Sample Efficient Learning of Factored Embeddings of Tensor Fields

Taemin Heo **Chandrajit Bajaj**
Department of Computer Science and Oden Institute of Computational Engineering and Sciences
University of Texas at Austin, Texas, USA

Abstract

Data tensors of orders 2 and greater are now routinely being generated. These data collections are increasingly huge and growing. Many scientific and medical data tensors are tensor fields (e.g., images, videos, geographic data) in which the spatial neighborhood contains important information. Directly accessing such large data tensor collections for information has become increasingly prohibitive. We learn approximate full-rank and compact tensor sketches with decompositive representations providing compact space, time and spectral embeddings of tensor fields. All information querying and post-processing on the original tensor field can now be achieved more efficiently and with customizable accuracy as they are performed on these compact factored sketches in latent generative space. We produce optimal rank-$\boldsymbol{r}$ sketchy Tucker decomposition of arbitrary order data tensors by building compact factor matrices from a sample-efficient sub-sampling of tensor slices. Our sample efficient policy is learned via an adaptable stochastic Thompson sampling using Dirichlet distributions with conjugate priors.

1 INTRODUCTION

Data tensors of orders 2 and greater are routinely being generated. These data collections are increasingly huge and growing. For instance, a North American Regional Reanalysis (NARR) has been collecting 70 climate variables every 3 hours from 1979 to the present, and it is currently at a total size of 29.4 Terabytes (Mesinger et al., 2006). These data tensors are tensor fields in which each location of data contains important information. Directly accessing such large data tensor collections for information has become increasingly prohibitive. Approximate low rank and low memory representation (compact) generation of tensor sketches provide a space and time efficient alternative as all information querying and post-processing with high accuracy is performed on the sketches.

Randomized sketching algorithms have been a popular approach for producing a low rank approximation of very large matrices and tensors (Woolfe et al., 2008; Halko et al., 2011; Woodruff et al., 2014; Cohen et al., 2015; Boutsidis et al., 2016; Tropp et al., 2017). The algorithms are faster than the singular value decomposition (SVD) while keeping the approximation accuracy similar to the SVD. Recently, SketchyCoreSVD has been proposed to further speed up the computation and reduce the memory requirement by building random sketches only from sub-sampled and projected columns and rows (Bajaj et al., 2019). It advances SketchySVD (Tropp et al., 2019) by exploiting redundancy in the data matrix, which can be characterized by incoherence. SketchySVD had been extended to compute the approximate low-rank Tucker decomposition of tensors (Sun et al., 2020). Using similar ideas as in SketchyCoreSVD (Bajaj et al., 2019), we derive a Randomized SketchyCoreTucker (R-SCT) decomposition of tensor fields and a randomized sketchy version of the Tensor Train decomposition (R-SCTT) in the Appendix.

R-SCT is mathematically proven and computationally advantageous, but its performance has an inherent variance due to the random sampling protocol and user-defined sub-sampling ratios. R-SCT randomly samples data subsets from the original data tensor, and thus it can be suboptimal and inefficient for sparse structured data. In many modern scientific data, meaningful information is often concentrated in small regions — e.g., specific patterns of tissue in medical scans and extreme climate events such as storms or droughts in satellite climate data. Sampling irrelevant or duplicate subsets

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

yields unnecessary memory consumption without accuracy improvement. Moreover, the performance of the method can be impacted by additional requirements of optimal sampling ratios given available computational resources.

While randomized sampling based Tucker decompositions have been studied in the past, most algorithms sample *columns* of matrix unfoldings (Ahmadi-Asl et al., 2021). Also, most randomized sampling based algorithms utilize a prior multi-variate Gaussian distribution (Drineas and Mahoney, 2007), since these are theoretically the fastest. However, they are suboptimal over the sampling sequence and the total required sample size is inefficient. Other algorithms utilize measures such as leverage scores (Saibaba, 2016; Perros et al., 2015; Cheng et al., 2016) to infer an informativeness distribution. These algorithms use more information, but they are problematic for higher order and very large sized data tensors.

To overcome several of the above challenges, we present a new method called *Progressive Sketching* that progressively samples *row* vectors of matrix unfolding of input tensor (i.e., order $K-1$ tensor slice of order K input tensor) from the actively updated informativeness distribution. By doing so, we actively learn an optimal sub-sampling sequence policy for factored tensor sketches. By learning a sample efficient and near-optimal sub-sampling policy, one can then easily provide the best low rank and low memory approximation of the tensor fields with the smallest sample use.

Progressive SketchyCoreTucker sketching (P-SCT) produces more accurate, low rank approximations than R-SCT using the same amount of input data subsets. In other words, P-SCT yields a sample efficient factored tensor sketch. Our progressive learned factored sketching can be applied to any randomized sketching based machine learning algorithm, and formulated as learned sequential decision making agents. In this paper we first use a Bayesian Thompson sampling framework. Our Thompson sampling framework can be extended to a local uncertainty framework for contextual bandits (Wang and Zhou, 2020). Deep reinforcement learning algorithms, such as Soft Actor-Critic (Haarnoja et al., 2018a,b; Christodoulou, 2019) or Soft Q-learning with mutual information regularization (Grau-Moya et al., 2019), can be used for progressive sketching based on the algorithm introduced in this paper. We validate the performance of P-SCT on various large static and time-varying datasets while quantitatively comparing it to a class of randomized decomposition methods.

Overall, the main contributions of this paper are the following:

- A new method of progressive sketching (P-SCT) applicable to tensor fields, to further optimize the performance of randomized sketching, by additionally learning an optimal Thompson sampling policy for tensor sketching, using Dirichlet distribution;
- P-SCT is a learned generation of compact (low memory) and effective latent space embeddings. Factored tensor can be universally utilized for several very large data driven tasks;
- By complexity analysis and empirical comparisons, we show how progressive sketching can further optimize the randomized sketching to produce accurate low rank approximations. Moreover, P-SCT utilizes a much smaller number of data fiber samples than R-SCT and other full-scan tensor factorization algorithms.

