n) \mu_n \\
&+ 24 I_n (q_n - 1) L_n^2 + (160 q_n - 140) L_n^3
\end{aligned}$$

Algorithm 5 Randomized ST-HOSVD with Algorithm 3

```
**Input**: A tensor $\mathcal{A} \in \mathbb{R}^{I_1\times I_2\times\dots\times I_N}$, the desired Tucker-rank $\{\mu_1,\mu_2,\dots,\mu_N\}$, two tuples of sketch sizes $\{L_1',L_2',\dots,L_N'\}$ and $\{L_1,L_2,\dots,L_N\}$ with $L_n' > L_n$, and the tuple of numbers of blocks $\{q_1,q_2,\dots,q_N\}$.
**Output**: $N$ orthonormal matrices $\mathbf{Q}_n$ and a core tensor $\mathcal{G}$ such that $\mathcal{A} \approx \widehat{\mathcal{A}} = \mathcal{G}\times_1 \mathbf{Q}_1\times_2\mathbf{Q}_2\cdots\times_N\mathbf{Q}_N$.
1: Set a temporary tensor $\mathcal{C} = \mathcal{A}$.
2: for $n = 1,2,\dots,N$ do
3:     Form the mode-$n$ unfolding $\mathbf{C}_{(n)}$ of $\mathcal{C}$.
4:     Obtain $\mathbf{Q}_n$ by applying Algorithm 3 to $\mathbf{C}_{(n)}$ with $\mu_n$, $L_n'$, $L_n$ and $q_n$.
5:     Update $\mathcal{C} = \mathcal{C}\times_n\mathbf{Q}_n^\top$.
6: end for
7: Write the core tensor $\mathcal{G}$ as $\mathcal{G} = \mathcal{C}$.
```

operations to obtain a rank-μ_n approximation $(\widetilde{\mathbf{C}}_{(n)})_{\mu_n}$ to $\mathbf{C}_{(n)}$. For each n, updating the tensor $\mathcal{C}$ needs $2\mu_1\dots\mu_nI_n\dots I_N$ operations. Hence, if for each n, $\mu_n < L_n < L_n' < \min\{I_n,\mu_1\dots\mu_{n-1}I_{n+1}\dots I_N\}$, then Algorithm 5 needs

$$\sum_{n=1}^{N}(I_n + 2I_n(L_n+\mu_n) + L_n^2 + 1)\cdot\Theta(\mu_1\dots\mu_{n-1}I_{n+1}\dots I_N)$$

operations to finding an approximate Tucker decomposition $\widetilde{\mathcal{A}}$ to $\mathcal{A}$ with a given Tucker-rank $\{\mu_1,\mu_2,\dots,\mu_N\}$. The computational complexity of Algorithm 5 is similarly considered for the case that for each n, each $\mathbf{Q}_n$ is obtained by applying Algorithm 3 with SpEmb+SRHT to $\mathbf{C}_{(n)}$ with μ_n, L_n', L_n and q_n.

It is shown in Algorithm 5 that for each n, we have $\mathcal{C} = \mathcal{A}\times_1\mathbf{Q}_1^\top\cdots\times_{n-1}\mathbf{Q}_{n-1}^\top$. Then from Lemma 9 in (Che et al. (2025a)), we have

$$\sum_{i_n=\mu_n+1}^{I_n}(\sigma_{i_n}(\mathbf{C}_{(n)}))^2 \le \sum_{i_n=\mu_n+1}^{I_n}(\sigma_{i_n}(\mathbf{A}_{(n)}))^2.$$

Meanwhile, we also have $\|\mathcal{C}\|_F \le \|\mathcal{A}\|_F$. Hence, by combining these two facts with Corollary 9 and (3), we deduce the upper bound for $\|\mathcal{A}-\widehat{\mathcal{A}}\|_F^2$, when $\widehat{\mathcal{A}} = \mathcal{G}\times_1\mathbf{Q}_1\times_2\mathbf{Q}_2\cdots\times_N\mathbf{Q}_N$ is obtained by Algorithm 5.

Theorem 20 *For each n, let μ_n, q_n, I_n, L_n' and L_n be positive integers such that $\mu_n < L_n < L_n' < \min\{I_n,(\mu_1\dots\mu_{n-1}I_{n+1}\dots I_N)/q_n\}$, $\mu_1\dots\mu_{n-1}I_{n+1}\dots I_N$ is a multiple of q_n and*

$$q_nL_n' \ge \frac{12(I_n^2+I_n)}{\epsilon_n^2\delta_n},\quad L_n' = \left\lceil I_n + \frac{3I_n}{\epsilon_n\delta_n}\right\rceil,\quad L_n = \Omega((\tau_n+\log(1/\delta_n))/\epsilon_n^2),$$

with $\epsilon_n,\delta_n,\tau_n\in(0,1/2)$ and $I_n < \mu_1\dots\mu_{n-1}I_{n+1}\dots I_N$. For the case of Algorithm 3 with SpEmb+Gaussian, when the tensor $\widehat{\mathcal{A}}\in\mathbb{R}^{I_1\times I_2\times\dots\times I_N}$ is obtained by applying Algorithm 5 *to* $\mathcal{A}\in\mathbb{R}^{I_1\times I_2\times\dots\times I_N}$*, then with probability at least* $1-\sum_{n=1}^N(\delta_n+\exp(-\epsilon_n^2/2))$*, we have*

$$\|\mathcal{A}-\widehat{\mathcal{A}}\|_F^2\leq\sum_{n=1}^N\left(\sum_{i_n=\mu_n+1}^{I_n}(\sigma_{i_n}(\mathbf{A}_{(n)}))^2+T_n\|\mathcal{A}\|_F^2\right),$$

where for each n*, the coefficient* T_n *is given by*

$$T_n=2\mu_n\left(\epsilon_n+\epsilon_n\mu_1\dots\mu_{n-1}I_{n+1}\dots I_N\left(1+\frac{1}{\tau_n}\right)\right)+\frac{4I_n\mu_n(\sqrt{L_n'}+\sqrt{L_n}+\epsilon_n)^2}{q_nL_n}.$$

4.5 The FD-based randomized variant for TT-SVD

For $m=1,2,\dots,M$ and $n=1,2,\dots,N$, let M positive integers J_m and N positive integers I_n satisfy $J_1J_2\dots J_M=I_1I_2\dots I_N$. If $\mathcal{A}\in\mathbb{R}^{I_1\times I_2\times\dots\times I_N}$, then *reshape*$(\mathcal{A},[J_1,J_2,\dots,J_M])$ returns a tensor in $\mathbb{R}^{J_1\times J_2\times\dots\times J_M}$ by stacking entries according to their multi-index.

Following from (Che and Wei (2019); Che et al. (2026)), the approximate TT decomposition of $\mathcal{A}\in\mathbb{R}^{I_1\times I_2\times\dots\times I_N}$ with a given TT-rank $\{\mu_1,\mu_2,\dots,\mu_{N-1}\}$ is summarized in the following problem.

Problem 4 (see (Che et al., 2026, Problem 3.1)) *Let* $\mu_n<\min\{\mu_{n-1}I_n,\prod_{k=n+1}^N I_k\}$ *with* $\mu_0=1$ *and* $n=1,2,\dots,N-1$*. For a tensor* $\mathcal{A}\in\mathbb{R}^{I_1\times I_2\times\dots\times I_N}$*, we want to find* N *tensors* $\mathcal{Q}_n\in\mathbb{R}^{\mu_{n-1}\times I_n\times\mu_n}$ *with* $\mu_N=1$ *to solve the following optimization problem*

$$\min_{\mathcal{G}_1,\dots,\mathcal{G}_N}\|\mathcal{A}-\mathcal{G}_1\times_3^1\mathcal{G}_2\times_3^1\dots\times_3^1\mathcal{G}_N\|_F,$$

where for $n=1,2,\dots,N-1$*, the core tensors* $\mathcal{G}_n$ *satisfy*

$$\mathbf{G}_n^\top\mathbf{G}_n=\mathbf{I}_{\mu_n},\quad \mathbf{G}_n=\mathrm{reshape}(\mathcal{G}_n,[\mu_{n-1}I_n,\mu_n]).$$

Suppose that $\{\mathcal{Q}_1,\mathcal{Q}_2,\dots,\mathcal{Q}_N\}$ is a solution of Problem 4. Let $\mathbf{Q}_1=\mathcal{Q}_1$ and $\mathbf{Q}_n=\mathrm{reshape}(\mathcal{Q}_n,[\mu_{n-1}I_n,\mu_n])$ with $n=2,3,\dots,N-1$. Let $\mathcal{A}_0=\mathcal{A}$ and define

$$\begin{cases}\mathcal{A}_1=\mathcal{Q}_1\times_1^1\mathcal{A}_0\in\mathbb{R}^{\mu_1\times I_2\times\dots\times I_N},\\ \mathcal{A}_n=\mathcal{Q}_n\times_{1,2}^{1,2}\mathcal{A}_{n-1}\in\mathbb{R}^{\mu_n\times I_{n+1}\times\dots\times I_N},\end{cases}$$

with $n=2,3,\dots,N-1$. Hence, we have (see (Che et al. (2026)))

$$\|\mathcal{A}-\mathcal{Q}_1\times_2^1\mathcal{Q}_2\times_3^1\dots\times_3^1\mathcal{Q}_N\|_F\leq\sum_{n=1}^{N-1}\|\mathbf{A}_n-\mathbf{Q}_n\mathbf{Q}_n^\top\mathbf{A}_n\|_F,\tag{6}$$

with $\mathbf{A}_n=\mathrm{reshape}(\mathcal{A}_{n-1},[\mu_{n-1}I_n,I_{n+1}\dots I_N])$. Therefore, an approximate solution for Problem 4 can be obtained by solving the following problem.

Problem 5 (Che et al., 2026, Problem 3.2) *Suppose that* $\mathcal{A}\in\mathbb{R}^{I_1\times I_2\times\dots\times I_N}$*. For a given TT-rank* $\{\mu_1,\mu_2,\dots,\mu_{N-1}\}$*, an approximate solution for Problem 4 can be obtained solving the following* $(N-1)$ *optimization problems:*

(a) *For the case of $n = 1$, when $\mu_1 \leq \min\{I_1, I_2 \dots I_N\}$, the objective is to find an orthonormal matrix $\mathbf{Q}_1 \in \mathbb{R}^{I_1 \times \mu_1}$ such that*

$$\mathbf{Q}_1 = \text{argmin}_{\mathbf{U}_1} \|\mathbf{A}_1 - \mathbf{U}_1\mathbf{U}_1^\top \mathbf{A}_1\|_F$$

where $\mathbf{U}_1 \in \mathbb{R}^{I_1 \times \mu_1}$ is orthonormal.

(b) *For the case of $n = 2, 3, \dots, N-1$, when $\mu_n < \min\{\mu_{n-1}I_n, I_{n+1} \dots I_N\}$, the objective is to find an orthonormal matrix $\mathbf{Q}_n \in \mathbb{R}^{\mu_{n-1}I_n \times \mu_n}$ such that*

$$\mathbf{Q}_n = \text{argmin}_{\mathbf{U}_n} \|\mathbf{A}_n - \mathbf{U}_n\mathbf{U}_n^\top \mathbf{A}_n\|_F,$$

where $\mathbf{U}_n \in \mathbb{R}^{\mu_{n-1}I_n \times \mu_n}$ is orthonormal.

Similar to the process for deducing Algorithms 4 and 5, Algorithm 6 is a FD-based randomized variant for TT-SVD, which is derived by applying Algorithm 3 for each subproblem in Problem 5. We now count the computational complexity of Algorithm 6. For

Algorithm 6 Randomized TT-SVD with Algorithm 3

```
Input: $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, a given TT-rank $\{\mu_1, \mu_2, \dots, \mu_{N-1}\}$, two tuples of sketch sizes $\{L'_1, L'_2, \dots, L'_{N-1}\}$ and $\{L_1, L_2, \dots, L_{N-1}\}$ with $L'_n > L_n$, and the tuple of numbers of blocks $\{q_1, q_2, \dots, q_{N-1}\}$.
Output: An approximation of the TT decomposition of $\mathcal{A}$: $\widetilde{\mathcal{A}} = \mathcal{Q}_1 \times_2^1 \mathcal{Q}_2 \times_3^1 \cdots \times_3^1 \mathcal{Q}_N$, where $\mathcal{Q}_n \in \mathbb{R}^{\mu_{n-1} \times I_n \times \mu_n}$ with $\mu_0 = \mu_N = 1$.
1: Set a temporary matrix $\mathbf{A}_1 = \text{reshape}(\mathcal{A}, [I_1, I_2 \dots I_N])$.
2: for $n = 1, 2, \dots, N-1$ do
3:   Obtain $\mathbf{Q}_n$ by applying Algorithm 3 to $\mathbf{A}_n$ with $\mu_n$, $L'_n$, $L_n$ and $q_n$.
4:   Reshape $\mathbf{Q}_n$ as a tensor $\mathcal{Q}_n \in \mathbb{R}^{\mu_{n-1} \times I_n \times \mu_n}$.
5:   Compute $\mathbf{B}_n = \mathbf{Q}_n^\top \mathbf{A}_n$.
6:   if $n < N-1$ then
7:     Form the matrix $\mathbf{A}_{n+1}$ as $\mathbf{A}_{n+1} = \text{reshape}(\mathbf{B}_n, [\mu_n I_{n+1}, I_{n+2} \dots I_N])$.
8:   else
9:     Reshape $\mathbf{B}_{N-1}$ as a tensor $\mathcal{Q}_N \in \mathbb{R}^{\mu_{N-1} \times I_N}$.
10:  end if
11: end for
```

clarity, for each $n = 1, 2, \dots, N-1$, let $\prod_{j=n}^{N} I_j$ be a multiple of q_n and

$$\mu_n < L_n < L'_n < \min\{\mu_{n-1}I_n, (I_{n+1} \dots I_N)/q_n\}.$$

Then, for each $n = 1, 2, \dots, N-1$, the application of Algorithm 3 with SpEmb+Gaussian to $\mathbf{A}_n$ with μ_n, L'_n, L_n and q_n requires

$$\begin{aligned}
&\Theta(I_{n+1} \dots I_N + q_n L'_n L_n + \mu_{n-1} I_n I_{n+1} \dots I_N) \\
&+ 2q_n \mu_{n-1} I_n L'_n L_n + 2\mu_{n-1} I_n I_{n+1} \dots I_N L_n \\
&+ 6 I_{n+1} \dots I_N L_n^2 + 2\mu_{n-1} I_n (I_{n+1} \dots I_N + L_n)\mu_n \\
&+ 24\mu_{n-1} I_n (q_n - 1) L_n^2 + (160 q_n - 140) L_n^3
\end{aligned}$$

operations to obtain a rank-μ approximation $(\widetilde{\mathbf{A}}_n)_{\mu_n}$ to $\mathbf{A}_n$. Meanwhile, for $n=1$, forming $\mathbf{A}_1$ and $\mathcal{Q}_1$ needs $\Theta(I_1I_2\dots I_N+I_1\mu_1)$ operations, for $n=2,3,\dots,N-2$, forming $\mathbf{A}_n$ and $\mathcal{Q}_n$ requires $2\mu_{n-1}\mu_nI_nI_{n+1}\dots I_N+\Theta(\mu_{n-1}\mu_nI_{n+1}\dots I_N+I_n\mu_{n-1}\mu_n)$ operations, and for $n=N-1$, forming $\mathcal{Q}_N$ amends $2\mu_{N-2}\mu_{N-1}I_{N-1}I_N$ operations.

Hence, if for $n=1,2,\dots,N-1$, when $\mu_n<L_n<L_n'<\min\{\mu_{n-1}I_n,I_{n+1}\dots I_N\}$, then Algorithm 6 needs

$$\sum_{n=1}^{N}((L_n+1)\mu_{n-1}I_n+(I_n+1)\mu_{n-1}\mu_n+L_n^2+1)\cdot\Theta(I_{n+1}\dots I_N)$$

operations to finding an approximate TT decomposition $\widehat{\mathcal{A}}$ to $\mathcal{A}$ with a given TT-rank $\{\mu_1,\mu_2,\dots,\mu_{N-1}\}$.

It is shown in Algorithm 6 that for each n, we have $\mathbf{A}_n\in\mathbb{R}^{\mu_{n-1}I_n\times I_{n+1}\dots I_N}$. From (Hochstenbach and Reichel (2010)), one has

$$\sum_{i_n=\mu_n+1}^{I_n}(\sigma_{i_n}(\mathbf{A}_n))^2\leq\sum_{i_n=\mu_n+1}^{I_n}(\sigma_{i_n}(\mathbf{A}_{(n)}))^2,$$

and $\|\mathbf{A}_n\|_F\leq\|\mathcal{A}\|_F$ with $n=1,2,\dots,N-1$. Hence, by combining these two facts with Corollary 9 and (6), we deduce the upper bound for $\|\mathcal{A}-\widetilde{\mathcal{A}}\|_F^2$, when $\widetilde{\mathcal{A}}=\mathcal{Q}_1\times_2^1\mathcal{Q}_2\times_3^1\dots\times_3^1\mathcal{Q}_N$ is obtained by Algorithm 6.