2 BACKGROUND

2.1 Tucker Decomposition and Higher Order SVD (HOSVD)

Tucker decomposition has been considered as a multilinear generalization of SVD, and HOSVD is a constrained Tucker decomposition that ensures the orthogonality of factor matrices and all-orthogonality of the core tensor (De Lathauwer et al., 2000). For order 3 tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, the *matrix unfolding* $\boldsymbol{A}_{(1)} \in \mathbb{R}^{n_1 \times n_2 n_3}$ contains the element a_{i_1,i_2,i_3} at the position with row number i_1 and column number equal to $i_2 \times n_3 + n_2$. Other two matrix unfoldings $\boldsymbol{A}_{(2)}$ and $\boldsymbol{A}_{(3)}$ can be defined in the same manner. Then, we define *k-mode vectors*, defined as the n_k-dimensional vectors obtained from $\mathcal{A}$ by varying the index i_k and keeping the other indices fixed. With this definition, the 1-mode product of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ by a matrix $\boldsymbol{U} \in \mathbb{R}^{m_1 \times n_1}$, denoted by $\mathcal{A} \times_1 \boldsymbol{U}$ is an $(m_1 \times n_2 \times n_3)$-tensor of which the entries are given by $\sum_{i_1} a_{i_1,i_2,i_3} u_{j_1,i_1}$. Other two mode product $\mathcal{A} \times_2 \boldsymbol{U}$ and $\mathcal{A} \times_2 \boldsymbol{U}$ can be also defined accordingly. Finally, order 3 SVD can be stated as following Theorem 1.

Theorem 1 (see (De Lathauwer et al., 2000))**.** *Every order 3 tensor $\mathcal{A}$ can be written as the product*

$$\mathcal{A} = \mathcal{S} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \times_3 \boldsymbol{U}^{(3)},$$

in which $\boldsymbol{U}^{(k)}$ is a unitary $(n_k \times n_k)$-matrix; $\mathcal{S}$ is a $(n_1 \times n_2 \times n_3)$-tensor of which subtensors $\mathcal{S}_{i_k=\alpha}$, obtained by fixing the kth index to α, have the property of all-orthogonality and ordering for all possible k.

The algorithm for calculating core tensor $\mathcal{S}$ and scaling matrices $\boldsymbol{U}^{(k)}$ is given in the Algorithm 1.

Algorithm 1 HOSVD

```
Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$
Output: $\mathcal{S}, \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}$
for $k = 1$ to 3 do
    $\boldsymbol{U}^{(k)} \leftarrow r_k$ leading left singular vectors of $\boldsymbol{A}_{(k)}$
end for
Set $\mathcal{S} = \mathcal{A} \times_1 \boldsymbol{U}^{(1)T} \times_2 \boldsymbol{U}^{(2)T} \times_3 \boldsymbol{U}^{(3)T}$
```

2.2 Thompson Sampling (TS)

Thompson sampling utilizes online decision making where actions are taken sequentially in a manner that must balance between exploiting what is known to maximizing immediate performance and exploring to accumulate new information that may improve future performance (Russo et al., 2018; Agrawal and Goyal, 2012; Chapelle and Li, 2011; Thompson, 1933).

2.3 Uniqueness of R-SCT and P-SCT

Ahmadi-Asl et al. (2021) categorized randomized algorithms for the Tucker decomposition and HOSVD into four groups: **Random projection** (Che and Wei, 2019; Minster et al., 2020; Zhou et al., 2014; Sun et al., 2021; Wolf, 2019; Batselier et al., 2018; Che et al., 2021), **Sampling** (Caiafa and Cichocki, 2010; Sun et al., 2009; Oseledets et al., 2008; Saibaba, 2016; Tsourakakis, 2010; Song et al., 2019; Perros et al., 2015; Traoré et al., 2019), **Random least-squares** (Malik and Becker, 2018), and **Count sketch** (Wang et al., 2015; Gittens et al., 2020; Malik and Becker, 2018, 2020). Numerical experiments showed that sampling group outperform random projection group in computation time with approximately the same results (Ahmadi-Asl et al., 2021). The random projection based algorithms are not suitable for huge tensors stored outside-of-cores since they need to pass the original data tensor multiple times. Randomized pass-efficient algorithms have been introduced to overcome this issue by minimizing the number of passes (Halko et al., 2011; Woodruff et al., 2014; Boutsidis et al., 2016; Upadhyay, 2018), but none utilize the sampling approach for the solution. Randomized SketchyCoreTucker (R-SCT) is a unique algorithm that combines random projection and sampling to utilize the advantage of both approaches and overcome limitations rooted in using them separately.

Many sampling-based algorithms compute each factor matrix by sampling the "*columns*" of the corresponding unfolding matrix or equivalently sampling the fibers of the original data tensor. However, our P-SCT samples the "*rows*" of the unfolding matrix or equivalently sampling the tensor slices. By doing so, P-SCT assures all sampled slices are intersected with each other and progressively updates the approximation of the core from the intersection of all sampled slices.

One main differentiation of our Progressive Sketchy-CoreTucker (P-SCT) is its sample efficiency earned by active learning the informativeness distribution utilizing the idea of Thompson sampling. Many sampling-based algorithms use different sampling protocols to minimize the number of samples and computation time, but none use active learning for progressiveness. Sampling based on leverage scores are proposed in (Saibaba, 2016; Perros et al., 2015; Cheng et al., 2016). These algorithms are not sample efficient due to prior scanning entire original data tensor to compute the leverage scores. CUR decomposition can be considered a sampling technique with a difference that the sampling procedure is performed heuristically instead of randomly (Caiafa and Cichocki, 2010; Oseledets, 2011; Saibaba, 2016). It is known that CUR approximations are less accurate than the SVD, and the quality of decomposition quite depends on selection of fibers. The optimal selection has been considered an NP-hard problem, and thus heuristic algorithms have been used for computing suboptimal solutions. Our progressive sketching approach can be applied to CUR decomposition to overcome this issue too. Recently, Song et al. (2019) extended matrix CUR decomposition to tensor CURT decomposition and proposed an randomized algorithm to compute a low-rank CURT approximation to a tensor (Song et al., 2019). But this method does not use active learning to achieve sample efficiency.

Random least-squares and count-sketch groups are a randomized algorithms for ALS-based computation of the best mulitilinear rank approximation such as Higher-Order Orthogonal Iteration (HOOI) (De Lathauwer et al., 2000). A randomized least-squares HOOI algorithm is proposed in (Malik and Becker, 2018), but numerical experiment in (Ahmadi-Asl et al., 2021) showed that sampling group have better computation efficiency. Count-sketch technique has been implemented in several algorithms, mostly for CANDECOMP/PARAFAC (CP) decomposition (Wang et al., 2015; Gittens et al., 2020; Malik and Becker, 2020).

Other prior work on approximating tensor decompositions include Oh et al. (2018) which proposed P-TUCKER, a scalable Tucker decomposition that resolves the computational and memory bottleneck of ALS by row-wise updating factor matrices with parallelization (Oh et al., 2018). FREDE is a graph embedding based on matrix sketching that applies Frequent Directions — a deterministic matrix sketching in the row-updates model — on a matrix-factorization interpretation of a neural embedding, VERSE (Tsitsulin et al., 2021). Others introduced a randomized CP decomposition algorithms (Wang et al., 2015; Gittens et al., 2020; Malik and Becker, 2020). The CP decomposition factorizes a tensor into a sum of component rank-one tensors (Kolda and Bader, 2009), and it is a special case of Tucker decomposition with hyper-diagonal core tensor. Another type of tensor decomposition is the Tensor-Train (TT) decomposition (Oseledets, 2011) which requires iterative matrix SVD computation, and thus SketchyCoreSVD or any randomized matrix SVD algorithm can be applied. Tucker, CP and TT decompositions produces different factored forms, each suitable for different generative applications.

3 PROGRESSIVE SKETCHING

3.1 Progressive SketchyCoreTucker (P-SCT)

P-SCT is for any order K tensor field, so let us assume order 3 tensor $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ for explanation purposes, which is huge in size, and its spectrum is decaying fast enough for low rank approximation. Its matricizations along the three dimensions are denoted by $\boldsymbol{A}_{(1)} \in \mathbb{R}^{N_1 \times (N_2 N_3)}$, $\boldsymbol{A}_{(2)} \in \mathbb{R}^{N_2 \times (N_3 N_1)}$, $\boldsymbol{A}_{(3)} \in \mathbb{R}^{N_3 \times (N_1 N_2)}$. Let us denote row space of $\boldsymbol{A}_{(k)}$ as $Row_k = \left\{\boldsymbol{A}_{(k)}^{(1,:)}, \ldots, \boldsymbol{A}_{(k)}^{(N_k,:)}\right\}$. Then, Row_k where $k = 1, 2, 3$ are bandit arms of TS. As an action, we choose a Row_k and sample from it without replacement. Note that a row vector of a matrix unfolding is a slice matrix in original geometric dimension of order 3 tensor. Thus, P-SCT samples order $K-1$ slice tensors to sketch an order K tensor field.

The amount of information contained in the samples ($\equiv$ order $K-1$ slice tensors) is the reward of the action. An empirical variance of data in the samples would be the first option for estimating the amount of information. However, it only provides us with the overall variation without considering directional variation along with each dimension. In order words, the variance ignores the different variations in spatial, temporal, and spectral dimensions of the data tensors. For tensor decomposition and our progressive sketching, this different information in different dimensions should be effectively and efficiently measured from the sampled data to learn the sampling policy in TS.

For Tucker decomposition, the information in different dimensions can be measured as the accumulated variation in data fibers along with $K-1$ dimensions. For instance, let us assume there is a sample $\boldsymbol{A}_{(1)}^{(\delta_1,:)}$ which is equivalent to the slice matrix $\mathcal{A}^{(\delta_1,:,:)}$. Both rows and columns (i.e., data fibers) of $\mathcal{A}^{(\delta_1,:,:)}$ are used in Tucker decomposition. We define the Sum of Absolute Differences (SAD) to capture the accumulated variation in these data fibers. The computation of SAD has a similar time complexity with the variance where SAD of $\boldsymbol{A}_{(1)}^{(\delta_1,:)} \equiv \mathcal{A}^{(\delta_1,:,:)}$ is

$$\frac{1}{N_2 N_3}\left(\sum_{j=1}^{N_3}\sum_{i=1}^{N_2-1}\left|\mathcal{A}^{(\delta_1,i,j)} - \mathcal{A}^{(\delta_1,i+1,j)}\right| + \sum_{i=1}^{N_2}\sum_{j=1}^{N_3-1}\left|\mathcal{A}^{(\delta_1,i,j)} - \mathcal{A}^{(\delta_1,i,j+1)}\right|\right). \tag{1}$$

The absolute differences of data in data fibers of $\mathcal{A}^{(\delta_1,:,:)}$ are accumulated and normalized by the size of slice matrix. The normalization is required because what we want to measure is the amount of information per usage of the memory for the sample efficient Tucker decomposition. SAD of samples from other Row_k can be calculated in the same manner.

Dirichlet distribution is used to balance exploitation and exploration of Row_k. We update the concentration parameters of Dirichlet distribution by the entropy of normalized SAD of samples from Row_k. By doing so, P-SCT samples more from Row_k of which information is uniformly distributed rather than concentrated in a narrow region. It effectively reduces the mutual information between samples.

At first, we assume concentration parameters of the Dirichlet distribution as unity to make the distribution equivalent to the uniform distribution $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \alpha_3\} = \{1, 1, 1\}$. Then, sampling ratios $\boldsymbol{p} = \{p_1, p_2, p_3\}$ are drawn from the $Dirich$ $(\boldsymbol{\alpha})$. By multipling the batch-size per round N_{batch} to $\boldsymbol{p}$, we get the number of samples $\boldsymbol{n} = \{n_1, n_2, n_3\}$ that we newly sample from each Row_k.

We utilize SAD once again here for the sampling. Rather than randomly sample, we assign weights by SAD. As a result, P-SCT searches samples that contain the highest amount of information. Combined two search schemes — 1) TS with Dirichlet distribution that balances Row_k selection minimizing the mutual information between samples and 2) SAD-weighted sampling that maximizes the use of information of samples — is the core of P-SCT algorithm that learns optimal subsampling policy for streaming tensor sketching.