Theorem 21 *For each n, let μ_n, q_n, I_n, L_n' and L_n be positive integers such that $\mu_n<L_n<L_n'<\min\{\mu_{n-1}I_n,(I_{n+1}\dots I_N)/q_n\}$, $I_{n+1}\dots I_N$ is a multiple of q_n and*

$$q_nL_n'\geq\frac{12(\mu_{n-1}^2I_n^2+\mu_{n-1}I_n)}{\epsilon_n^2\delta_n},\quad L_n'=\left\lceil\mu_{n-1}I_n+\frac{3\mu_{n-1}I_n}{\epsilon_n\delta_n}\right\rceil,\quad L_n=\Omega((\tau_n+\log(1/\delta_n))/\epsilon_n^2),$$

with $\epsilon_n,\delta_n,\tau_n\in(0,1/2)$ and $\mu_{n-1}I_n<I_{n+1}\dots I_N$. For the case of Algorithm 3 with SpEmb+Gaussian, when the tensor $\widehat{\mathcal{A}}\in\mathbb{R}^{I_1\times I_2\times\dots\times I_N}$ is obtained by applying Algorithm 6 to $\mathcal{A}\in\mathbb{R}^{I_1\times I_2\times\dots\times I_N}$, then with probability at least $1-\sum_{n=1}^{N-1}(\delta_n+\exp(-\epsilon_n^2/2))$, we have

$$\|\mathcal{A}-\widehat{\mathcal{A}}\|_F^2\leq\sum_{n=1}^{N-1}\left(\sum_{i_n=\mu_n+1}^{I_n}(\sigma_{i_n}(\mathbf{A}_{(n)}))^2+T_n\|\mathcal{A}\|_F^2\right),$$

where for each n, the coefficient T_n is given by

$$T_n=2\mu_n\left(\epsilon_n+\epsilon_nI_{n+1}\dots I_N\left(1+\frac{1}{\tau_n}\right)\right)+\frac{4\mu_{n-1}I_n\mu_n(\sqrt{L_n'}+\sqrt{L_n}+\epsilon_n)^2}{q_nL_n}.$$

Remark 22 *In the above descriptions, we have discussed the computational complexity and theoretical results for Algorithm 4, 5, and 6 under the case that appling Algorithm 3 with SpEmb+Gaussian to Problems 2, 3 and 5. Similarly, we can also analyze the computational complexity and theoretical properties for these three algorithms based on Algorithm 3 with SpEmb+SRHT, which is left to the interested readers.*

5. Numerical examples

In this section, we use the numerical computation software MATLAB R2023b to develop computer programs and implement the calculations on a desktop computer (Dell Precision 3460) with an Intel Core i7-10700 CPU (2.90GHz) and 64GB RAM (63.5G usable). We set MATLAB maxNumCompThreads to 1 and use "tic" and "toc" to measure running time when applying all of the algorithms to the test matrices and tensors. The CPU time is measured in seconds. We run each algorithm 10 times and take the average result.

For the low-rank approximation, the competing algorithms include

1. FD: see Algorithm 1;
2. SpFD, GaussianFD and DctFD: Algorithm 2 with any matrix $\mathbf{G}_i$ being a SpEmb matrix, a standard Gaussian matrix and a SRDCT matrix, respectively;
3. SFD and GA-BKIFD: two randomized FD algorithm by using standard Gaussian matrices in the power iteration step (Ghashami et al. (2016a)) and the block Krylov iteration step (Wang et al. (2023)), respectively;
4. Gaussian-SpFD and Dct-SpFD: Algorithm 3 with any matrix $\mathbf{G}_i$ being a SpEmb matrix and a SRDCT matrix, respectively.

Suppose that I_1, I_2 and μ are positive integers such that $\mu < \min\{I_1, I_2\}$. For a given matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$, the relative error (RE) is defined as

$$\text{RE} = \|\mathbf{A} - \widetilde{\mathbf{A}}_\mu\|_F^2 / \|\mathbf{A}\|_F,$$

where $\widetilde{\mathbf{A}}_\mu$ is the rank-μ approximation of $\mathbf{A}$ and obtained by the FD algorithm and its variants.

5.1 Classes of test matrices

We list four test matrices from synthetic and real databases. In detail, following the setting in (Ghashami et al. (2016b); Liberty (2013); Teng and Chu (2018)), we generate the first test matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ as $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{S}^\top + \mathbf{N}/\zeta$, where $\mathbf{U} \in \mathbb{R}^{I_1 \times K}$ is an orthonormal matrix which contains a random K dimensional subspace in $\mathbb{R}^{I_1}$, $\mathbf{D} \in \mathbb{R}^{K \times K}$ is a diagonal matrix with $d_{ii} = 1 - (i-1)/K$, each entry of $\mathbf{S} \in \mathbb{R}^{I_2 \times K}$ is a standard Gaussian random variable, and $\mathbf{N} \in \mathbb{R}^{I_1 \times I_2}$ is a noisy matrix with each entry being a standard Gaussian random variable. Here we set $\zeta = 10$ and $K = 50$.

The second one is a sparse nonnegative matrix (cf. Sorensen and Embree (2016)) $\mathbf{B} \in \mathbb{R}^{I_1 \times I_2}$, defined as

$$\mathbf{B} = \sum_{j=1}^{200} \frac{1000}{j} \mathbf{x}_j \mathbf{y}_j^\top + \sum_{j=201}^{400} \frac{1}{j} \mathbf{x}_j \mathbf{y}_j^\top$$

where $\mathbf{x}_j \in \mathbb{R}^{I_1}$ and $\mathbf{y}_j \in \mathbb{R}^{I_2}$ are sparse vectors with random nonnegative entries (in MATLAB, $\mathbf{x}_j = \text{sprand}(\text{I}_1, 1, 0.025)$ and $\mathbf{y}_j = \text{sprand}(\text{I}_2, 1, 0.025)$). For these two matrices from synthetic databases, we also set $I_1 = 4000$ and $I_2 = 40000$.

We consider two real world databases: CIFAR-10 (Krizhevsky et al. (2009)) and MNIST-all (LeCun et al. (1998)). The CIFAR-10 database consists of 60000 32×32 colour images

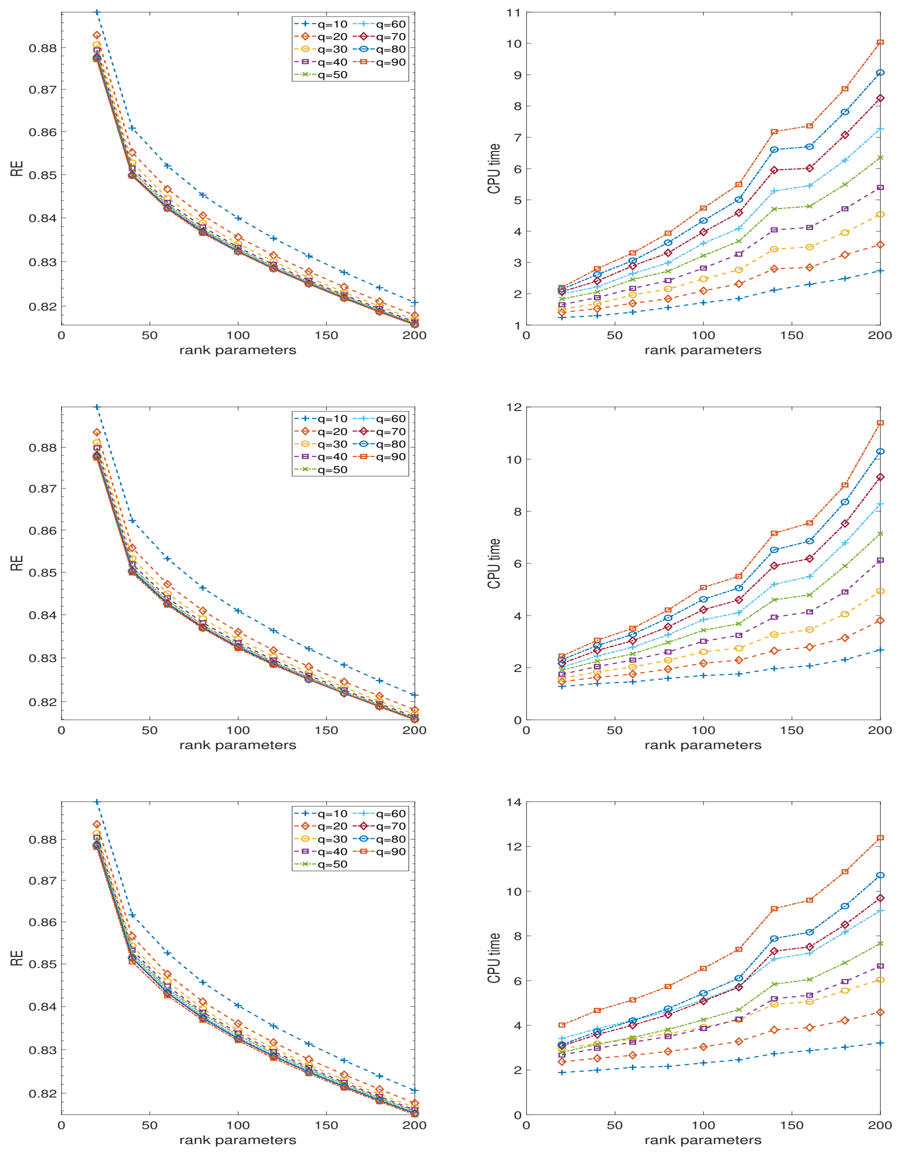


Figure 1: For μ being from 20 to 400 with step 20, the diagrams show results of applying GaussianFD (top row), SpFD (middle row) and DctFD (Bottom row) to the matrix $\mathbf{A}$ with different number of blocks q.

in 10 classes, with 6000 images per class, which is collected into a nonnegative matrix of size 3072×60000. The MNIST database contains 60,000 training images and 10,000 test images of size 28×28. Each image has 8-bit gray-scales. This database can be collected into a nonnegative matrix of size 784×70000.

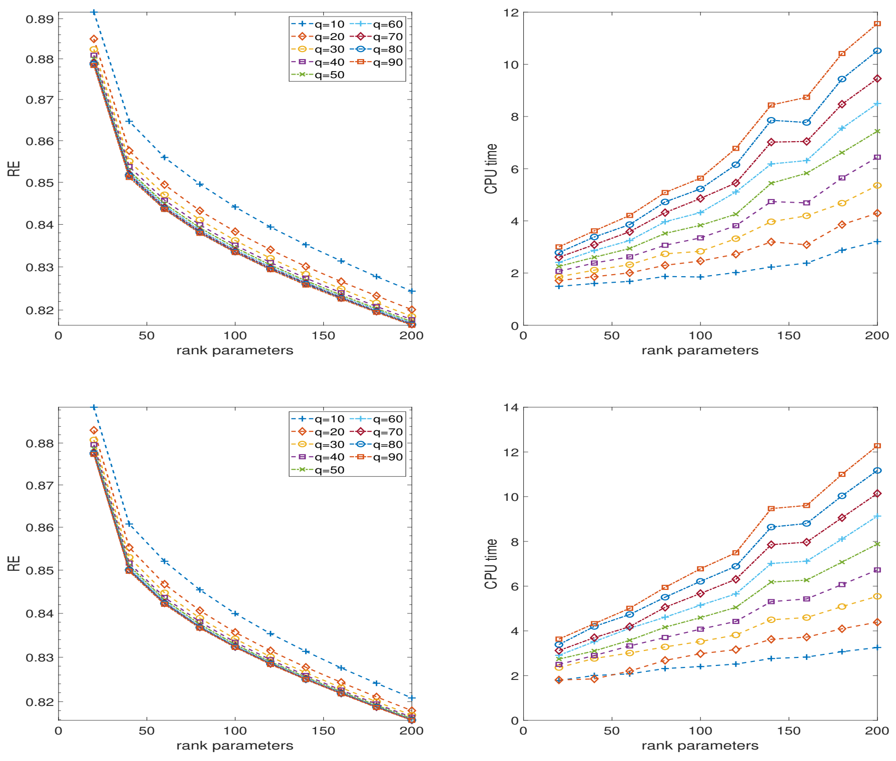


Figure 2: For μ being from 20 to 400 with step 20, the diagrams show results of applying Gaussian-SpFD (top row) and Dct-SpFD (bottom row) to the matrix $\mathbf{A}$ with different number of blocks q.

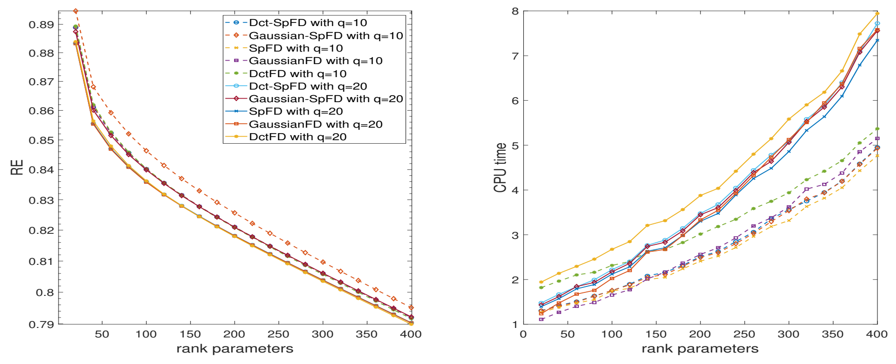


Figure 3: With different rank parameter μ, the diagrams show results of applying SpFD, GaussianFD, DctFD, Gaussian-SpFD and Dct-SpFD to the matrix $\mathbf{A}$ with $q = 10$ and $q = 20$.

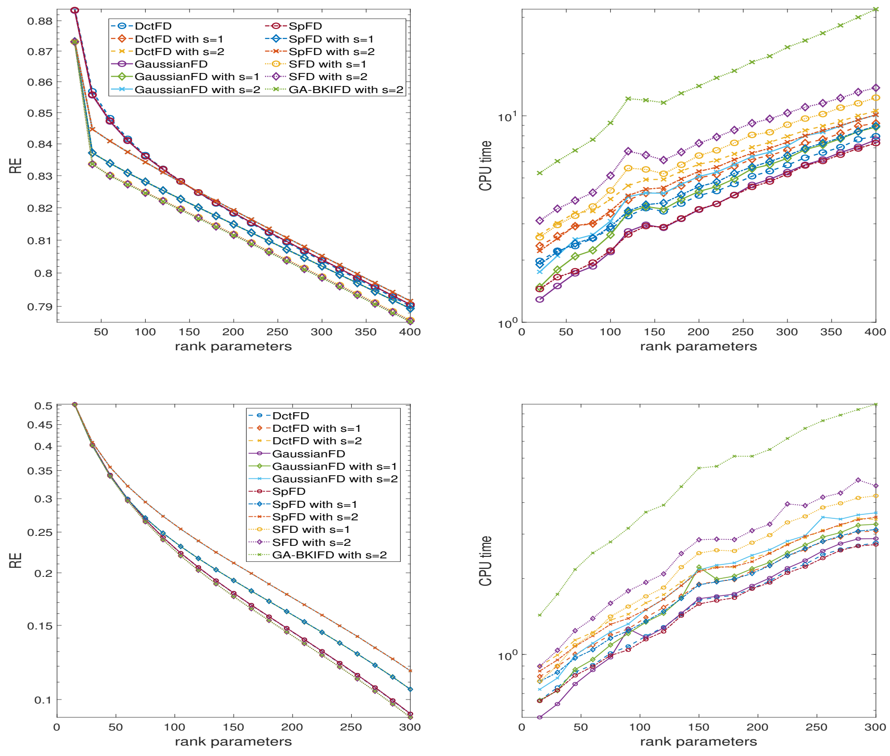


Figure 4: With different rank parameter μ, the diagrams show results of applying the power scheme versions of SpFD, GaussianFD, DctFD and SFD, and GA-BKIFD to the matrix $\mathbf{A}$ (top row) and the MNIST-all database (bottom row) with different number of subspace iterations.

5.2 The matrix case

For SpFD, Teng and Chu (2018) employed three different q for comparison: $q = 5$, $q = 10$ and $q = 50$. For Algorithm 2 with all the matrices $\mathbf{G}_i$ being SRHT matrices, the values of q be set to $\{\lceil I_2/I_1\rceil, \lceil I_2/(2I_1)\rceil, \lceil I_2/(4I_1)\rceil, \lceil I_2/(8I_1)\rceil\}$ in (Chen et al. (2017)). For the FD algorithm and its randomized variants, all experiments are based on fixing the desired rank parameter μ and varying the sketching size $L \geq \mu$ (cf. Chen et al. (2017); Teng and Chu (2018); Wang et al. (2023)). As shown in (Aizenbud and Averbuch (2019); Aizenbud et al. (2016); Halko et al. (2011)), the numerical examples are based on varying the desired rank parameter μ and setting $L = \mu + \mathrm{OV}$, where OV is a oversampling parameter. For FD, SpFD, GaussianFD, DctFD, SFD, GA-BKIFD, Gaussian-SpFD and Dct-SpFD, we set $L = \mu + 50$, where μ is any desired rank parameter. For Gaussian-SpFD and Dct-SpFD, we also set $L' = \mu + 200$. Note that the choices of L and L' are consistent with the choices of k_1 and l_1 in (Aizenbud et al. (2016)).

Example 1 *The main purpose of this example is to discuss a reasonable value for the number of blocks q in Algorithms 2 and 3. We first illustrate the accuracy of Algorithms*

2 and 3 with different blocks q via the test matrix $\mathbf{A}$. We also compare the efficiency and accuracy of Algorithms 2 and 3 with two choices of q (that is, $q = 10$ and $q = 20$).