At the first round, we know nothing about the input tensor. Thus, weights are set as unity that can be denoted by $\boldsymbol{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}, \boldsymbol{w}_k = \{1, ..., 1\}, n(\boldsymbol{w}_k) = N_k$. where $\boldsymbol{w_k}$ is weights for Row_k and $n(\boldsymbol{w}_k)$ denotes the length of weights. Let us define sampled indices $\boldsymbol{\Delta} = \{\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2, \boldsymbol{\Delta}_3\}$ and unsampled indices $\boldsymbol{\Omega} = \{\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2, \boldsymbol{\Omega}_3\}$ and an allowed total number of samples N_{allow}. Then, the main steps of P-SCT are the following. In the algorithm, $\lfloor\cdot\rfloor$ represents the floor function.

1. *Sample from Row_k*
   - Draw sampling ratios $\boldsymbol{p} = \{p_1, p_2, p_3\}$ from *Dirich* $(\boldsymbol{\alpha})$.
   - For Row_k, sample $n_k = \lfloor p_k N_{batch} \rfloor$ indices from unsampled indices $\boldsymbol{\Omega}_k$ following weights $\boldsymbol{w}_k$. Denote the indices of the sampled rows to be $\boldsymbol{\Delta}_k$.
2. Compute SAD and update TS parameters
   - Compute SAD of $\boldsymbol{A}_{(k)}^{(\boldsymbol{\Delta}_k,:)}$. Denote SAD of $\boldsymbol{A}_{(k)}^{(\delta_k,:)}, \delta_k \in \boldsymbol{\Delta}_k$ as $SAD(\delta_k)$.
   - Compute SAD entropy $\mathcal{H}_k = -\sum_{\delta_k \in \Delta_k} \frac{SAD(\delta_k)}{\sum_{\delta_k} SAD(\delta_k)} \log\left(\frac{SAD(\delta_k)}{\sum_{\delta_k} SAD(\delta_k)}\right)$.
   - Update the concentration parameter $\alpha_k \leftarrow \mathcal{H}_k$.
   - Update the unsampled indices $\boldsymbol{\Omega}_k \leftarrow \boldsymbol{\Omega}_k \backslash \boldsymbol{\Delta}_k$.
   - Linearly interpolate SAD of unsampled rows $SAD(\boldsymbol{\Omega}_k)$ based on the index differences using $SAD(\boldsymbol{\Delta}_k)$.
   - Set $SAD(\boldsymbol{\Delta}_k) = 0$ to avoid sampling duplicates, and update $\boldsymbol{w}_k$ by normalized SAD.
   - Repeat Steps 1 and 2 until $n(\boldsymbol{\Delta}_1) + n(\boldsymbol{\Delta}_2) + n(\boldsymbol{\Delta}_3) > N_{allow}$.
3. *Final SketchyCoreTucker*
   - Compute rank-$\boldsymbol{r}$ approximation using SketchyCoreTucker (see Supplementary).
   - Instead of uniformly sampling rows of $\boldsymbol{A}_{(1)}$,$\boldsymbol{A}_{(2)}$, $\boldsymbol{A}_{(3)}$, use $\boldsymbol{\Delta}_1$, $\boldsymbol{\Delta}_2$, $\boldsymbol{\Delta}_3$.
   - For the columns of $\boldsymbol{A}_{(1)}$,$\boldsymbol{A}_{(2)}$, $\boldsymbol{A}_{(3)}$, use corresponding fibers crossing the intersection $\mathcal{A}^{(\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2, \boldsymbol{\Delta}_3)}$.
   - Dimension reduction parameters $\boldsymbol{k}$ and $\boldsymbol{s}$ can be decided as
   $$\boldsymbol{k} = \left\lfloor \boldsymbol{r} + \frac{1}{3}(\boldsymbol{n} - \boldsymbol{r}) \right\rfloor, \quad \boldsymbol{s} = \left\lfloor \boldsymbol{r} + \frac{2}{3}(\boldsymbol{n} - \boldsymbol{r}) \right\rfloor,$$
   where $\boldsymbol{n} = \{n(\boldsymbol{\Delta}_1), n(\boldsymbol{\Delta}_2), n(\boldsymbol{\Delta}_3)\}$.

3.2 Complexity Analysis

Table 1 shows the complexity of R-SCT, P-SCT, and full-scan algorithms (RP-HOSVD, R-STHOSVD, R-PET, R-ST, and R-HOID (Ahmadi-Asl et al., 2021)), an ALS-based algorithm (RP-HOOI Ibid.), and SkethcyCore version of Tensor Train decomposition (R-SCTT) for the order 3 tensor case. For a simple comparison, we assume that $N_1 = N_2 = N_3 = N$, $r_1 = r_2 = r_3 = r$, $m_1 = m_2 = m_3 = m$, $s_1 = s_2 = s_3 = s$, and $k_1 = k_2 = k_3 = k$. Also, it is assumed that the same $b = N_{batch}/3$ rows are sampled from each Row_k every iteration for P-SCT. Thus, a product of total number of iteration I and b is m. Also, note that $N > m > r > s > k$.

3.3 Theoretical Guarantees

Our proofs are based on the framework of SketchyCoreSVD, as outlined in (Bajaj et al., 2019), with modifications to accommodate varying numbers of random subsamples from each dimension of the data tensor.

Suppose $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ has an SVD of the following form

$$\boldsymbol{U\Sigma V}^* = \begin{bmatrix}\boldsymbol{U}_1 & \boldsymbol{U}_2\end{bmatrix} \begin{bmatrix}\boldsymbol{\Sigma}_1 & \\ & \boldsymbol{\Sigma}_2\end{bmatrix} \begin{bmatrix}\boldsymbol{V}_1 & \boldsymbol{V}_2\end{bmatrix}^* \tag{2}$$

where $[[\boldsymbol{A}]]_r = \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^*$ is the best rank r approximation of $\boldsymbol{A}$ and (μ, ν)-incoherent.

SketchyCoreSVD proved that the basis computed from a randomly selected subset of the columns captures the range of $\boldsymbol{A}$ with bounded error

$$\begin{aligned} ||\boldsymbol{A} - \boldsymbol{QQ}^*\boldsymbol{A}||_F \leq (C_1(p,k,r) + 1) \cdot ||\boldsymbol{\Sigma}_2||_F + \\ C_2(p,k,r) \cdot ||\boldsymbol{\Sigma}_2||_2 \end{aligned} \tag{3}$$

with probability at least $1 - \frac{4}{r^3} - \frac{4}{k^3}$, where

$$\begin{aligned} C_1(p,k,r) &= \sqrt{\frac{6e^2}{p}} \cdot \frac{k}{k-r+1} \cdot k^{\frac{3}{k-r+1}}, \\ C_2(p,k,r) &= \sqrt{\frac{36e^2}{p}} \cdot \frac{\sqrt{k \log k}}{k-r+1} \cdot k^{\frac{3}{k-r+1}}, \end{aligned} \tag{4}$$

$k \geq r + 4$ is sketch size, $n \geq 8\mu r \log r$ is number of columns, and $p = n/N$. The same proof can be applied to matrix unfoldings $\boldsymbol{A}_{(k)}$ of data tensor $\mathcal{A}$ to show that a randomly selected subset of the columns captures the range of matrix unfoldings.

The bounded error of rank r approximation $||\boldsymbol{A} - [[\hat{\boldsymbol{A}}]]_r||_F$ also has been proved conditioned on Equation 3 in SkethcyCoreSVD. The algorithm samples the same number of rows and columns to capture the range and co-range of data matrix $\boldsymbol{A}$, but the theoretical guarantee has shown independently for each dimension of $\boldsymbol{A}$. Therefore, our proposed P-SCT also has a bounded error for the multi-linear $\boldsymbol{r}$ approximation with the different number of subsamples from each dimension.

Note that the error bound in Equation 3 is a function of $C_1(p,k,r)$, $C_2(p,k,r)$, $||\boldsymbol{\Sigma}_2||_F$, and $||\boldsymbol{\Sigma}_2||_2$. For different dimensions of data tensor, the target rank r is given, and both $C_1(p,k,r)$ and $C_2(p,k,r)$ decreases as p and k increase, i.e., as we sample more. If $||\boldsymbol{\Sigma}_2||_F$ and $||\boldsymbol{\Sigma}_2||_2$ of a dimension is greater than others, larger p and k (i.e., more samples) are required to ensure the error bound is tight as needed. This observation resembles the idea of P-SCT, where we can get sample-efficient rank-$\boldsymbol{r}$ approximation by balancing sampling between dimensions based on SAD entropy. Therefore,

Table 1: The computational complexity for decomposing order 3 data tensor. The computational benefits of SketchyCoreTucker depend on how much m is smaller than N. If the required m for target accuracy is close to N, SketchyCoreTucker has similar computational complexity to other algorithms. The main advantage of P-SCT is reducing m for target accuracy sustaining computational complexity of decomposition close to N^2.

Algorithm	Projection/Sampling	QR	Core	Total
RP-HOSVD	$O(N^3r)$	$O(Nr^2)$	$O(N^3r)$	$O(N^3r)$
RP-STHOSVD	$O(N^4)$		$O(N^3r)$	$O(N^4)$
R-PET	$O(N^3s)$	$O(N^2k)$	$O(Nsk + s^3k)$	$O(N^3s)$
R-ST	$O(r)$		$O(N^3r)$	$O(N^3r)$
R-HOID	$O(N^3k)$		$O(N^3r)$	$O(N^3r)$
RP-HOOI	$O(IN^3r)$	$O(INr^2)$	$O(IN^3r)$	$O(IN^3r)$
R-SCTT				$O(N^2mk)$
R-SCT	$O(Nmk + m^3s)$	$O(Nk^2)$	$O(msk + r^3k)$	$O(Nmk + m^3s)$
P-SCT	$O(N^2m + m^3s)$	$O(Nk^2)$	$O(msk + r^3k)$	$O(N^2m + m^3s)$

P-SCT can have a tighter error bound than R-SCT as we demonstrate with empirical results in the following section.

4 EXPERIMENTS

The proper multilinear ranks, $\boldsymbol{r}$, for each dataset are identified from the scree plots for the modes calculated by HOSVD (see Algorithm 1). There are no strong guarantees on the performance of HOSVD; however, it is widely believed to produce an approximation usually quite close to the best multilinear rank approximation. At rank r for mode k, scree plot can be computed by $scree_k(r) = \frac{1}{\|\boldsymbol{A}_{(k)}\|_F^2} \sum_{i=r+1} \sigma_i^2(\boldsymbol{A}_{(k)})$. Scree plots are shown in the Appendix.

For a rank $\boldsymbol{r} = (r_1, r_2, r_3)$ approximation $[\![\mathcal{A}]\!]_{\boldsymbol{r}}$ to $\mathcal{A}$, we measure its approximation error as

$$err = \frac{\|[\![\mathcal{A}]\!]_{\boldsymbol{r}} - \mathcal{A}\|_F^2}{\|\mathcal{A}\|_F^2}. \quad (5)$$

To the best of our knowledge, P-SCT is the first algorithm that introduces active learning in the streaming Tucker decomposition of data tensors. Thus, our performance comparison has been made two-fold. First, we compare the low-rank approximation accuracy with several full-scan algorithms with the same target rank (RP-HOSVD, RP-STHOSVD, R-PET, R-ST, R-HOID, and RP-HOOI (Ahmadi-Asl et al., 2021)), an ALS-based algorithm (RP-HOOI Ibid.), and SketchyCore version of Tensor Train decomposition algorithm (R-SCTT). We represent the distribution and mean of errors and computation times evaluated over 100 trials. The computation time reported does not include computing matrix unfoldings and the final approximation $[\![\boldsymbol{A}]\!]_r$ since we can avoid unfolding using index-wise access to the data tensor and, in practice, normally use the method only for calculating decompositions. Second, we compare the sample efficiency of our progressive sketch generation. To fairly compare the performance of P-SCT by using the same size of partial data tensor and undecided parameters, the sampling ratio of R-SCT is randomly selected in every trial. Sampling numbers of rows for R-SCT are randomly generated, and other parameters are decided in the same way described in P-SCT algorithm Step 3. We compare how low rank approximation error decreases as the number of samples increases. We represent this in the learning curves evaluated in 100 trials. To emphasize the sample efficiency of our method, the mean approximation accuracy of other algorithms that use all data have been shown together. We also provide visual comparisons on the few snapshots of data tensors in the Appendix. The experiments are executed from Python on a 64-bit Microsoft machine with an Intel i7-12700H CPU and 32 GB of RAM. Our implementation is available at `https://github.com/CVC-Lab/ProgressiveSketching`.