By varying the number of blocks q, when applying SpFD, GaussianFD, DctFD, Gaussian-SpFD and Dct-SpFD to the matrix $\mathbf{A}$ with different μ from 20 to 400 with step 20, the related results are shown in Figures 1 and 2. These two figures illustrate that for SpFD, GaussianFD, DctFD, Gaussian-SpFD and Dct-SpFD, the case of small q is faster than the case of large q, but the case of large q is better than the case of large q, and for the same q, the accuracy of SpFD, GaussianFD, DctFD, Gaussian-SpFD and Dct-SpFD are comparable.

From Figure 3, on the one hand, for SpFD, GaussianFD, DctFD, Gaussian-SpFD and Dct-SpFD, the case of $q = 10$ is faster than the case of $q = 20$, and on the other hand, the values of RE for Gaussian-SpFD and Dct-SpFD with $q = 10$ are similar to that for SpFD, GaussianFD, DctFD, Gaussian-SpFD and Dct-SpFD with $q = 20$, are better than that for SpFD, GaussianFD and DctFD with $q = 10$. Hence, according to Figures 2 and 3, for SpFD, GaussianFD and DctFD, we set $q = 20$, and for Gaussian-SpFD and Dct-SpFD, we set $q = 10$.

Example 2 *As shown in Remark 4, we can obtain three more efficient algorithms by combining Algorithm 2 and the power scheme. For clarity, we use s to denote the number of subspace iterations. For SpFD, GaussianFD, DctFD and SFD, we consider two choices of s (that is, $s = 1$ and $s = 2$), and for GA-BKIFD, the number of subspace iterations is set to $s = 2$. In this example, we compare the efficiency and accuracy of these algorithms on the matrix $\mathbf{A}$ and the MNIST-all database.*

The values of μ for $\mathbf{A}$ are the same as that in Example 1. For the MNIST-all database, the values of μ are set from 15 to 300 with step 15. For The related results are shown in Figure 4. From this figure, the accuracy of SpFD, GaussianFD, DctFD, SFD and GA-BKIFD with $s > 0$ are better than that of SpFD, GaussianFD, DctFD and SFD with $s = 0$. Meanwhile, for each algorithm, the running time for the case of large s is larger than that for the case of small s.

Example 3 *In particular, when $q = 1$, Algorithm 2 is a random projection method for the low-rank matrix approximation, in which the random projection matrix is any SpEmb matrix, any standard Gaussian matrix or any SRHT/SRDCT matrix.*

We compare the accuracy and efficiency of FD, SpEmb, Gaussian, SRDCT, SpFD, GaussianFD, DctFD, Gaussian-SpFD and Dct-SpFD via two synthetic matrices and two real world databases.

For the matrix $\mathbf{A}$ and the MNIST-all database, the values of μ are the same as that in Example 2. For the matrix $\mathbf{B}$, the values of μ are set from 20 to 400 with step 20. For the CIFAR-10 database, the values of μ are set from 50 to 1000 with step 50. From Figures 5 and 6, we illustrate that i) SpEmb, Gaussian and SRDCT are faster but worse than FD, SpFD, GaussianFD, DctFD, Gaussian-SpFD and Dct-SpFD; ii) the accuracy of Gaussian-SpFD and Dct-SpFD is comparable to and slightly less than that of FD, SpFD, GaussianFD and DctFD; and iii) in terms of running time, Gaussian-SpFD and Dct-SpFD are faster than SpFD, GaussianFD and DctFD.

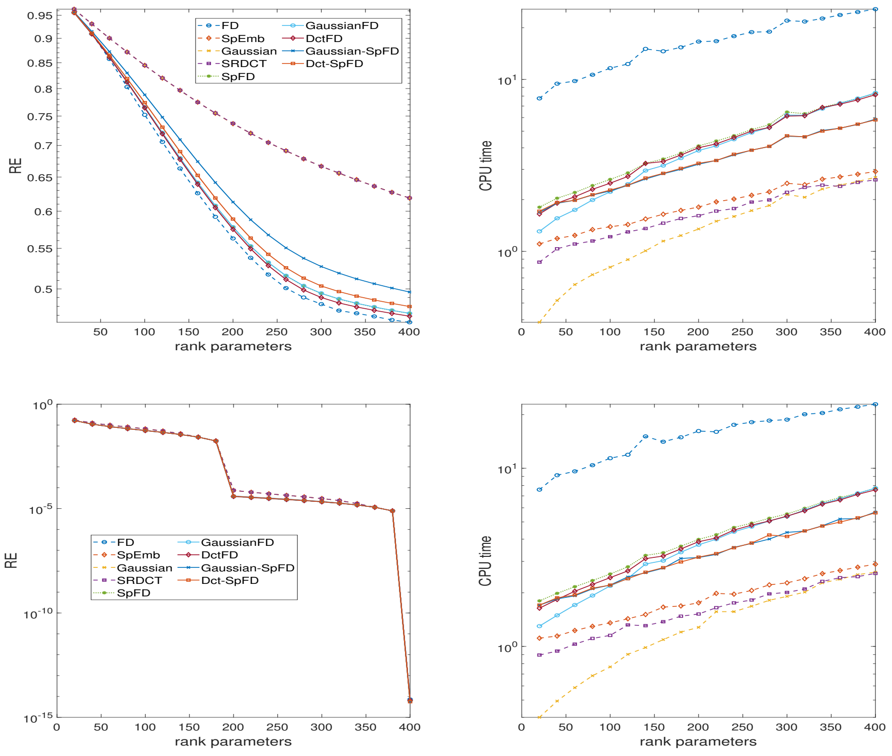


Figure 5: With different rank parameter μ, the diagrams show results of applying FD, SpEmb, Gaussian, SRDCT, SpFD, GaussianFD, DctFD, Gaussian-SpFD and Dct-SpFD to the matrices $\mathbf{A}$ (top row) and $\mathbf{B}$ (bottom row).

5.3 The Tucker decomposition case

For an approximate Tucker decomposition with a given Tucker-rank, we compare the efficiency and accuracy of Algorithm 5 with several existing algorithms through several test tensors. The competing algorithms include Tucker-ALS (De Lathauwer et al. (2000b)), Tucker-TS and Tucker-TTMTS (Malik and Becker (2018)), Sketch-Tucker-ALS (Ma and Solomonik (2021)), Randomized-Tucker-ALS, RP-HOSVD (Algorithm 2 in Zhou et al. (2014) or Algorithm 6 in Ahmadi-Asl et al. (2021)), R-PET (Algorithm 4.3 in Sun et al. (2020) or Algorithm 9 in Ahmadi-Asl et al. (2021)), T-HOSVD (Kolda and Bader (2009)), ST-HOSVD (Vannieuwenhoven et al. (2012)), randomized T-HOSVD (R-T-HOSVD) (Minster et al. (2020)), randomized ST-HOSVD (R-ST-HOSVD) (Minster et al. (2020)), and rSTHOSVDkron/rHOSVDkronreuse (Minster et al. (2024)).

Note that RP-HOSVD, R-T-HOSVD, rSTHOSVDkron, rHOSVDkronreuse, Sketch-Tucker-ALS, and Algorithm 5 are implemented by MATLAB codes, the MATLAB codes for Tucker-TS/TTMTS can be found in https://github.com/OsmanMalik/tucker-tensorsketch, HOOI is implemented by the function tucker_als (Bader and Kolda (April 5, 2021)), T-HOSVD and ST-HOSVD are implemented by the function mlsvd (Vervliet et al. (2016)) with different parameters, and R-ST-HOSVD is implemented by the mlsvd_rsi (Vervliet

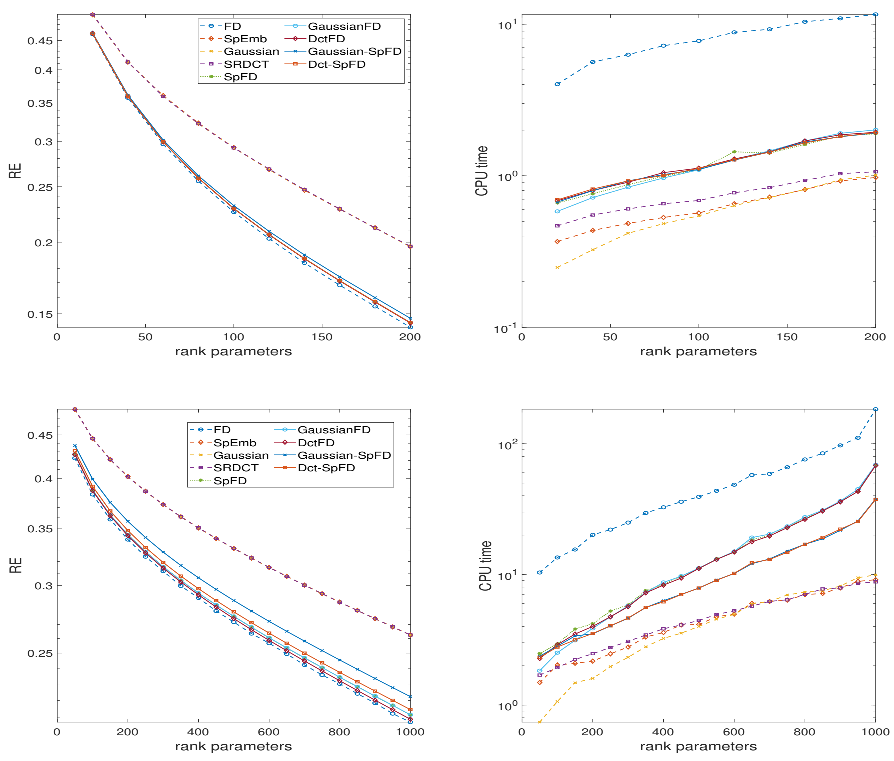


Figure 6: With different rank parameter μ, the diagrams show results of applying FD, SpEmb, Gaussian, SRDCT, SpFD, GaussianFD, DctFD, Gaussian-SpFD and Dct-SpFD to the MNIST-all database (top row) and the CIFAR-10 database (bottom row).

et al. (2016)). We also compare Algorithm 5 with RP-HOSVD, R-PET, rSTHOSVDkron, rHOSVDkronreuse, R-T-HOSVD and R-ST-HOSVD under subspace iterations. When we set the number of subspace iterations to ($s = 1$), then these algorithms are denoted by RP-HOSVD ($s = 1$), R-PET ($s = 1$), rSTHOSVDkron ($s = 1$), rHOSVDkronreuse (s=1), R-T-HOSVD ($s = 1$) and R-ST-HOSVD ($s = 1$), respectively.

For clarity, in Algorithms 4 and 5, we set $L_n = \mu_n + 10$, $L'_n = \mu_n + 100$ and $q_n = q$ with $n = 1, 2, \ldots, N$. For all algorithms, the factor matrices are orthonormal and for ST-HOSVD and their randomized variants, the processing order is $\{1, 2, \ldots, N\}$. We then define the relative error (RE) as

$$\mathrm{RE} = \|\mathcal{A} - \widetilde{\mathcal{A}}\|_F / \|\mathcal{A}\|_F$$

with $\widetilde{\mathcal{A}} = \mathcal{A} \times_1 (\mathbf{Q}_1\mathbf{Q}_1^\top) \times_2 (\mathbf{Q}_2\mathbf{Q}_2^\top) \cdots \times_N (\mathbf{Q}_N\mathbf{Q}_N^\top)$, where for N given positive integers $\mu_n < I_n$, an optimal solution $\{\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_N\}$ for Problem 1 is obtained from by applying the numerical algorithms to $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$.

Example 4 *Note that in Algorithms* 4 *and* 5*, we can also use Algorithm* 1 *or* 2 *to obtain the mode-n factor matrix* $\mathbf{Q}_n$*. For example, we use the term Algorithm* 4 *with FD to denote the case of using the FD algorithm to obtain the mode-n factor matrix* $\mathbf{Q}_n$ *in Algorithm*

4. *We now compare the efficiency and accuracy of these cases via the following two test tensors.*

The first one is a sparse tensor $\mathcal{A} \in \mathbb{R}^{800\times 800\times 800}$ (cf. Ahmadi-Asl et al. (2021); Minster et al. (2020); Saibaba (2016)), *which is given by*

$$\mathcal{A} = \sum_{i=1}^{200} \frac{\gamma}{i} \mathbf{x}_i \circ \mathbf{y}_i \circ \mathbf{z}_i + \sum_{i=201}^{800} \frac{1}{i} \mathbf{x}_i \circ \mathbf{y}_i \circ \mathbf{z}_i$$

where $\mathbf{x}_i$, $\mathbf{y}_i$ *and* $\mathbf{z}_i$ *are sparse vectors in* $\mathbb{R}^{800}$ *with only 0.05 sparsity. Here we set* $\gamma = 1000$.

The second tensor $\mathcal{B} \in \mathbb{R}^{800\times 800\times 800}$ *is given as* $\mathcal{A} = \mathcal{D} \times_1 \mathbf{S} \times_2 \mathbf{U} \times_3 \mathbf{V} + \mathcal{N}/\zeta$, *where each entry of* $\mathbf{S} \in \mathbb{R}^{800\times 200}$ *is a standard Gaussian random variable,* $\mathcal{D} \in \mathbb{R}^{200\times 200\times 200}$ *is a diagonal tensor with* $d_{iii} = 1 - (i-1)/200$, *two orthonormal matrices* $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{800\times 200}$ *contain a random* 200 *dimensional subspace in* $\mathbb{R}^{800}$, *and* $\mathcal{N} \in \mathbb{R}^{800\times 800\times 800}$ *is a noisy tensor with each entry being a standard Gaussian random variable. Here we set* $\zeta = 10$.

For the tensors $\mathcal{A}$ and $\mathcal{B}$, we set the Tucker-rank $\{\mu, \mu, \mu\}$ as $\{200, 200, 200\}$. Hence, for FD, SpFD, GaussianFD and DctFD, we have $L = \mu + 10$ with $L_n = L$, and for Gaussian-SpFD and Dct-SpFD, we have $L : L_n = \mu + 10$ and $L' : L'_n = \mu + 100$ with $n = 1, 2, 3$. With different number of blocks q, the related results are shown in Table 2.

According to Table 2, we illustrate that the accuracy of Algorithms 4 and 5 with FD, SpFD, DctFD, GaussianFD, Dct-SpFD and Gaussian-SpFD is comparable, and Algorithm 4 is slower than Algorithm 5. In the rest, we set $q = 20$ for Algorithm 5 with SpFD, DctFD and GaussianFD, and $q = 10$ for Algorithm 5 with Dct-SpFD and Gaussian-SpFD.

Example 5 *We now consider the efficiency and accuracy of Algorithm 5 with Tucker-ALS, T-HOSVD, ST-HOSVD and their randomized variants in image data compression. Three databases are used in this example.*

The extended Yale B database[2] (Georghiades et al. (2001)) *contains* 560 *images with each image containing* 480×640 *pixels in the gray-scale range. This data is extracted into a tensor* $\mathcal{A}_{\text{YaleB}} \in \mathbb{R}^{480\times 640\times 560}$.

The Columbia object image library COIL-100[3] *consists of* 7200 *color images that contain 100 objects under 72 different rotations. Each image has* 128×128 *pixels in the RGB color range. We reshape this data to a third-order tensor* $\mathcal{A}_{\text{COIL}}$ *of size* $1024 \times 1152 \times 300$.

In the Washington DC Mall database[4], *the sensor system used in this case measured pixel response in 210 bands in the 0.4 to 2.4 um region of the visible and infrared spectrum. Bands in the 0.9 and 1.4 um region where the atmosphere is opaque have been omitted from the data set, leaving 191 bands. The data set contains 1280 scan lines with 307 pixels in each scan line. This database is collected as a tensor* $\mathcal{A}_{\text{DCmall}}$ *of size* $1280 \times 307 \times 191$.

When we compare the proposed algorithms with Tucker-ALS and its randomized variants, we set the Tucker-rank parameter $\{\mu_1, \mu_2, \mu_3\}$ for Yale, COIL-100 and Washington DC

2. The extended Yale Face Database B is at `http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/ExtYaleB.html`.
3. COIL-100 can be downloaded from `https://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php`.
4. The Washington DC Mall database is at `https://engineering.purdue.edu/biehl/MultiSpec/hyperspectral.html`.

Tensor	Algorithms	10		20		30	
		RE	time (s)	RE	time (s)	RE	time (s)
$\mathcal{A}$	Algorithm 4 with FD	3.84e-5±7.14e-21	226.57±2.67	3.84e-5±8.03e-21	224.46±1.65	3.84e-5±7.45e-21	228.63±6.96
	Algorithm 5 with FD	3.84e-5±7.13e-21	106.09±4.36	3.84e-5±7.14e-21	101.56±1.92	3.84e-5±7.14e-21	103.98±5.20
	Algorithm 4 with SpFD	4.27e-5±2.02e-8	13.91±2.32	4.05e-5±1.22e-8	16.51±2.62	3.98e-5±9.56e-9	16.76±2.04
	Algorithm 5 with SpFD	4.03e-5±1.02e-8	6.54±0.20	3.93e-5±5.49e-9	6.97±0.27	3.90e-5±4.73e-9	8.00±0.29
	Algorithm 4 with DctFD	4.27e-5±2.28e-8	18.57±2.70	4.05e-5±1.09e-8	19.71±1.96	3.98e-5±8.70e-9	21.85±3.93
	Algorithm 5 with DctFD	4.03e-5±1.50e-8	8.84±0.84	3.93e-5±4.40e-9	9.26±0.78	3.90e-5±4.00e-9	10.01±0.36
	Algorithm 4 with GaussianFD	4.27e-5±1.50e-8	30.48±8.65	4.05e-5±6.07e-9	27.14±5.01	3.98e-5±7.60e-9	38.54±2.57
	Algorithm 5 with GaussianFD	4.03e-5±1.30e-8	11.27±0.41	3.93e-5±4.78e-9	11.11±0.47	3.90e-5±4.12e-9	16.79±0.51
	Algorithm 4 with Dct-SpFD	4.64e-5±4.40e-8	16.73±3.07	4.65e-5±2.47e-8	15.29±2.57	4.65e-5±3.49e-8	14.70±3.49
	Algorithm 5 with Dct-SpFD	4.22e-5±2.53e-8	7.03±0.56	4.21e-5±2.42e-8	6.29±0.47	4.21e-5±3.32e-8	6.43±0.53
	Algorithm 4 with Gaussian-SpFD	4.27e-5±2.30e-8	14.37±1.34	4.27e-5±1.43e-8	14.14±2.14	4.27e-5±2.25e-8	13.96±1.76
	Algorithm 5 with Gaussian-SpFD	4.03e-5±9.70e-9	6.76±0.49	4.03e-5±1.45e-8	6.24±0.47	4.03e-5±9.38e-9	6.36±0.45
$\mathcal{B}$	Algorithm 4 with FD	7.22e-1±1.17e-16	268.99±4.20	7.21e-1±1.17e-16	269.51±2.11	7.22e-1±1.17e-16	269.68±8.66
	Algorithm 5 with FD	7.10e-1±1.17e-16	118.75±4.78	7.10e-1±1.17e-16	118.12±2.12	7.10e-1±1.17e-16	121.00±7.55
	Algorithm 4 with SpFD	7.72e-1±2.28e-4	15.11±3.55	7.51e-1±2.16e-4	17.25±3.58	7.43e-1±1.22e-4	16.33±2.68
	Algorithm 5 with SpFD	7.37e-1±3.37e-4	6.34±0.31	7.25e-1±1.59e-4	7.42±0.47	7.21e-1±1.38e-4	8.51±0.77
	Algorithm 4 with DctFD	7.72e-1±2.73e-4	19.17±2.67	7.51e-1±2.25e-4	21.07±3.19	7.43e-1±1.57e-4	21.56±6.47
	Algorithm 5 with DctFD	7.37e-1±1.84e-4	8.88±0.81	7.25e-1±8.75e-5	9.48±0.92	7.21e-1±9.78e-5	10.20±0.85
	Algorithm 4 with GaussianFD	7.72e-1±2.25e-4	29.12±6.58	7.51e-1±1.47e-4	27.83±4.91	7.42e-1±1.30e-4	39.14±7.59
	Algorithm 5 with GaussianFD	7.37e-1±1.79e-4	11.22±0.37	7.25e-1±2.11e-4	11.33±0.49	7.21e-1±1.12e-4	17.08±0.87
	Algorithm 4 with Dct-SpFD	8.07e-1±3.63e-4	15.32±2.08	8.07e-1±3.66e-4	14.95±2.47	8.07e-1±2.62e-4	18.07±3.57
	Algorithm 5 with Dct-SpFD	7.60e-1±2.73e-4	6.64±0.40	7.60e-1±2.88e-4	6.59±0.65	7.60e-1±3.92e-4	6.86±0.93
	Algorithm 4 with Gaussian-SpFD	7.72e-1±1.37e-4	14.88±2.48	7.72e-1±3.16e-4	14.67±2.74	7.72e-1±2.25e-4	15.57±3.52
	Algorithm 5 with Gaussian-SpFD	7.37e-1±2.73e-4	6.67±0.56	7.37e-1±2.36e-4	6.46±0.51	7.37e-1±1.87e-4	6.51±0.61

Table 2: For the tensors $\mathcal{A}$ and $\mathcal{B}$ in Example 4, with different number of blocks q, the values of the means and standard deviations for RE and running time obtained from Algorithms 4 and 5 by using different algorithms to compute all the factor matrices.

as $\{50, 50, 50\}$, $\{30, 30, 15\}$ and $\{40, 30, 20\}$, respectively. When we compare the proposed algorithms with T-HOSVD, ST-HOSVD and their randomized variants, we set the Tucker-rank parameter $\{\mu_1, \mu_2, \mu_3\}$ for Yale, COIL-100 and Washington DC as $\{200, 200, 200\}$, $\{450, 450, 120\}$ and $\{500, 100, 50\}$, respectively. By using Algorithm 5, Tucker-ALS and its randomized variants to the tensors from these three databases, the related results are shown in Table 3, which illustrates that, Algorithm 5 with SpFD, DctFD, GaussianFD, Gaussian-SpFD and Dct-SpFD is comparable to and faster than Algorithm 5 with FD, Tucker-ALS and its randomized variants.

Algorithms	Yale		COIL-100		Washington DC	
	RE	time (s)	RE	time (s)	RE	time (s)
Algorithm 5 with FD	2.17e-1	16.17	4.53e-1	18.11	2.41e-1	3.43
Algorithm 5 with SpFD	2.17e-1	1.60	4.53e-1	2.55	2.44e-1	0.56
Algorithm 5 with DctFD	2.16e-1	2.82	4.50e-1	5.37	2.43e-1	1.87
Algorithm 5 with GaussianFD	2.18e-1	1.64	4.53e-1	2.36	2.44e-1	0.58
Algorithm 5 with Gaussian-SpFD	2.26e-1	1.48	4.63e-1	2.47	2.52e-1	0.53
Algorithm 5 with Dct-SpFD	2.22e-1	1.57	4.57e-1	2.48	2.48e-1	0.58
Tucker-ALS	3.45e-1	1.06	4.34e-1	6.43	2.30e-1	1.54
Randomized-Tucker-ALS	3.61e-1	6.12	4.48e-1	26.12	2.36e-1	3.51
Tucker-TS	4.14e-1	501.03	5.10e-1	2.04e+3	2.72e-1	3.56e+4
Tucker-TTMTS	6.65e-1	9.34	7.23e-1	32.54	5.99e-1	67.72
Sketch-Tucker-ALS	3.79e-1	60.15	4.58e-1	137.01	2.72e-1	65.21

Table 3: Numerical simulation results of the FD-based algorithms (i.e., Algorithm 5 with FD, SpFD, DctFD, GaussianFD, Gaussian-SpFD and Dct-SpFD) with Tucker-ALS, randomized Tucker-ALS, Tucker-TS, Tucker-TTMTS, and Sketch-Tucker-ALS to the tensors from three real databases.

By using Algorithm 5, T-HOSVD, ST-HOSVD and their randomized variants to the tensors from these three databases, the related results are shown in Table 4. According to this table, in terms of RE, Algorithm 5 with SpFD, DctFD, GaussianFD, Gaussian-SpFD and Dct-SpFD is comparable to Algorithm 5 with FD, T-HOSVD, ST-HOSVD and randomized T-HOSVD/ST-HOSVD with $s = 1$, and better than randomized T-HOSVD/ST-HOSVD with $s = 0$, and in terms of time, Algorithm 5 with SpFD, DctFD, GaussianFD, Gaussian-SpFD and Dct-SpFD is compatible to ST-HOSVD and randomized T-HOSVD/ST-HOSVD with $s = 0$, and faster than Algorithm 5 with FD, T-HOSVD and randomized T-HOSVD/ST-HOSVD with $s = 1$.

5.4 The TT decomposition case

For the approximate TT decomposition with a given TT-rank, we compare the efficiency of Algorithm 6 and several existing algorithms, such as TT-SVD (cf. Oseledets (2011)), R-TT-SVD (randomized TT-SVD) with Gaussian (see Huber et al. (2017)), R-TT-SVD with SpEmb, R-TT-SVD with DCT, R-TT-SVD with KR-Gaussian and PSTT (see Shi et al. (2023)). Note that three other randomized variants for TT-SVD are also proposed in Che et al. (2026): R-TT-SVD with SpEmb, R-TT-SVD with DCT and R-TT-SVD with KR-Gaussian. Meanwhile, Gaussian-SpFD or Dct-SpFD used in Algorithm 6 can be replaced by GaussianFD or SpFD.

Algorithms	Yale		COIL-100		Washington DC	
	RE	time (s)	RE	time (s)	RE	time (s)
Algorithm 5 with FD	8.60e-2	62.20	2.30e-1	69.48	1.39e-1	30.73
Algorithm 5 with SpFD	8.39e-2	4.12	2.21e-1	12.65	1.11e-1	5.35
Algorithm 5 with DctFD	8.36e-2	5.83	2.21e-1	16.50	1.11e-1	7.14
Algorithm 5 with GaussianFD	8.38e-2	5.32	2.21e-1	16.38	1.11e-1	5.83
Algorithm 5 with Dct-SpFD	8.58e-2	3.33	2.26e-1	9.36	1.14e-1	3.19
Algorithm 5 with Gaussian-SpFD	8.91e-2	3.29	2.31e-1	10.16	1.18e-1	3.29
T-HOSVD	7.95e-2	25.92	2.08e-1	78.58	1.06e-1	9.64
ST-HOSVD	7.94e-2	2.99	2.08e-1	10.24	1.06e-1	1.85
R-T-HOSVD ($s = 0$)	1.34e-1	10.62	2.87e-1	52.19	1.61e-1	4.46
R-T-HOSVD ($s = 1$)	8.46e-2	12.68	2.22e-1	38.85	1.12e-1	5.39
R-ST-HOSVD ($s = 0$)	1.29e-1	5.20	2.81e-1	16.48	1.58e-1	2.79
R-ST-HOSVD ($s = 1$)	8.38e-2	9.01	2.21e-1	28.04	1.11e-1	4.72
RP-HOSVD ($s = 0$)	1.40e-1	3.46	2.94e-1	10.65	1.69e-1	1.69
RP-HOSVD ($s = 1$)	8.61e-2	10.63	2.25e-1	42.27	1.15e-1	4.02
R-PET ($s = 0$)	1.89e-1	3.97	4.02e-1	20.22	2.25e-1	2.00
R-PET ($s = 1$)	1.17e-1	11.42	3.08e-1	41.19	1.53e-1	4.87
rSTHOSVDkron ($s = 0$)	1.32e-1	2.12	3.10e-1	7.07	1.82e-1	1.22
rSTHOSVDkron ($s = 1$)	8.27e-2	6.62	2.20e-1	21.09	1.11e-1	3.03
rSTHOSVDkronreuse ($s = 0$)	1.36e-1	2.98	3.07e-1	10.12	2.05e-1	1.50
rSTHOSVDkronreuse ($s = 1$)	8.33e-2	11.38	2.21e-1	49.26	1.12e-1	4.32

Table 4: Numerical simulation results of the FD-based algorithms (i.e., Algorithm 5 with FD, SpFD, DctFD, GaussianFD, Gaussian-SpFD and Dct-SpFD) with T-HOSVD, ST-HOSVD, and several randomized variants to the tensors from three real databases.

Note that R-TT-SVD with Gaussian, SpEmb, DCT and KR-Gaussian are also discussed in (Che et al. (2026)), and the number of subspace iterations used in these R-TT-SVD is set to $s = 0$ or $s = 1$. The oversampling parameter in each R-TT-SVD is set to 10.

For a given TT-rank $\{\mu_1, \mu_2, \ldots, \mu_{N-1}\}$ with $\mu_0 = \mu_N = 1$, the relative error for $\widehat{\mathcal{A}} = \mathcal{Q}_1 \times_3^1 \mathcal{Q}_2 \times_3^1 \cdots \times_3^1 \mathcal{Q}_N$ of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is defined as

$$\mathrm{RE} = \|\mathcal{A} - \widehat{\mathcal{A}}\|_F / \|\mathcal{A}\|_F,$$

where all the $\mathcal{Q}_n \in \mathbb{R}^{\mu_{n-1} \times I_n \times \mu_n}$ are obtained from the proposed algorithms. For clarity, in Algorithm 6, we set $L_n = L$, $L_n' = L'$ and $q_n = q$ with $n = 1, 2, \ldots, N-1$.

Example 6 *The first tensor* $\mathcal{A} \in \mathbb{R}^{50 \times 50 \times 50 \times 50 \times 50}$ *is given as*

$$\mathcal{A} = \mathcal{P} + \frac{\gamma \|\mathcal{P}\|_F}{\sqrt{50^5}} \mathcal{N}$$

with $\mathcal{P} = \mathcal{G}_1 \times_2^1 \mathcal{G}_2 \times_3^1 \mathcal{G}_3 \times_3^1 \mathcal{G}_4 \times_3^1 \mathcal{G}_5$, *where the entries of* $\mathcal{G}_1 \in \mathbb{R}^{50 \times 10}$, $\mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4 \in \mathbb{R}^{10 \times 50 \times 10}$, $\mathcal{G}_5 \in \mathbb{R}^{10 \times 50}$ *and* $\mathcal{N} \in \mathbb{R}^{50 \times 50 \times 50 \times 50 \times 50}$ *are Gaussian variables with mean zero and variance unit. Here, we set* $\gamma = 10^{-4}$.

Another tensor is the same as that in Example 5, which is denoted by $\mathcal{A}_{\mathrm{Yale}}$.

For the tensor $\mathcal{A}$, we set $\{\mu, \mu, \mu, \mu\} = \{10, 10, 10, 10\}$. Then, in Algorithm 6, one has that $L_n = 20$ and $L_n' = 110$ with $n = 1, 2, 3, 4$. Let $q = 10, 20, 30$, the related results are listed in Table 5, which illustrate that for Algorithm 6 with SpFD, GaussianFD, Dct-SpFD and Gaussian-SpFD, the number of blocks q is set to 20.

q	Metric	Algorithm 6 with FD	Algorithm 6 with SpFD	Algorithm 6 with GaussianFD	Algorithm 6 with Dct-SpFD	Algorithm 6 with Gaussian-SpFD
10	RE	1.58e-2±3.66e-18	1.62e-2±2.41e-5	1.62e-2±2.78e-5	1.62e-2±2.13e-5	1.66e-2±5.45e-5
	time (s)	67.82±1.24	3.48±9.99e-2	3.82±6.35e-2	3.48±5.77e-2	3.53±7.60e-2
20	RE	1.58e-2±3.66e-18	1.60e-2±1.31e-5	1.60e-2±1.31e-5	1.60e-2±1.40e-5	1.62e-2±2.62e-5
	time (s)	68.72±1.03	3.59±1.54e-1	3.82±1.34e-1	3.63±1.51e-1	3.56±8.30e-2
30	RE	1.58e-2±3.66e-18	1.59e-2±7.03e-6	1.59e-2±8.36e-6	1.59e-2±5.94e-6	1.61e-2±1.83e-5
	time (s)	68.72±7.19e-1	3.56±9.91e-2	3.84±1.07e-1	3.67±7.01e-2	3.55±7.80e-2

Table 5: For the test tensor $\mathcal{A}$ in Example 6, with different number of blocks q, the values of the means and standard deviations for RE and running time obtained from Algorithm 6 by using different algorithms to compute all the core tensors.

Algorithms	$\mathcal{A}$		$\mathcal{A}_{\text{Yale}}$	
	RE	time (s)	RE	time (s)
Algorithm 6 with FD	1.58e-2	77.09	7.09e-2	43.03
Algorithm 6 with SpFD	1.60e-2	3.82	7.67e-2	42.58
Algorithm 6 with GaussianFD	1.60e-2	3.97	7.66e-2	40.78
Algorithm 6 with Dct-SpFD	1.62e-2	3.74	7.78e-2	21.79
Algorithm 6 with Gaussian-SpFD	1.66e-2	3.67	8.03e-2	21.10
TT-SVD	1.58e-2	26.51	6.96e-2	9.75
R-TT-SVD with Gaussian ($s = 0$)	1.46e-1	1.23	1.17e-1	1.87
R-TT-SVD with Gaussian ($s = 1$)	1.58e-2	4.71	7.47e-2	5.07
R-TT-SVD with SpEmb ($s = 0$)	2.86e-1	1.27	1.22e-1	1.45
R-TT-SVD with SpEmb ($s = 1$)	1.58e-2	4.87	7.60e-2	4.73
R-TT-SVD with DCT ($s = 0$)	1.44e-1	9.99	1.17e-1	3.30
R-TT-SVD with DCT ($s = 1$)	1.58e-2	13.38	7.48e-2	6.59
R-TT-SVD with KR-Gaussian ($s = 0$)	1.96e-1	0.47	1.17e-1	1.49
R-TT-SVD with KR-Gaussian ($s = 1$)	1.58e-2	3.96	7.47e-2	4.70
PSTT ($s = 0$)	2.11e-1	4.81	1.17e-1	3.11
PSTT ($s = 1$)	1.58e-2	12.32	7.50e-2	7.79

Table 6: Numerical simulation results of the FD-based algorithms (i.e., Algorithm 6 with FD, SpFD, GaussianFD, Gaussian-SpFD and Dct-SpFD) with TT-SVD and several randomized TT-SVD to the tensors $\mathcal{A}$ and $\mathcal{A}_{\text{Yale}}$ in Example 6.