4.1 Cardiac Magnetic Resonance Imaging

A collection of time-varying, 2D slice stacks of Cardiac MRI is tested: $\mathcal{A} \in \mathbb{R}^{256\times176\times160}$ consists of 160 time snapshots 2D images, each of size 256×176. The data is sparsely structured, where only a portion of the 2D slice contains the beating motion of the heart. Based on the scree plots shown in the Appendix, we choose $\boldsymbol{r} = (20, 20, 5)$. We also choose $N_{allow} = 300$. Table 2 and Figure 1 show that P-SCT produces better performance than most full-scan algorithms only using approximately half of the data tensor. Figure 2 shows that P-SCT's performance converges faster than R-SCT to outperform most full-scan algorithms.

4.2 NARR Air Temperature

NARR air temperature dataset is spatiotemporal series covering the North American continent (Mesinger et al.,

Table 2: Performance comparison. Displayed values are mean ± standard deviation over 100 trials. Full-scan algorithms (RP-HOSVD, RP-STHOSVD, R-PET, R-ST, R-HOID) use all entries of the data tensor, but P-SCT uses only about half of the data tensor. Nevertheless, P-SCT produces good approximation results with acceptable computation time (sec). An ALS-based algorithm, RP-HOOI, also uses the original data tensor. Its accuracy can be adjusted by setting smaller error tolerance, but it substantially increases iteration and computational complexity. A SketchyCore version of tensor train decomposition, R-SCTT, randomly samples 50% of input. RP-STHOSVD shows a better averaged performance, but the method is easily prohibitive as the size of the data tensor grows since its complexity is $O(N^4)$. Whereas, P-SCT is more scalable with a small sample size since its complexity is $O(N^2m + m^3s)$ where $N > m > s$.

	Cardiac MRI		NARR Air Temperature		Hyperspectral Image	
	err	time	err	time	err	time
RP-HOSVD	0.22 ± 5e-2	0.23 ± 9e-3	3.2e-5 ± 5e-6	1.3 ± 3e-2	8.7e-3 ± 2e-3	0.35 ± 4e-2
RP-STHOSVD	0.085 ± 4e-4	1.0 ± 3e-2	8.2e-5 ± 2e-5	3.5 ± 0.2	3.2e-3 ± 3e-5	1.0 ± 9e-2
R-PET	2.4 ± 4e-1	0.57 ± 2e-2	3.9 ± 2e-1	2.7 ± 0.1	3.4 ± 5e-1	0.78 ± 8e-2
R-ST	0.32 ± 7e-2	0.071 ± 5e-3	5.6e-4 ± 8e-4	0.22 ± 2e-2	8.2e-3 ± 2e-3	6.0e-2 ± 1e-2
R-HOID	0.13 ± 8e-3	0.63 ± 3e-2	2.1e-5 ± 2e-6	2.9 ± 8e-2	7.4e-3 ± 4e-4	0.60 ± 7e-2
RP-HOOI	0.20 ± 4e-2	1.0 ± 4e-2	2.4e-5 ± 5e-6	2.6 ± 0.2	5.4e-3 ± 1e-3	0.57 ± 7e-2
R-SCTT	0.35 ± 7e-2	0.64 ± 3e-2	0.21 ± 2e-2	7.2 ± 0.1	0.14 ± 3e-2	1.9 ± 0.2
R-SCT	1.7 ± 9	0.63 ± 8e-2	0.10 ± 3e-1	1.6 ± 0.2	0.13 ± 0.2	0.59 ± 6e-2
P-SCT	0.094 ± 1e-2	1.4 ± 5e-2	2.4e-5 ± 5e-6	2.7 ± 0.1	1.3e-2 ± 2e-3	0.98 ± 0.1

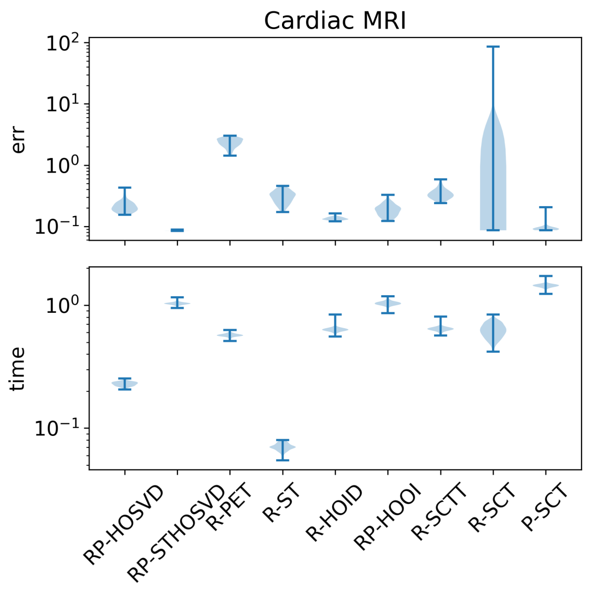


Figure 1: Performance comparison for Cardiac MRI dataset. Violin plots of error and computation time (sec) from 100 trials. (see Table 2 for more information).

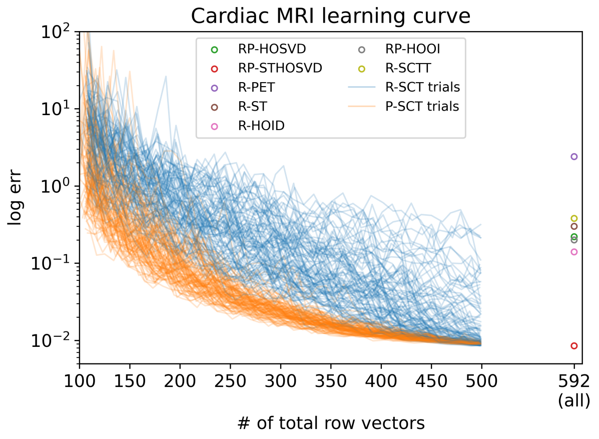


Figure 2: Learning curve comparison. P-SCT (orange curves) converges faster than R-SCT (blue curves), showing sample-efficient behavior. P-SCT's error is smaller than R-SCT with smaller variances in almost every trial, empirically showing P-SCT's tighter error bounds. P-SCT produces a more accurate tensor decomposition than most full-scan algorithms using less than half of the input tensor.