We now compare the efficiency of Algorithm 6, TT-SVD, and all the R-TT-SVDs through two test tensors in Example 6. For the tensor $\mathcal{A}$, all the parameters in Algorithm 6 are the same as discussed above. For the tensor $\mathcal{A}_{\text{Yale}}$, we assume that the TT-rank $\{\mu, \mu\} = \{200, 200\}$. Then, we have $\{L_n, L'_n\} = \{210, 300\}$ with $n = 1, 2$ in Algorithm 6. Hence, the related results are listed in Table 6. From this table, one has that: (a) in terms of RE, Algorithm 6 with each case is comparable to TT-SVD, R-TT-SVD ($s = 1$) and PSTT ($s = 1$), and is better than R-TT-SVD ($s = 0$) and PSTT ($s = 0$); (b) for the tensor $\mathcal{A}$, Algorithm 6 with SpFD, GaussianFD, Dct-SpFD and Gaussian-SpFD are similar to each other and faster than TT-SVD, R-TT-SVD ($s = 1$) and PSTT ($s = 1$); and (c) for the tensor $\mathcal{A}_{\text{Yale}}$, Algorithm 6 with each case is slower than TT-SVD, R-TT-SVD and PSTT. Note that Algorithm 6 for each case is not suitable to calculate an approximate TT decomposition of a third-order tensor with a given TT-rank.

6. An application to classification

Gillet et al. (2023) study the capabilities of the Tucker decomposition when it is used in data mining techniques such as exploratory analysis, clustering and classification of data. In this section, We will compare the performance of the proposed algorithms in computing Tucker decomposition on two databases: COIL-20 (Nene et al. (1996)) and MNIST (Deng (2012)). The COIL-20 database gathers 20 different objects. For each object, there are 72 pictures that represent the object in a specific position (a difference of 5° in the orientation of the object). The pictures have 128×128 pixels. The MNIST database is composed of 70 000 images representing a hand-written digit, of 28×28 pixels. It is often used to evaluate algorithms of image classification, as the hand-written nature of the images induces a different complexity to deal with than a static object.

6.1 Classification process with Tucker decomposition: an overview

Using Tucker decomposition, one can classify new elements by first constructing a model based on elements with known categories. The new element is then projected into the same feature space, allowing for comparison with existing classes to identify the most appropriate match. In this scenario, Tucker decomposition is employed to construct a model using training data. To enable this, the data must be structured into a tensor that includes a dedicated dimension representing the known classes. This dimension serves to label each sub-tensor with its corresponding class, facilitating the classification process.

We now provide a detailed explanation of this process based on the MNIST database. We first build a 4-order tensor $\mathcal{A}_{\text{MNIST}} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$, with two dimensions to represent the pixels I_1 and I_2, one dimension to represent the samples I_3, and a last one for the digits written on images I_4. For a given Tucker-rank $\{\mu_1, \mu_2, \mu_3, \mu_4\}$, the approximate Tucker decomposition to $\mathcal{A}_{\text{MNIST}}$ is denoted by $\widetilde{\mathcal{A}}_{\text{MNIST}} = \mathcal{G} \times_1 \mathbf{Q}_1 \times_2 \mathbf{Q}_2 \times_3 \mathbf{Q}_3 \times_4 \mathbf{Q}_4$, where $\mathcal{G} \in \mathbb{R}^{\mu_1 \times \mu_2 \times \mu_3 \times \mu_4}$ is the core tensor, $\mathbf{Q}_1 \in \mathbb{R}^{I_1 \times \mu_1}$ and $\mathbf{Q}_2 \in \mathbb{R}^{I_2 \times \mu_2}$ are the factor matrices for the two pixel dimensions, $\mathbf{Q}_3 \in \mathbb{R}^{I_3 \times \mu_3}$ is the factor matrix for the sample dimension, and $\mathbf{Q}_4 \in \mathbb{R}^{I_4 \times \mu_4}$ is the factor matrix for the class dimension.

We consider a new image in the MNIST database, which is denoted by $\mathbf{B} \in \mathbb{R}^{I_1 \times I_2}$. We duplicate the matrix $\mathbf{B}$ I_3 times to obtain the 4-order tensor $\mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times 1}$ as

$$\mathcal{B}(:,:,i_3,1) = \mathbf{B}$$

with $i_3 = 1, 2, \ldots, I_3$, and compute

$$\mathcal{G}_{\mathbf{B}} = \mathcal{B} \times_1 \mathbf{Q}_1^\top \times_2 \mathbf{Q}_2^\top \times_3 \mathbf{Q}_3^\top.$$

To finally classify the element, we compare $\mathcal{G}_{\mathbf{B}} \in \mathbb{R}^{\mu_1 \times \mu_2 \times \mu_3 \times 1}$ with $\mathcal{G} \in \mathbb{R}^{\mu_1 \times \mu_2 \times \mu_3 \times \mu_4}$ by keeping only one class at a time in $\mathcal{G}$. To do so, for each class $i = 1, 2, \ldots, I_4$ we use the product $\mathcal{G} \times_4 \mathbf{Q}_4(:,i)^\top$ to produce a class specific core tensor $\mathcal{G}_i \in \mathbb{R}^{\mu_1 \times \mu_2 \times \mu_3 \times 1}$:

$$\mathcal{G}_i = \mathcal{G} \times_4 \mathbf{Q}_4(:,i)^\top.$$

As $\mathcal{G}_i$ and $\mathcal{G}_{\mathbf{B}}$ are now of the same size and in the same space, they can be compared with the Frobenius norm applied on the difference of the two tensors. The class for which the Frobenius norm is the lowest (i.e., for which the two tensors are the closest) can be considered as the class of the element. Hence, for any tensor $\mathcal{A}_{\text{train}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ and the corresponding element of classify $\mathcal{E} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_M}$ with $M \leq (N-1)$, the detailed process is summarized in Algorithm 7.

Algorithm 7 Classification process ((Gillet et al., 2023, Algorithm 3))

```
    Input: Training tensor $\mathcal{A}_{\text{train}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, the element of classify $\mathcal{E} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_M}$ with $M \leq (N-1)$, and the number of classes $I_N$.
    Output: Class $c$, the best matching class for $\mathcal{E}$.
 1: Initialize $N$ positive integers $\mu_n$ such that $\mu_n < \min\{I_n, I_1 \dots I_{n-1}I_{n+1}\dots I_N\}$ with $n = 1, 2, \dots, N$.
 2: Obtain $\{\mathcal{G}; \mathbf{Q}_1, \dots, \mathbf{Q}_N\}$ from $\mathcal{A}_{\text{train}}$ with a given Tucker-rank $\{\mu_1, \mu_2, \dots, \mu_N\}$.
 3: for $n = M, M+1, \dots, N-1$ do
 4:    Set the temporary tensor $\mathcal{F} \in \mathbb{R}^{I_1\times I_2\times\dots I_M\times I_{M+1}\times\cdots\times I_n}$ being the zero tensor.
 5:    for $i_n = 1, 2, \dots, I_n$ do
 6:       Form $\mathcal{F}(:, \dots, :, i_n) = \mathcal{E}$.
 7:    end for
 8:    Update $\mathcal{E} = \mathcal{F}$.
 9: end for
10: Set $\mathcal{G}_{\text{partial}} = \mathcal{E}$.
11: for $n = 1, 2, \dots, N-1$ do
12:    Compute $\mathcal{G}_{\text{partial}} = \mathcal{G}_{\text{partial}} \times_n \mathbf{Q}_n^\top$.
13: end for
14: Set best_result = max(Double).
15: for $i_N = 1, 2, \dots, I_N$ do
16:    Compute $\mathcal{G}_{i_N} = \mathcal{G} \times_N \mathbf{Q}_N(:, i_N)^\top$.
17:    Update result = $\|\mathcal{G}_{\text{partial}} - \mathcal{G}_{i_N}\|_F$.
18:    if result < best_result then
19:       Set best_result = result and $c = i_N$.
20:    end if
21: end for
```

Note that the desired Tucker-rank $\{\mu_1, \mu_2, \dots, \mu_N\}$ is one of key points to determine the quality of Algorithm 7. In practice, the Tucker-rank of $\mathcal{A}_{\text{train}}$ is unknown in advance. To estimate a suitable Tucker-rank is called the fixed precision problem for Tucker decomposition (Che et al. (2025b)): for a given $0 < \epsilon < 1$, the goal is to find N minimum positive integers μ_n and N orthonormal matrices $\mathbf{Q}_n \in \mathbb{R}^{I_n\times\mu_n}$ such that

$$\left\|\mathcal{A}_{\text{train}} - \mathcal{A}_{\text{train}} \times_1 (\mathbf{Q}_1\mathbf{Q}_1^\top) \cdots \times_N (\mathbf{Q}_N\mathbf{Q}_N^\top)\right\|_F \leq \epsilon.$$

There exist several algorithms to estimate a suitable Tucker-rank, including a greedy strategy in (Ehrlacher et al. (2021)), the rank-adaptive (RA) variant of HOOI in (Xiao and Yang (2024)), randomized Tucker via single-mode-sketching (Hashemi and Nakatsukasa (2023)), and the adaptive randomized variants of T-HOSVD and ST-HOSVD (cf. Che and Wei (2019); Che et al. (2025b)). Another key point to determine the quality of Algorithm 7 is the methods to obtain $\{\mathcal{G}; \mathbf{Q}_1, \dots, \mathbf{Q}_N\}$ from $\mathcal{A}_{\text{train}}$ with a given Tucker-rank $\{\mu_1, \mu_2, \dots, \mu_N\}$.

6.2 Numerical results

In Algorithm 7, the proposed algorithms used in Step 2 include the FD-based algorithms (i.e., Algorithm 5 with FD, SpFD, GaussianFD, Gaussian-SpFD and Dct-SpFD), Tucker-ALS, T-HOSVD and ST-HOSVD, and the desired Tucker-ranks for COIL-20 and MNIST are specially set: $\{20, 20, 72, 20\}$ for COIL-20 and $\{9, 8, 1, 10\}$ for MNIST. For MNIST, the number for training samples per class is 2000, for COIL-20 with position, the number for training samples per class is 20, and for COIL-20 without position, the number for training samples per class is 40. To experiment the technique, we use the cross validation method and perform the training and the classification task 5 times. The data used for the training step are modified at each iteration to avoid overfitting bias. The related results are summarized in Table 7. For each class of MNIST and COIL-20 (w), detailed metrics (i.e., precision, recall and F1-score) obtained by Algorithm 5 with Gaussian-SpFD, Tucker-ALS, T-HOSVD and ST-HOSVD are presented in Appendix C. Note that CPU time in this section is to implement Step 2 in Algorithm 7.

Dataset	Algorithms	Precision	CPU time (s)
MNIST	Algorithm 5 with FD	0.8035	3.7959
	Algorithm 5 with SpFD	0.8029	0.3326
	Algorithm 5 with GaussianFD	0.8043	0.3734
	Algorithm 5 with Gaussian-SpFD	0.8031	0.2030
	Algorithm 5 with Dct-SpFD	0.7994	0.3465
	Tucker-ALS	0.8066	0.4179
	T-HOSVD	0.8005	0.9687
	ST-HOSVD	0.8029	0.3017
COIL-20 (w)	Algorithm 5 with FD	1.0000	4.9010
	Algorithm 5 with SpFD	1.0000	0.2981
	Algorithm 5 with GaussianFD	1.0000	0.3196
	Algorithm 5 with Gaussian-SpFD	1.0000	0.1990
	Algorithm 5 with Dct-SpFD	1.0000	0.2990
	Tucker-ALS	1.0000	0.3522
	T-HOSVD	1.0000	1.9219
	ST-HOSVD	1.0000	0.2983
COIL-20 (w/o)	Algorithm 5 with FD	0.6203	4.8387
	Algorithm 5 with SpFD	0.6175	0.2834
	Algorithm 5 with GaussianFD	0.6178	0.3209
	Algorithm 5 with Gaussian-SpFD	0.6194	0.1908
	Algorithm 5 with Dct-SpFD	0.6172	0.2912
	Tucker-ALS	0.6175	0.3531
	T-HOSVD	0.6169	1.8690
	ST-HOSVD	0.6169	0.2964

Table 7: The related results of the classification experiment on MNIST and COIL-20. For COIL-20, "without position (w/o)" indicates that images were classified only regarding objects, and "with position (w)" indicates that the images were classified regarding positions and objects.

The results on MNIST show a significant enhancement over traditional clustering methods, with global precision improving by nearly a factor of eight. This suggests that incorporating richer contextual information¡ªsuch as the actual digit representation¡ªinto the tensor allows the Tucker decomposition to uncover underlying patterns more effectively. For COIL-20, the Tucker decomposition reveals a versatile classification mechanism. Rather than limiting classification to a single dimension, the model enables categorization across multiple class dimensions simultaneously, offering a more nuanced and flexible approach to object recognition. For both MNIST and COIL-20, Algorithm 5 with Gaussian-SpFD is comparable to Tucker-ALS when using them in classification tasks.

7. Conclusions

In this paper, we discussed several randomized variants of the FD algorithm for low-rank matrix approximations and derived the FD-based algorithms for computing an approximate Tucker and TT decomposition in the framework of T-HOSVD, ST-HOSVD and TT-SVD. Firstly, we obtained a new randomized FD algorithm with rigorous theoretical analysis (see Theorem 10) by replacing the SpEmb matrices as the standard Gaussian matrices in SpFD. Secondly, in the framework of SpFD, two other randomized FD algorithms were obtained by replacing the SpEmb matrices as the product of the standard Gaussian and SpEmb matrices or the product of the SRHT and SpEmb matrices. Thirdly, with a given Tucker-rank, we presented two randomized variants of T-HOSVD and ST-HOSVD by applying all the variants of the FD algorithm to obtain all the factor matrices. We also discussed the FD-based randomized algorithms for computing an approximate TT decomposition with a given TT-rank. Finally, the efficiency and accuracy of these proposed algorithms were illustrated by numerical examples.

Acknowledgements

The authors would like to thank the Action Editor Professor Animashree Anandkumar and three anonymous reviewers for their careful and very detailed comments on our paper. This work is supported by the Natural Science Special Project Research Fund of Guizhou University (No. 2025-06), the Hong Kong Innovation and Technology Commission (InnoHK Project CIMDA), the Hong Kong Research Grants Council (Project 11204821), City University of Hong Kong (Projects 9610034 and 9610460) and the National Natural Science Foundation of China under the grant 12561095, 12271108 and U24A2001.

Appendix A. Proof for several theorems in Sections 3.2 and 3.3

In this section, we give a rigorous proof for Theorems 7, 10 and 13. Following Algorithm 3, one has $\mathbf{A}\mathbf{A}^\top = \widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top$, where $\widetilde{\mathbf{A}} = \mathbf{A}\mathbf{P}$. Hence, we rewrite (1) as

$$\|\mathbf{A} - \widetilde{\mathbf{A}}_\mu\|_F^2 \leq \|\mathbf{A} - \mathbf{A}_\mu\|_F^2 + 2\mu\|\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top - \mathbf{B}\mathbf{B}^\top\|_2.$$

We now introduce some notations. Let $\widetilde{\mathbf{A}} = [\widetilde{\mathbf{A}}_1, \widetilde{\mathbf{A}}_2, \dots, \widetilde{\mathbf{A}}_q]$, $\mathbf{C} = [\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_q]$, and $\mathbf{D} = [\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_q]$, where $\widetilde{\mathbf{A}}_i = \widetilde{\mathbf{A}}(:, iI_2/q+1, (i+1)I_2/q)$, $\mathbf{C}_i = \widetilde{\mathbf{A}}_i\mathbf{S}_i$ and $\mathbf{D}_i = \mathbf{C}_i\mathbf{G}_i = \widetilde{\mathbf{A}}_i\mathbf{S}_i\mathbf{G}_i$. Then we have

$$\|\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top - \mathbf{B}\mathbf{B}^\top\|_2 \leq \|\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top - \mathbf{C}\mathbf{C}^\top\|_2 + \|\mathbf{C}\mathbf{C}^\top - \mathbf{B}\mathbf{B}^\top\|_2. \tag{7}$$

A.1 Proof of Theorem 7

In order to obtain upper bound for $\|\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top - \mathbf{C}\mathbf{C}^\top\|_2$, we now introduce the following lemma.

Lemma 23 (Teng and Chu, 2018, lemma 1) *Suppose that* $\mathbf{V} \in \mathbb{R}^{I_2 \times K}$ *is an orthonormal matrix. For any* $\epsilon, \delta \in (0,1)$*, let*

$$qL' \geq \frac{12(K^2+K)}{\epsilon^2\delta}, \quad L' = \left\lceil K + \frac{3K}{\epsilon\delta} \right\rceil$$

and $\mathbf{S} \in \mathbb{R}^{I_2 \times qL'}$ *be defined as* $\mathbf{S} = \mathrm{bdiag}(\mathbf{S}_1, \ldots, \mathbf{S}_q)$*, where* $\mathbf{S}_i \in \mathbb{R}^{I_2/q \times L'}$ *is a SpEmb matrix. Let* $\mathbf{P} \in \mathbb{R}^{I_2 \times I_2}$ *be a random permutation matrix. Then with probability at least* $1-\delta$*, we have*

$$\|(\mathbf{V}^\top\mathbf{P})\mathbf{S}\mathbf{S}^\top(\mathbf{V}^\top\mathbf{P})^\top - \mathbf{I}_K\|_2 \leq \epsilon.$$

The block diagonal matrix $\mathrm{bdiag}(\mathbf{S}_1, \ldots, \mathbf{S}_q)$ in Lemma 23 is defined by

$$\mathbf{S} = \mathrm{bdiag}(\mathbf{S}_1, \ldots, \mathbf{S}_q) = \begin{pmatrix} \mathbf{S}_1 & & & \\ & \mathbf{S}_2 & & \\ & & \ddots & \\ & & & \mathbf{S}_q \end{pmatrix}. \tag{8}$$

By the definition of $\mathbf{S}$, we have $\mathbf{C} = \mathbf{APS}$. Let $\mathbf{A} = \mathbf{U\Sigma V}^\top$ with $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_{I_1})$, where $\mathbf{U} \in \mathbb{R}^{I_1 \times I_1}$ is orthogonal, $\mathbf{V} \in \mathbb{R}^{I_2 \times I_1}$ is orthonormal, and all the σ_i are the singular values of $\mathbf{A}$. From Lemma 23, for any $\epsilon, \delta \in (0,1)$, when

$$qL' \geq \frac{12(I_1^2+I_1)}{\epsilon^2\delta}, \quad L' = \left\lceil I_1 + \frac{3I_1}{\epsilon\delta} \right\rceil, \tag{9}$$

with probability at least $1-\delta$, we have

$$\begin{aligned} \|\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top - \mathbf{C}\mathbf{C}^\top\|_2 &= \|\mathbf{U\Sigma\Sigma U}^\top - \mathbf{U\Sigma}(\mathbf{V}^\top\mathbf{PS})(\mathbf{S}^\top\mathbf{P}^\top\mathbf{V})\mathbf{\Sigma U}^\top\|_2 \\ &= \|\mathbf{U\Sigma}(\mathbf{I}_{I_2} - (\mathbf{V}^\top\mathbf{PS})(\mathbf{S}^\top\mathbf{P}^\top\mathbf{V}))\mathbf{\Sigma U}^\top\|_2 \\ &\leq \|\mathbf{I}_{I_2} - (\mathbf{V}^\top\mathbf{PS})(\mathbf{S}^\top\mathbf{P}^\top\mathbf{V})\|_2\|\mathbf{A}\|_F^2 \\ &\leq \epsilon\|\mathbf{A}\|_F^2. \end{aligned} \tag{10}$$

We now list the following property for a standard Gaussian matrix, regarding to the error bound of extreme singular values.

Lemma 24 (Vershynin, 2010, Corollary 5.35) *Let* $\mathbf{G} \in \mathbb{R}^{I_1 \times I_2}$ *be a matrix whose entries are independent standard normal random variables. Then for every* $\epsilon > 0$*, with probability at least* $1 - 2\exp(-\epsilon^2/2)$*, one has*

$$\sqrt{I_1} - \sqrt{I_2} - \epsilon \leq \sigma_{\min}(\mathbf{G}) \leq \sigma_{\max}(\mathbf{G}) \leq \sqrt{I_1} + \sqrt{I_2} + \epsilon.$$

Remark 25 *Since the matrix* $\mathbf{G}_i$ *is a standard Gaussian matrix, the upper bound of* $\|\mathbf{G}_i\|_2$ *can be also obtained from the following two results from* (Coakley et al. (2011); Litvak et al. (2005))*:*

(a) *Let* $\mathbf{G} \in \mathbb{R}^{I \times L}$ *be a standard Gaussian matrix with* $L < I$*. For* $\gamma > 1$*, if*

$$1 - \frac{1}{4(\gamma^2-1)\sqrt{\pi I\gamma^2}}\left(\frac{2\gamma^2}{\exp(\gamma^2-1)}\right)^2 \tag{11}$$

is nonnegative, then, the largest singular value of $\mathbf{G}$ *is at most* $\sqrt{2I}\gamma$ *with probability not less than the amount in* (11).

(b) *Suppose that* $\mathbf{G} \in \mathbb{R}^{L\times L}$ *is a sub-Gaussian matrix with* $L \leq I$*,* $s \geq 1$ *and* $a_2 > 0$*. Then* $\mathbf{P}(\|\mathbf{G}\|_2 > a_1\sqrt{I}) \leq \exp(-a_2 I)$*, where* $a_1 = 6s\sqrt{a_2+4}$*.*

We now consider the term $\|\mathbf{C}\mathbf{C}^\top - \mathbf{B}\mathbf{B}^\top\|_2$ in the right-hand side of (7). It is easy to see that

$$\|\mathbf{C}\mathbf{C}^\top - \mathbf{B}\mathbf{B}^\top\|_2 \leq \|\mathbf{C}\mathbf{C}^\top - \mathbf{D}\mathbf{D}^\top\|_2 + \|\mathbf{D}\mathbf{D}^\top - \mathbf{B}\mathbf{B}^\top\|_2.$$

Note that the matrix $\mathbf{B}$ is obtained by applying Algorithm 1 to the matrix $\mathbf{D}$. Then from (Ghashami et al. (2016b); Liberty (2013)), we have

$$\|\mathbf{D}\mathbf{D}^\top - \mathbf{B}\mathbf{B}^\top\|_2 \leq \frac{2}{L}\|\mathbf{D}\|_F^2 = \frac{2}{L}\sum_{i=1}^q \|\mathbf{D}_i\|_F^2 \leq \frac{2}{L}\sum_{i=1}^q \|\widetilde{\mathbf{A}}_i\|_F^2\|\mathbf{S}_i\|_2^2\|\mathbf{G}_i\|_2^2.$$

It follows from Lemma 2.3 in (Aizenbud et al. (2016)) that for the SpEmb matrix $\mathbf{S}_i \in \mathbb{R}^{I_2/q\times L'}$, we have $\|\mathbf{S}_i\|_F = \sqrt{I_2/q}$, which implies that $\|\mathbf{S}_i\|_2 \leq \|\mathbf{S}_i\|_F = \sqrt{I_2/q}$. From Lemma 24, for any $\epsilon > 0$, with probability at least $1 - 2\exp(-\epsilon^2/2)$, one has $\|\mathbf{G}_i\|_2 \leq \sqrt{L} + \sqrt{L'} + \epsilon$. Then, with probability at least $1 - 2\exp(-\epsilon^2/2)$, one has

$$\begin{aligned}\|\mathbf{D}\mathbf{D}^\top - \mathbf{B}\mathbf{B}^\top\|_2 &\leq \frac{2}{L}\sum_{i=1}^q \|\mathbf{D}_i\|_F^2 = \frac{2I_2(\sqrt{L}+\sqrt{L'}+\epsilon)^2}{qL}\sum_{i=1}^q \|\widetilde{\mathbf{A}}_i\|_F^2 \\ &= \frac{2I_2(\sqrt{L}+\sqrt{L'}+\epsilon)^2}{qL}\|\mathbf{A}\|_F^2.\end{aligned}$$

Finally, we consider the upper bound for $\|\mathbf{C}\mathbf{C}^\top - \mathbf{D}\mathbf{D}^\top\|_2$. By the triangular inequality of the spectral norm, we have

$$\begin{aligned}\|\mathbf{C}\mathbf{C}^\top - \mathbf{D}\mathbf{D}^\top\|_2 &\leq \sum_{i=1}^q \|\mathbf{C}_i\mathbf{C}_i^\top - \mathbf{D}_i\mathbf{D}_i^\top\|_2 \\ &= \sum_{i=1}^q \|\mathbf{C}_i\mathbf{C}_i^\top - \mathbf{C}_i\mathbf{G}_i(\mathbf{C}_i\mathbf{G}_i)^\top\|_2.\end{aligned}$$

Here the equality holds for the definition of $\mathbf{C}_i$ and $\mathbf{D}_i$.

As shown in (Litvak et al. (2005)), a standard Gaussian matrix is also a sub-Gaussian matrix. Then, from Theorem 1 in (Cohen et al. (2016)), for any $\epsilon, \delta, \tau \in (0, 1/2)$, when $L = \Omega((\tau + \log(1/\delta))/\epsilon^2)$, with probability at least $1 - \delta$, one has

$$\|\mathbf{C}_i\mathbf{C}_i^\top - \mathbf{C}_i\mathbf{G}_i(\mathbf{C}_i\mathbf{G}_i)^\top\|_2 \leq \epsilon\left(\|\mathbf{C}_i\|_2^2 + \frac{\|\mathbf{C}_i\|_F^2}{\tau}\right),$$

which implies that

$$\begin{aligned}\|\mathbf{C}\mathbf{C}^\top - \mathbf{D}\mathbf{D}^\top\|_2 &\leq \epsilon\left(1 + \frac{1}{\tau}\right)\sum_{i=1}^q \|\mathbf{C}_i\|_F^2 \\ &\leq \epsilon\left(I_2 + \frac{I_2}{\tau}\right)\sum_{i=1}^q \|\widetilde{\mathbf{A}}_i\|_F^2 \\ &= \epsilon\left(I_2 + \frac{I_2}{\tau}\right)\|\mathbf{A}\|_F^2.\end{aligned}$$

Here the first inequality holds for the fact that $\|\mathbf{C}_i\|_2 \le \|\mathbf{C}_i\|_F$, the second inequality is derived from Lemma 2.3 in (Aizenbud et al. (2016)) and the equality is from $\widetilde{\mathbf{A}} = \mathbf{AP}$ and the fact that $\mathbf{P}$ is an orthogonal matrix.

Therefore, we prove Theorem 7 completely.

A.2 Proof of Theorem 10

As shown in Algorithm 2 with standard Gaussian matrices, the matrices $\mathbf{C}_i$ in (7) is replaced as $\mathbf{C}_i = \widetilde{\mathbf{A}}_i\mathbf{G}_i$, and the matrix $\mathbf{B}$ is obtained by applying Algorithm 2 to $\mathbf{C}$. Hence, according to (Ghashami et al. (2016b); Liberty (2013)) and Lemma 24, with probability at least $1 - 2\exp(-\epsilon^2/2)$, one has

$$\begin{aligned} \|\mathbf{CC}^\top - \mathbf{BB}^\top\|_2 &\le \frac{2}{L}\|\mathbf{C}\|_F^2 = \frac{2}{L}\sum_{i=1}^{q}\|\mathbf{C}_i\|_F^2 \\ &\le \frac{2}{L}\sum_{i=1}^{q}\|\mathbf{A}_i\|_F^2\|\mathbf{G}_i\|_2^2 \\ &\le \frac{2(\sqrt{I_1}+\sqrt{I_2}+\epsilon)^2}{L}\|\mathbf{A}\|_F^2. \end{aligned} \tag{12}$$

Meanwhile, we also have

$$\begin{aligned} \|\mathbf{AA}^\top - \mathbf{CC}^\top\|_2 &\le \sum_{i=1}^{q}\|\mathbf{A}_i\mathbf{A}_i^\top - \mathbf{C}_i\mathbf{C}_i^\top\|_2 \\ &\le \sum_{i=1}^{q}\epsilon\left(\|\mathbf{A}_i\|_2^2 + \frac{\|\mathbf{A}_i\|_F^2}{\tau}\right) \\ &\le \sum_{i=1}^{q}\epsilon\left(1+\frac{1}{\tau}\right)\|\mathbf{A}_i\|_F^2 \\ &= \epsilon\left(1+\frac{1}{\tau}\right)\|\mathbf{A}\|_F^2. \end{aligned} \tag{13}$$

The first inequality holds for the triangular inequality of the spectral norm, the second inequality holds with probability at least $1-\delta$, where $\epsilon, \delta, \tau \in (0, 1/2)$, and the third inequality follows from $\|\mathbf{A}_i\|_2 \le \|\mathbf{A}_i\|_F$.

From (1), we have

$$\|\mathbf{A} - \widetilde{\mathbf{A}}_\mu\|_F^2 \le \|\mathbf{A} - \mathbf{A}_\mu\|_F^2 + 2\mu(\|\mathbf{AA}^\top - \mathbf{CC}^\top\|_2 + \|\mathbf{CC}^\top - \mathbf{BB}^\top\|_2). \tag{14}$$

Hence, Theorem 10 is completely proved by substituting (12) and (13) into (14).

A.3 Proof of Theorem 13

Note that we have

$$\|\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top - \mathbf{BB}^\top\|_2 \le \|\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top - \mathbf{CC}^\top\|_2 + \|\mathbf{CC}^\top - \mathbf{BB}^\top\|_2.$$

The upper bound for the first term $\|\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top - \mathbf{C}\mathbf{C}^\top\|_2$ is given in (10). We now consider the second term $\|\mathbf{C}\mathbf{C}^\top - \mathbf{B}\mathbf{B}^\top\|_2$.

The notations used in this section are the same as in Section 3.2. When all the matrices $\mathbf{G}_i$ are SRHT matrices, the matrix $\mathbf{B}$ can be obtained by applying the FFD (faster frequent directions) algorithm in (Chen et al. (2017)) to the matrix $\mathbf{C}$. Hence, the upper bound of $\|\mathbf{C}\mathbf{C}^\top - \mathbf{B}\mathbf{B}^\top\|_2$ is given in the following theorem.

Theorem 26 *For a given matrix $\mathbf{A} \in \mathbb{R}^{I_1\times I_2}$, let $\widetilde{\mathbf{A}} = \mathbf{A}\mathbf{P}$, where $\mathbf{P} \in \mathbb{R}^{I_2\times I_2}$ is a random permutation matrix. Let $\widetilde{\mathbf{A}} = [\widetilde{\mathbf{A}}_1, \widetilde{\mathbf{A}}_2, \dots, \widetilde{\mathbf{A}}_q]$, and $\mathbf{C} = [\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_q]$, where $\widetilde{\mathbf{A}}_i = \widetilde{\mathbf{A}}(:, iI_2/q+1, (i+1)I_2/q)$, and $\mathbf{C}_i = \widetilde{\mathbf{A}}_i\mathbf{S}_i \in \mathbb{R}^{I_1\times L'}$. Assume that assume that I_2 is a multiplier of q, L' is a power of two with $L \leq \min\{I_1, L'\}$ and $L' < I_2/q$, and $0 < \beta < 1$ satisfies $0 < I_2\beta/q < 1$.*

Let the matrix $\mathbf{B} \in \mathbb{R}^{I_1\times L}$ be constructed by Algorithm 3 with SpEmb+SRHT. Then with probability at least

$$1 - \frac{I_2}{q}\beta - \left(2\frac{I_2}{L'} + 1\right)\delta - \frac{2I_1}{\exp(\min\{L', I_1\})},$$

we have

$$\|\mathbf{C}\mathbf{C}^\top - \mathbf{B}\mathbf{B}^\top\|_2 \leq \widetilde{O}\left(\frac{1}{L} + \Gamma\left(L, \frac{I_2}{L'}, \min\{I_1, L'\}\right)\right)\frac{I_2}{q}\|\mathbf{A}\|_F^2,$$

where

$$\Gamma\left(L, \frac{I_2}{L'}, \min\{I_1, L'\}\right) = \sqrt{\frac{\min\{I_1, L'\}}{L(I_2/L')^2}} + \sqrt{\frac{1 + \sqrt{\min\{I_2/L'\}/L}}{I_1/L'}},$$

and $\widetilde{O}(\cdot)$ hides logarithmic factors on β, δ, $\min\{I_1, L'\}$, I_2 and L'.

Proof Theorem 26 is directly deduced from Theorem 1 in Chen et al. (2017). ∎

Hence, when $\widetilde{\mathbf{A}}_\mu$ is constructed by Algorithm 3 with SpEmb+SRHT, Theorem 13 is obtained by combining $\mathbf{A}\mathbf{A}^\top = \widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top$, (1), (10) and Theorem 26.

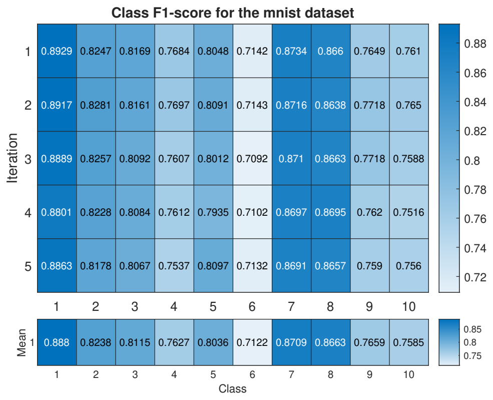
Appendix B. More considerations for $\|\mathbf{C}\mathbf{C}^\top - \mathbf{D}\mathbf{D}^\top\|_2$

Note that for each i, one has $\mathbf{D}_i = \mathbf{C}_i\mathbf{G}_i$, which implies that $\mathbf{D} = \mathbf{C}\mathbf{G}$, where $\mathbf{G} = \mathrm{bdiag}(\mathbf{G}_1, \dots, \mathbf{G}_q)$. Then, one has $\|\mathbf{C}\mathbf{C}^\top - \mathbf{D}\mathbf{D}^\top\|_2 = \|\mathbf{C}\mathbf{C}^\top - (\mathbf{C}\mathbf{G})(\mathbf{C}\mathbf{G})^\top\|_2$. Note that the matrix $\mathbf{G}$ is called a block diagonal sketching matrix. Srinivasa et al. (2020) considered approximate matrix multiplication and ridge regression by the block diagonal sketching matrix $\mathbf{G}$. Hence, the upper bound of $\|\mathbf{C}\mathbf{C}^\top - \mathbf{D}\mathbf{D}^\top\|_2$ can also be obtained by using Theorem 1 in (Srinivasa et al. (2020)).