2006). Its grid resolution is 349 × 277, which is approximately 0.3 degrees (32km) resolution at the lowest latitude. It provides 3 hourly series from 1979/01/01 to the present collected from 29 pressure levels (i.e., 29 different altitudes). We selected 1 month-long series in October 2020 at 500 hpa pressure level for the experiment. Spatial coverage is also clipped to discard null values around boundaries. As a result, $\mathcal{A} \in \mathbb{R}^{(248,252,336)}$. Based on scree plots shown in Appendix, we choose $\boldsymbol{r} = (50, 10, 20)$. N_{allow} is assumed to be 400. The NARR air temperature is less sparsely structured than Cardiac MRI as wide regions' air temperature fluctuate differently. Still, Table 2, Figures 3 and 4 show good results. P-SCT's performance converges faster to outperform some full-scan algorithms showing P-SCT's ability to find the important domain of the input tensor and selectively sample them for sample-efficient tensor sketching.

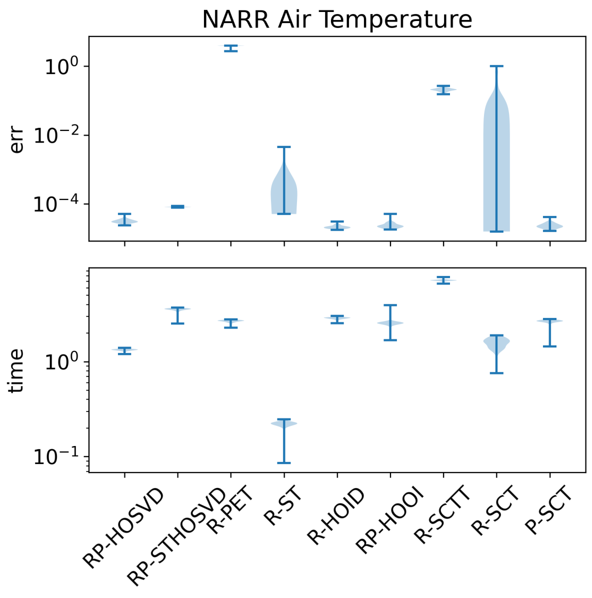


Figure 3: Performance comparison for NARR air temperature dataset. Violin plots of error and computation time (sec) from 100 trials. (see Table 2 for more information).

4.3 Hyperspectral Image

Hyperspectral image (HSI) contains a wide spectrum of light instead of simple primary colors (e.g., red, green, blue) in each pixel with the purpose of finding objects or identifying materials. For example, the Airborne Visible / Infrared Imaging Spectrometer (AVIRIS) hyperspectral sensor data were acquired on June 12, 1992, over the Purdue University Agronomy farm northwest of West Lafayette and the surrounding area (Baumgardner et al., 2015). The data includes three approximately 2-mile by 2-mile test sites that were used to identify land use over major portions of the Indian Creek and Pine Creek watersheds. We selected one near the center (site 3) with the grid resolution 145 $\times$ 145 and 220 electromagnetic spectrum channels, and thus $\mathcal{A} \in \mathbb{R}^{145\times145\times220}$. Based on scree plots shown in the Appendix, we choose $\boldsymbol{r} = (25, 25, 5)$. N_{allow} is assumed to be 300. HSI is sparsely structured, similar to NARR air temperature dataset, where the light spectrum is different for different land uses. Agian, Table 2 and Figure 5 show that P-SCT produces a good performance as other full-scan algorithms, although it only uses about half of the data tensor. Figure 6 compares progressive performance showing the sample efficiency of P-SCT.

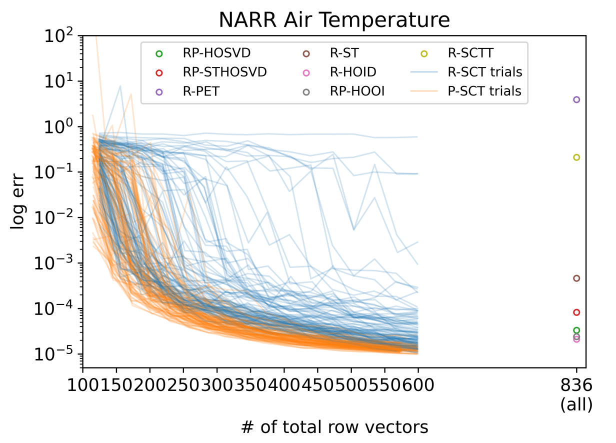


Figure 4: Learning curve comparison. P-SCT (orange curves) converges faster than R-SCT (blue curves), showing sample-efficient behavior. P-SCT's error is smaller than R-SCT with smaller variances in almost every trial, empirically showing P-SCT's tighter error bounds. P-SCT produces a more accurate tensor decomposition than most full-scan algorithms using less than half of the input tensor.

4.4 Discussion

The computation time of P-SCT is longer than R-SCT. This delay is inevitable since the progressive sketching approach adds more computations. The delay will be greater for higher order tensor or bigger sized data tensor. However, considering a combinatorial number of possible sampling ratio parameters, the amount of computation time we trade off for getting optimal sample efficient low rank approximation is sufficiently negligible. The optimally balanced performance of the progressive sketching approach is well addressed that P-SCT maintains a good accuracy and computation time compared to fast-but-erroneous (RP-HOSVD, R-ST), and accurate-but-slow (RP-STHOSVD, R-HOID) full-scan algorithms.