By the definitions of $\mathbf{C}$ and $\mathbf{D}$, we have $\mathbf{C} = \widetilde{\mathbf{A}}\mathbf{S}$ and $\mathbf{D} = \mathbf{C}\mathbf{G} = \widetilde{\mathbf{A}}\mathbf{S}\mathbf{G}$. As we know, the matrix $\mathbf{S}$ in (8) is a block diagonal sketching matrix, which implies that $\mathbf{S}\mathbf{G}$ is also a block diagonal sketching matrix. Hence, an issue is to deduce the bounds for $\|\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top - \mathbf{C}\mathbf{C}^\top\|_2$ and $\|\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\top - \mathbf{D}\mathbf{D}^\top\|_2$ in the framework of thw work in (Srinivasa et al. (2020)).

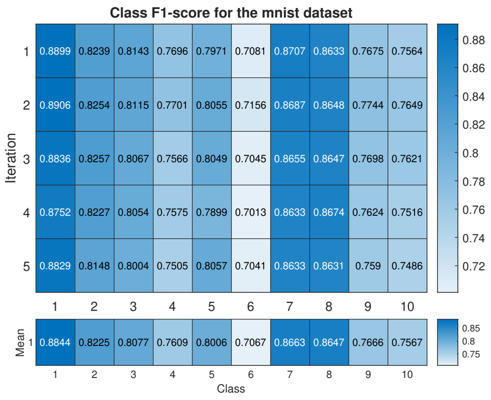
Appendix C. More detailed results in Section 6.2

For each class of MNIST and COIL-20 (w), the values of precision, recall and F1-score obtained by Algorithm 5 with Gaussian-SpFD, Tucker-ALS, T-HOSVD and ST-HOSVD are shown in Figure 7 to Figure 12, which illustrate that in terms of precision, recall and F1-score for the classification task, Algorithm 5 with Gaussian-SpFD is comparable to Tucker-ALS, T-HOSVD and ST-HOSVD.



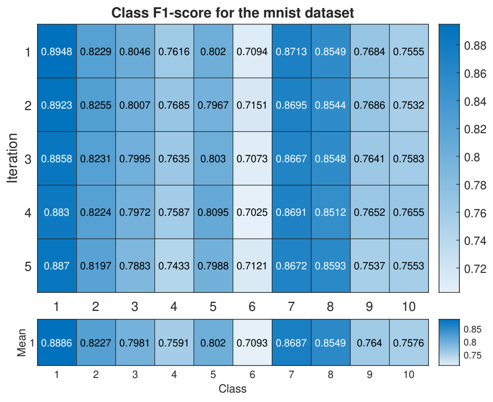
Iteration	1	2	3	4	5	6	7	8	9	10
1	0.8929	0.8247	0.8169	0.7684	0.8048	0.7142	0.8734	0.866	0.7649	0.761
2	0.8917	0.8281	0.8161	0.7697	0.8091	0.7143	0.8716	0.8638	0.7718	0.765
3	0.8889	0.8257	0.8092	0.7607	0.8012	0.7092	0.871	0.8663	0.7718	0.7588
4	0.8801	0.8228	0.8084	0.7612	0.7935	0.7102	0.8697	0.8695	0.762	0.7516
5	0.8863	0.8178	0.8067	0.7537	0.8097	0.7132	0.8691	0.8657	0.759	0.756
Mean 1	0.888	0.8238	0.8115	0.7627	0.8036	0.7122	0.8709	0.8663	0.7659	0.7585

(a) Tucker-ALS



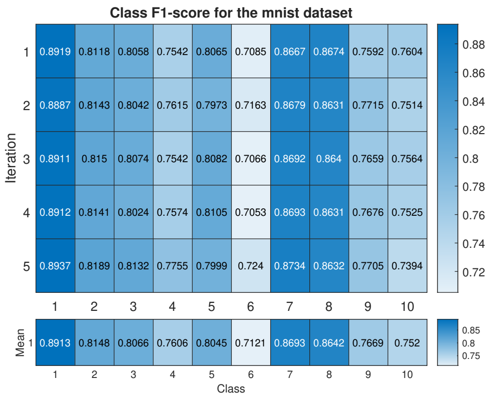
Iteration	1	2	3	4	5	6	7	8	9	10
1	0.8899	0.8239	0.8143	0.7696	0.7971	0.7081	0.8707	0.8633	0.7675	0.7564
2	0.8906	0.8254	0.8115	0.7701	0.8055	0.7156	0.8687	0.8648	0.7744	0.7649
3	0.8836	0.8257	0.8067	0.7566	0.8049	0.7045	0.8655	0.8647	0.7698	0.7621
4	0.8752	0.8227	0.8054	0.7575	0.7899	0.7013	0.8633	0.8674	0.7624	0.7516
5	0.8829	0.8148	0.8004	0.7505	0.8057	0.7041	0.8633	0.8631	0.759	0.7486
Mean 1	0.8844	0.8225	0.8077	0.7609	0.8006	0.7067	0.8663	0.8647	0.7666	0.7567

(b) T-HOSVD



Iteration	1	2	3	4	5	6	7	8	9	10
1	0.8948	0.8229	0.8046	0.7616	0.802	0.7094	0.8713	0.8549	0.7684	0.7555
2	0.8923	0.8255	0.8007	0.7685	0.7967	0.7151	0.8695	0.8544	0.7686	0.7532
3	0.8858	0.8231	0.7995	0.7635	0.803	0.7073	0.8667	0.8548	0.7641	0.7583
4	0.883	0.8224	0.7972	0.7587	0.8095	0.7025	0.8691	0.8512	0.7652	0.7655
5	0.887	0.8197	0.7883	0.7433	0.7988	0.7121	0.8672	0.8593	0.7537	0.7553
Mean 1	0.8886	0.8227	0.7981	0.7591	0.802	0.7093	0.8687	0.8549	0.764	0.7576

(c) ST-HOSVD

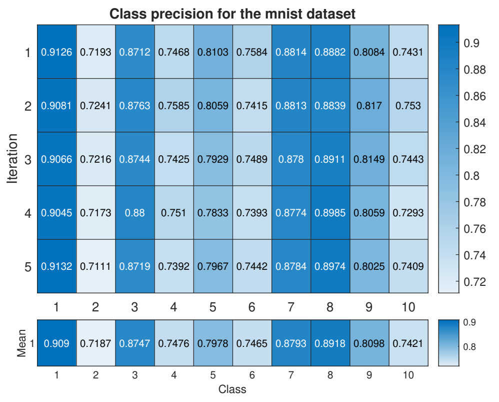


Iteration	1	2	3	4	5	6	7	8	9	10
1	0.8919	0.8118	0.8058	0.7542	0.8065	0.7085	0.8667	0.8674	0.7592	0.7604
2	0.8887	0.8143	0.8042	0.7615	0.7973	0.7163	0.8679	0.8631	0.7715	0.7514
3	0.8911	0.815	0.8074	0.7542	0.8082	0.7066	0.8692	0.864	0.7659	0.7564
4	0.8912	0.8141	0.8024	0.7574	0.8105	0.7053	0.8693	0.8631	0.7676	0.7525
5	0.8937	0.8189	0.8132	0.7755	0.7999	0.724	0.8734	0.8632	0.7705	0.7394
Mean 1	0.8913	0.8148	0.8066	0.7606	0.8045	0.7121	0.8693	0.8642	0.7669	0.752

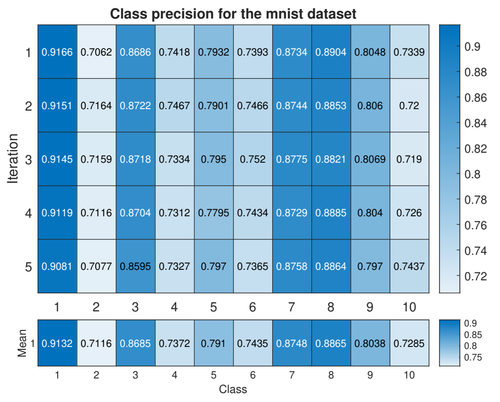
(d) Gaussian-SpFD

Figure 7: F1-score for each class of MNIST for the classification technique.

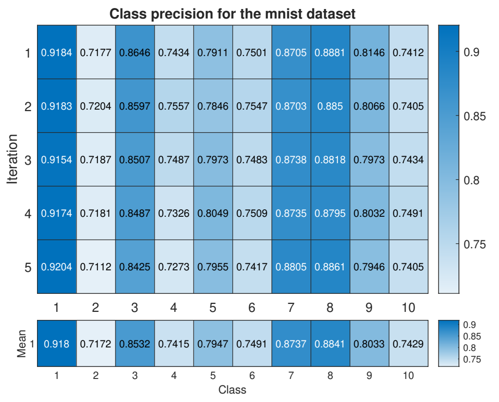
Remark 27 *Note that for MNIST, Figures 7, 8 and 9 illustrate that the values of precision, recall and F1-score are quite low. Hence, in order to improve the precision of Algorithm 7, we will combine this algorithm with some more effective classification strategies in future work. Another future work is to consider more downstream ML tasks such as classification, clustering, or model compression based on the proposed algorithm.*



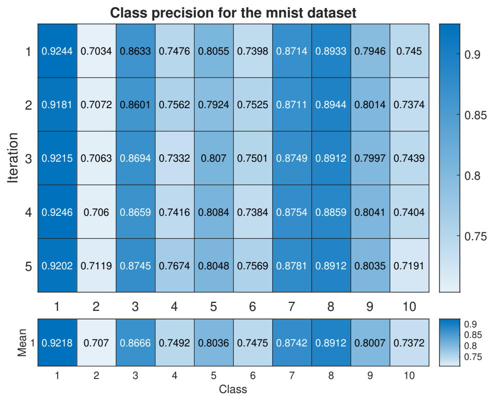
(a) Tucker-ALS



(b) T-HOSVD



(c) ST-HOSVD



(d) Gaussian-SpFD

Figure 8: Precision for each class of MNIST for the classification technique.

References

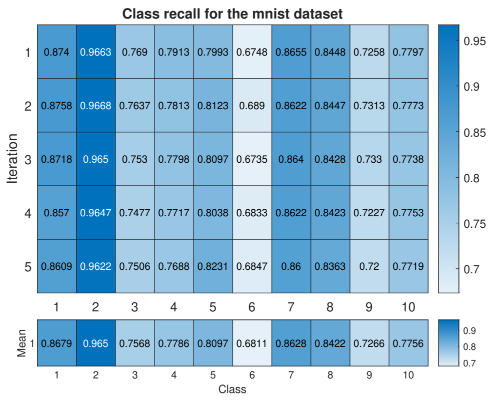
Salman Ahmadi-Asl, Stanislav Abukhovich, Maame G Asante-Mensah, Andrzej Cichocki, Anh Huy Phan, Tohishisa Tanaka, and Ivan Oseledets. Randomized algorithms for computation of Tucker decomposition and higher order SVD (HOSVD). *IEEE Access*, 9: 28684–28706, 2021.

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

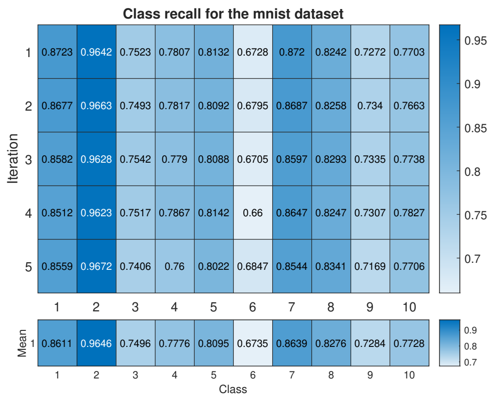
Nir Ailon and Edo Liberty. Fast dimension reduction using Rademacher series on dual BCH codes. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1–9, 2008.

Yariv Aizenbud and Amir Averbuch. Matrix decompositions using sub-gaussian random matrices. *Information and Inference: A Journal of the IMA*, 8(3):445–469, 2019.

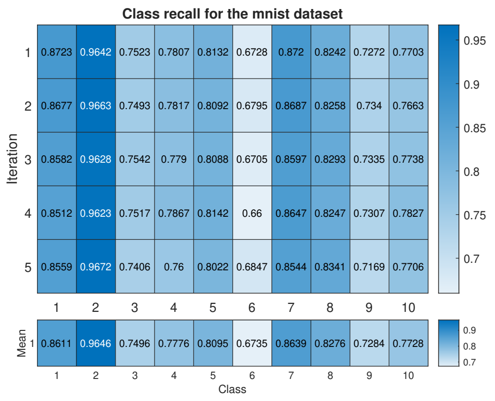
Yariv Aizenbud, Gil Shabat, and Amir Averbuch. Randomized LU decomposition using sparse projections. *Computers and Mathematics with Applications*, 72:2525–2534, 2016.



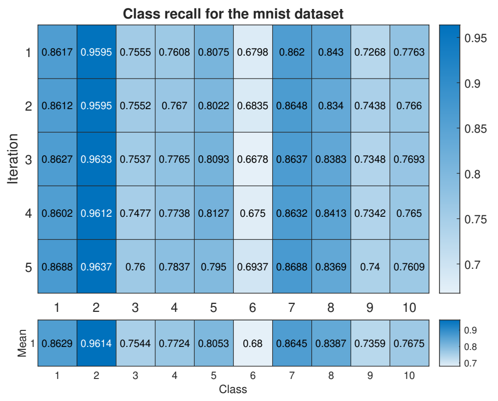
(a) Tucker-ALS



(b) T-HOSVD



(c) ST-HOSVD



(d) Gaussian-SpFD

Figure 9: Recall for each class of MNIST for the classification technique.

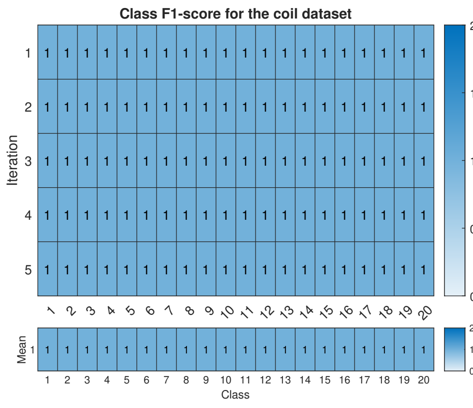
Nick Alger, Peng Chen, and Omar Ghattas. Tensor train construction from tensor actions, with application to compression of large high order derivative tensors. *SIAM Journal on Scientific Computing*, 42(5):A3516–A3539, 2020.

Brett W. Bader and Tamara G. Kolda. Matlab tensor toolbox version 3.2.1. Available online, April 5, 2021. `https://www.tensortoolbox.org`.

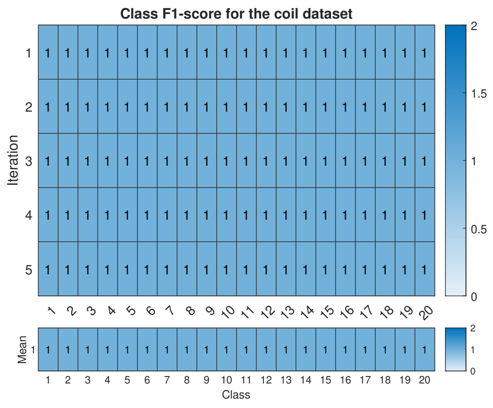
Christos Boutsidis and Alex Gittens. Improved matrix algorithms via the subsampled randomized Hadamard transform. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1301–1340, 2013.

Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near-optimal column-based matrix reconstruction. *SIAM Journal on Computing*, 43(2):687–717, 2014.

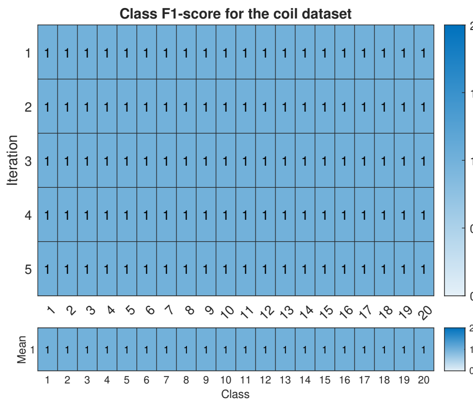
Cesar F. Caiafa and Andrzej Cichocki. Generalizing the column-row matrix decomposition to multi-way arrays. *Linear Algebra and its Applications*, 433(3):557–573, 2010.



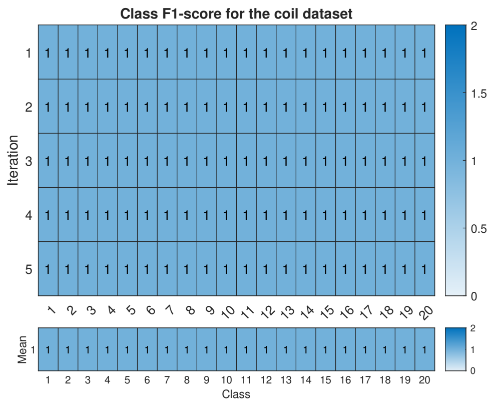
(a) Tucker-ALS



(b) T-HOSVD



(c) ST-HOSVD



(d) Gaussian-SpFD

Figure 10: F1-score for each class of COIL-20 with position for the classification technique.

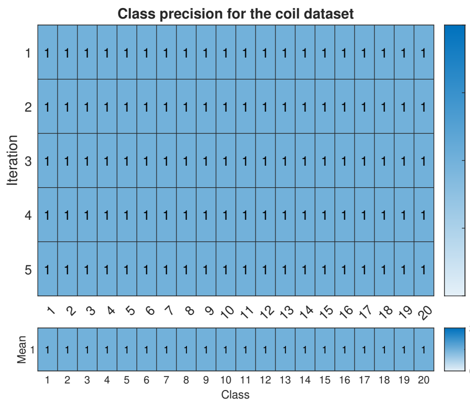
Maolin Che and Yimin Wei. Randomized algorithms for the approximations of Tucker and the tensor train decompositions. *Advances in Computational Mathematics*, 45(1): 395–428, 2019.

Maolin Che, Yimin Wei, and Hong Yan. The computation of low multilinear rank approximations of tensors via power scheme and random projection. *SIAM Journal on Matrix Analysis and Applications*, 41(2):605–636, 2020.

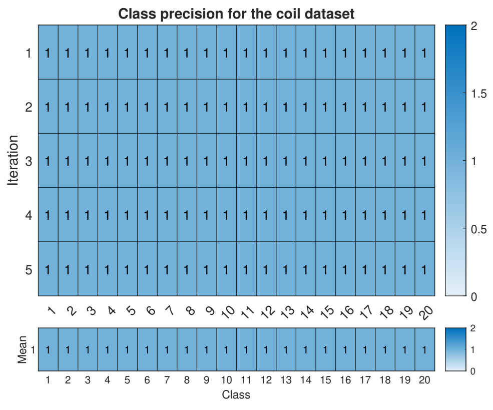
Maolin Che, Yimin Wei, and Hong Yan. An efficient randomized algorithm for computing the approximate Tucker decomposition. *Journal of Scientific Computing*, 88(2):32, 2021a.

Maolin Che, Yimin Wei, and Hong Yan. Randomized algorithms for the low multilinear rank approximations of tensors. *Journal of Computational and Applied Mathematics*, 390: 113380, 2021b.

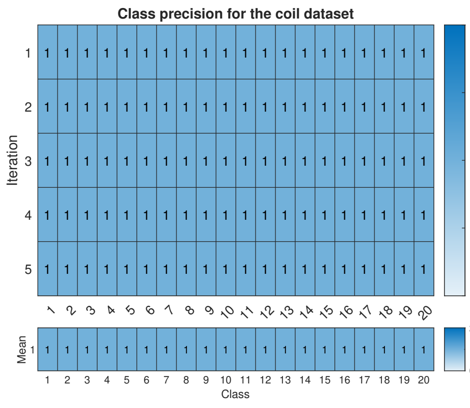
Maolin Che, Yimin Wei, and Hong Yan. Efficient algorithms for tucker decomposition via approximate matrix multiplication. *Advances in Computational Mathematics*, 51(3):20, 2025a.



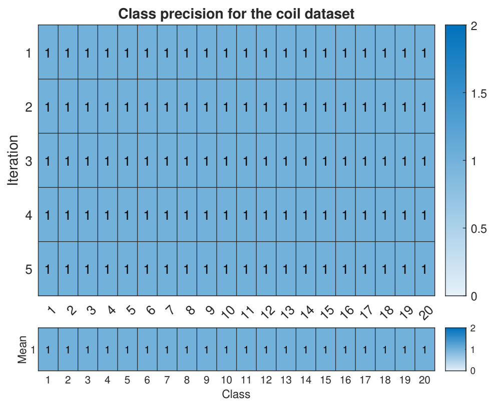
(a) Tucker-ALS



(b) T-HOSVD



(c) ST-HOSVD



(d) Gaussian-SpFD

Figure 11: Precision for each class of COIL-20 with position for the classification technique.

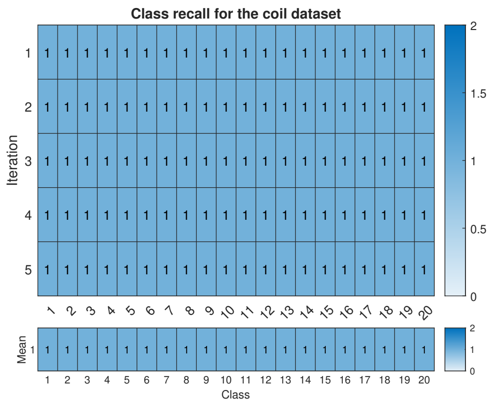
Maolin Che, Yimin Wei, and Hong Yan. Efficient randomized algorithms for fixed precision problem of approximate Tucker decomposition. *SIAM Journal on Matrix Analysis and Applications*, 46(1):256–297, 2025b.

Maolin Che, Yimin Wei, and Hong Yan. Efficient randomized algorithms for computing an approximation of the tensor train decomposition. *Journal of Scientific Computing*, 107 (2):40, 2026.

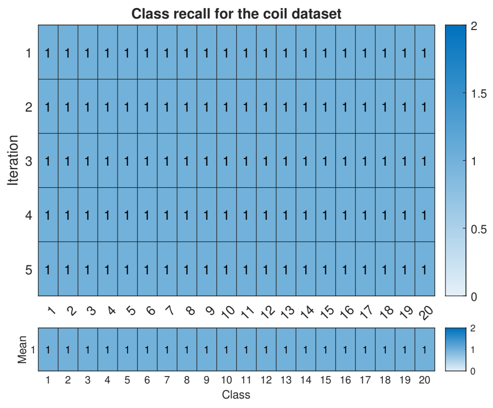
Cong Chen, Kim Batselier, Wenjian Yu, and Ngai Wong. Kernelized support tensor train machines. *Pattern Recognition*, 122, Article no. 108337, 2022.

Xixian Chen, Irwin King, Michael R Lyu, et al. FROSH: Faster online sketching hashing. In *UAI*, pages 1–10, 2017.

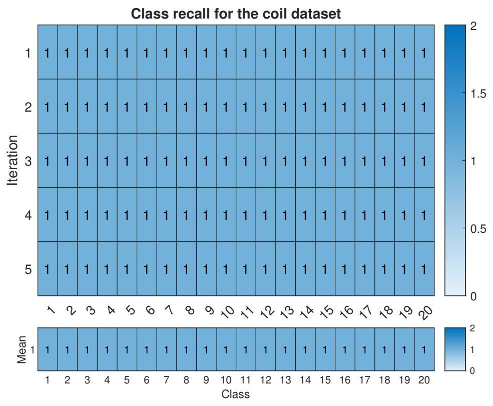
Andrzej Cichocki, Namgil Lee, Ivan V Oseledets, Anh Huy Phan, Qibin Zhao, and Danilo P Mandic. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends in Machine Learning*, 9(4-5): 249–429, 2016.



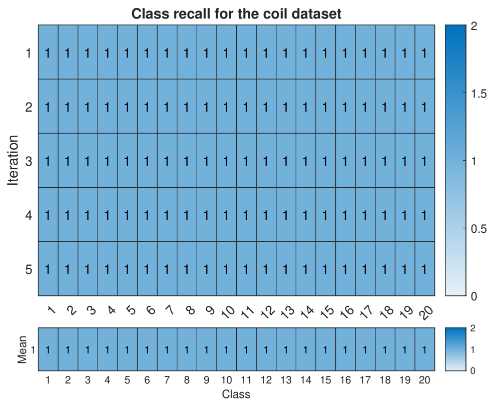
(a) Tucker-ALS



(b) T-HOSVD



(c) ST-HOSVD



(d) Gaussian-SpFD

Figure 12: Recall for each class of COIL-20 with position for the classification technique.

Andrzej Cichocki, Namgil Lee, Ivan V Oseledets, Anh Huy Phan, Qibin Zhao, and Danilo P Mandic. 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.

Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):1–45, 2017.

E. S. Coakley, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for orthogonal projection. *SIAM Journal on Scientific Computing*, 33(2):849–868, 2011.

Michael B Cohen, Jelani Nelson, and David P Woodruff. Optimal approximate matrix product in terms of stable rank. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, pages 11:1–11:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016.

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, 2000a.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-$(r_1, r_2, \cdots, r_n)$ approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000b.

Li Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Amey Desai, Mina Ghashami, and Jeff M Phillips. Improved practical matrix sketching with guarantees. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1678–1690, 2016.

Amit Deshpande, Luis Rademacher, Santosh S Vempala, and Grant Wang. Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2(1): 225–247, 2006.

S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov. Computation of extreme eigenvalues in higher dimensions using block tensor train format. *Computer Physics Communications*, 185(4):1207–1216, 2014.

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(2-3):553–571, 2007.

Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1): 132–157, 2006.

Virginie Ehrlacher, Laura Grigori, Damiano Lombardi, and Hao Song. Adaptive hierarchical subtensor partitioning for tensor compression. *SIAM Journal on Scientific Computing*, 43(1):A139–A163, 2021.

Athinodoros S Georghiades, Peter N Belhumeur, and David Kriegman. From few to many: illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

Mina Ghashami and Jeff M Phillips. Relative errors for deterministic low-rank matrix approximations. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 707–717. SIAM, 2014.

Mina Ghashami, Edo Liberty, and Jeff M Phillips. Efficient frequent directions algorithm for sparse matrices. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 845–854, 2016a.

Mina Ghashami, Edo Liberty, Jeff M Phillips, and David P Woodruff. Frequent directions: Simple and deterministic matrix sketching. *SIAM Journal on Computing*, 45(5):1762–1792, 2016b.

Annabelle Gillet, Éric Leclercq, and Lucile Sautot. *A Guide to the Tucker Tensor Decomposition for Data Mining: Exploratory Analysis, Clustering and Classification*, pages 56–88. Springer Berlin Heidelberg, Berlin, Heidelberg, 2023. ISBN 978-3-662-68014-8. doi: 10.1007/978-3-662-68014-8_3.

G. Golub and C. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, fourth edition, 2013. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

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.

Behnam Hashemi and Yuji Nakatsukasa. RTSMS: Randomized Tucker with single-mode sketching. *arXiv preprint arXiv:2311.14873*, 2023.

M. E. Hochstenbach and L. Reichel. Subspace-restricted singular value decompositions for linear discrete ill-posed problems. *Journal of Computational and Applied Mathematics*, 235(4):1053–1064, 2010.

John T Holodnak and Ilse CF Ipsen. Randomized approximation of the Gram matrix: exact computation and probabilistic bounds. *SIAM Journal on Matrix Analysis and Applications*, 36(1):110–137, 2015.

Zengfeng Huang. Near optimal frequent directions for sketching dense and sparse matrices. *The Journal of Machine Learning Research*, 20(1):2018–2040, 2019.

Benjamin Huber, Reinhold Schneider, and Sebastian Wolf. A randomized tensor train singular value decomposition. In *Compressed Sensing and its Applications: Second International MATHEON Conference 2015*, pages 261–290. Springer, 2017.

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

Kirandeep Kour, Sergey Dolgov, Martin Stoll, and Peter Benner. Efficient structure-preserving support tensor train machine. *Journal of Machine Learning Research*, 24 (4):1–22, 2023.

D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by Riemannian optimization. *BIT Numerical Mathematics*, 54:447–468, 2014a.

Daniel Kressner, Michael Steinlechner, and André Uschmajew. Low-rank tensor methods with subspace correction for symmetric eigenvalue problems. *SIAM Journal on Scientific Computing*, 36(5):A2346–A2368, 2014b.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Toronto, ON, Canada*, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lingjie Li, Wenjian Yu, and Kim Batselier. Faster tensor train decomposition for sparse data. *Journal of Computational and Applied Mathematics*, 405, Article no. 113972, 2022.

Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588, 2013.

Alexander E Litvak, Alain Pajor, Mark Rudelson, and Nicole Tomczak-Jaegermann. Smallest singular value of random matrices and geometry of random polytopes. *Advances in Mathematics*, 195(2):491–523, 2005.

Linjian Ma and Edgar Solomonik. Fast and accurate randomized algorithms for low–rank tensor decompositions. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24299–24312. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/cbef46321026d8404bc3216d4774c8a9-Paper.pdf`.

Osman Asif Malik and Stephen Becker. Low-rank Tucker decomposition of large tensors using TensorSketch. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/45a766fa266ea2ebeb6680fa139d2a3d-Paper.pdf`.

Xiangrui Meng and Michael W Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 91–100, 2013.

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

Rachel Minster, Zitong Li, and Grey Ballard. Parallel randomized Tucker decomposition algorithms. *SIAM Journal on Scientific Computing*, 46(2):A1186–A1213, 2024.

Jayadev Misra and David Gries. Finding repeated elements. *Science of Computer Programming*, 2(2):143–152, 1982.

Cameron Musco and Christopher Musco. Randomized block Krylov methods for stronger and faster approximate singular value decomposition. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/1efa39bcaec6f3900149160693694536-Paper.pdf`.

Jelani Nelson and Huy L Nguyên. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 117–126. IEEE, 2013.

Sameer A Nene, Shree K Nayar, and Hiroshi Murase. Columbia object image library (COIL-100). *Columbia University, New York, NY, USA, Tech. Rep.*, 1996.

I. V. Oseledets and S. V. Dolgov. Solution of linear systems and matrix inversion in the TT-format. *SIAM Journal on Scientific Computing*, 34(5):A2718–A2739, 2012.

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

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

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

Holger Rauhut, Reinhold Schneider, and Željka Stojanac. Low rank tensor recovery via iterative hard thresholding. *Linear Algebra and its Applications*, 523:220–262, 2017.

Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences*, 105(36):13212–13217, 2008.

Arvind K Saibaba. HOID: higher order interpolatory decomposition for tensors based on Tucker representation. *SIAM Journal on Matrix Analysis and Applications*, 37(3):1223–1249, 2016.

Berkant Savas and Lars Eldén. Handwritten digit classification using higher order singular value decomposition. *Pattern recognition*, 40(3):993–1003, 2007.

D. Savostyanov and I. V. Oseledets. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. In *the 2011 International Workshop on Multidimensional (nD) Systems*, pages 1–8, 2011.

Gil Shabat, Yaniv Shmueli, Yariv Aizenbud, and Amir Averbuch. Randomized LU decomposition. *Applied and Computational Harmonic Analysis*, 44(2):246–272, 2018.

Tianyi Shi, Maximilian Ruth, and Alex Townsend. Parallel algorithms for computing the tensor-train decomposition. *SIAM Journal on Scientific Computing*, 45(3):C101–C130, 2023.

Danny C Sorensen and Mark Embree. A DEIM induced CUR factorization. *SIAM Journal on Scientific Computing*, 38(3):A1454–A1482, 2016.

Rakshith S Srinivasa, Mark A Davenport, and Justin Romberg. Localized sketching for matrix multiplication and ridge regression. *arXiv preprint arXiv:2003.09097*, 2020.

Yiming Sun, Yang Guo, Charlene Luo, Joel Tropp, and Madeleine Udell. Low-rank Tucker approximation of a tensor from streaming data. *SIAM Journal on Mathematics of Data Science*, 2(4):1123–1150, 2020.

Panagiotis Symeonidis. ClustHOSVD: Item recommendation by combining semantically enhanced tag clustering with tensor HOSVD. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(9):1240–1251, 2015.

Dan Teng and Delin Chu. A fast frequent directions algorithm for low rank approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(6):1279–1293, 2018.

Joel A Tropp. Improved analysis of the subsampled randomized Hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126, 2011.

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

Charalampos E Tsourakakis. MACH: Fast randomized tensor decompositions. In *Proceedings of the 2010 SIAM international conference on data mining*, pages 689–700. SIAM, 2010.

Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966. ISSN 0033-3123.

Nick Vannieuwenhoven, Raf Vandebril, and Karl Meerbergen. A new truncation strategy for the higher-order singular value decomposition. *SIAM Journal on Scientific Computing*, 34(2):1027–1052, 2012.

M Alex O Vasilescu and Demetri Terzopoulos. Multilinear image analysis for facial recognition. In *2002 International Conference on Pattern Recognition*, volume 2, pages 511–514. IEEE, 2002.

M Alex O Vasilescu and Demetri Terzopoulos. Multilinear subspace analysis of image ensembles. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–93. IEEE, 2003.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

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

Chenhao Wang, Qianxin Yi, Xiuwu Liao, and Yao Wang. An improved frequent directions algorithm for low-rank approximation via block Krylov iteration. *IEEE Transactions on Neural Networks and Learning Systems*, 35(7):9428–9442, 2023.

Shusen Wang and Zhihua Zhang. Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling. *The Journal of Machine Learning Research*, 14(1): 2729–2769, 2013.

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.

Chuanfu Xiao and Chao Yang. RA-HOOI: Rank-adaptive higher-order orthogonal iteration for the fixed-accuracy low multilinear-rank approximation of tensors. *Applied Numerical Mathematics*, 201:290–300, 2024.

Yu Zhang, Guoxu Zhou, Qibin Zhao, Andrzej Cichocki, and Xingyu Wang. Fast nonnegative tensor factorization based on accelerated proximal gradient and low-rank approximation. *Neurocomputing*, 198:148–154, 2016.

Guoxu Zhou, Andrzej Cichocki, and Shengli Xie. Fast nonnegative matrix/tensor factorization based on low-rank approximation. *IEEE Transactions on Signal Processing*, 60(6): 2928–2940, 2012.

Guoxu Zhou, Andrzej Cichocki, and Shengli Xie. Decomposition of big tensors with low multilinear rank. *arXiv preprint arXiv:1412.1885*, 2014.

Guoxu Zhou, Andrzej Cichocki, Qibin Zhao, and Shengli Xie. Efficient nonnegative Tucker decompositions: Algorithms and uniqueness. *IEEE Transactions on Image Processing*, 24(12):4990–5003, 2015.