5 CONCLUSION

Our paper presents various low-memory, latent factored tensor field sketches via a learned sub-sampling policy. P-SCT actively learns the optimal subsampling sketch streaming protocol so as to maximize the accuracy while minimizing the space requirements of the final tensor sketch. Our numerical experiments show that P-SCT can get better performance with a limited number of samples than a random sampling based SketchyCoreTucker (R-SCT) and other full-scan algorithms (RP-HOSVD, RP-STHOSVD, R-PET, R-ST, R-HOID, and RP-HOOI from (Ahmadi-Asl et al., 2021)). R-SCT and R-SCTT were derived from a gen-

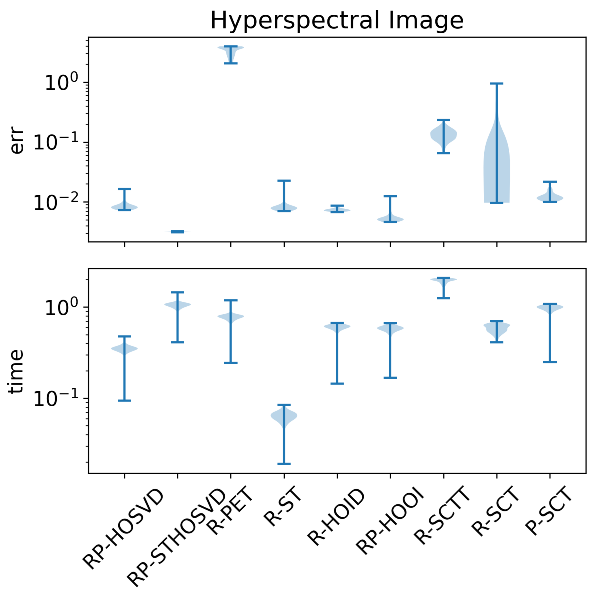


Figure 5: Performance comparison for hyperspectral image. Violin plots of error and computation time (sec) from 100 trials. (see Table 2 for more information).

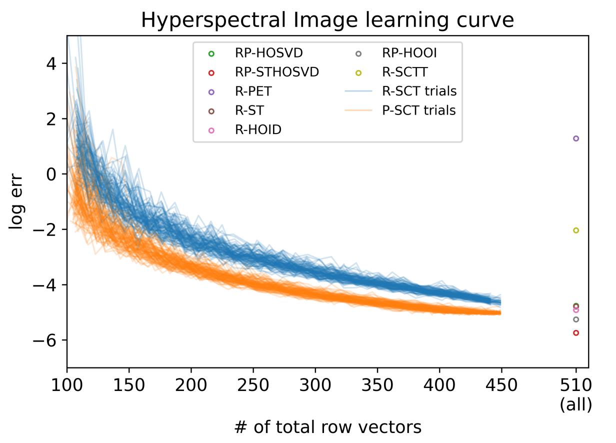


Figure 6: Learning curve comparison. P-SCT (orange curves) converges faster than R-SCT (blue curves), showing sample-efficient behavior. P-SCT's error is smaller than R-SCT with smaller variances in almost every trial, empirically showing P-SCT's tighter error bounds. P-SCT produces a more accurate tensor decomposition than most full-scan algorithms using less than half of the input tensor.

eralization of a previously published SketchyCoreSVD method. While R-SCT and R-SCTT rely on a preset sampling ratio of the tensor fibers, P-SCT does not need this preset since the algorithm actively balances the ratio of tensor fibers from each dimension to get the maximum reward. P-SCT and in comparison to R-SCT and R-SCTT, also avoids further fine tuning that was required for SCT and TT. Further details on the progressive sketching version of TT (P-SCTT) and its comparison with R-SCTT are left for future work.

Acknowledgements

This research was supported in part by a grant from the NIH DK129979, in part from the Peter O'Donnell Foundation, the Michael J. Fox Foundation, Jim Holland-Backcountry Foundation, and in part from a grant from the Army Research Office accomplished under Cooperative Agreement Number W911NF-19-2-0333. Additionally, we owe thanks to Tianming Wang for his help and insights when he was at UT Austin, and when this line of work was initiated.

References

Fedor Mesinger, Geoff DiMego, Eugenia Kalnay, Kenneth Mitchell, Perry C Shafran, Wesley Ebisuzaki, Dušan Jović, Jack Woollen, Eric Rogers, Ernesto H Berbery, et al. North american regional reanalysis. *Bulletin of the American Meteorological Society*, 87 (3):343–360, 2006.

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.

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.

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

Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 163–172. ACM, 2015.

Christos Boutsidis, David P Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 236–249, 2016.

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.

Chandrajit Bajaj, Yi Wang, and Tianming Wang. Sketchycoresvd: Sketchysvd from random subsampling of the data matrix. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 26–35. IEEE, 2019.

Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Streaming low-rank matrix approximation with an application to scientific simulation. *SIAM Journal on Scientific Computing*, 41(4):A2430–A2463, 2019.

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.

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.

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.

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.

Ioakeim Perros, Robert Chen, Richard Vuduc, and Jimeng Sun. Sparse hierarchical Tucker factorization and its application to healthcare. In *2015 IEEE International Conference on Data Mining*, pages 943–948. IEEE, 2015.

Dehua Cheng, Richard Peng, Yan Liu, and Ioakeim Perros. Spals: Fast alternating least squares via implicit leverage scores sampling. *Advances in neural information processing systems*, 29:721–729, 2016.

Zhendong Wang and Mingyuan Zhou. Thompson sampling via local uncertainty. In *International Conference on Machine Learning*, pages 10115–10125. PMLR, 2020.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018b. URL `http://arxiv.org/abs/1812.05905`.

Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

Jordi Grau-Moya, Felix Leibfried, and Peter Vrancx. Soft q-learning with mutual-information regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HyEtjoCqFX`.

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

Daniel J. Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A Tutorial on Thompson Sampling. *Found. Trends Mach. Learn.*, 11(1): 1–96, July 2018. ISSN 1935-8237.

Shipra Agrawal and Navin Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1. JMLR Workshop and Conference Proceedings, 2012.

Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson sampling. *Advances in neural information processing systems*, 24:2249–2257, 2011.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

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.

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.

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

Yiming Sun, Yang Guo, Joel A Tropp, and Madeleine Udell. Tensor random projection for low memory dimension reduction. *arXiv preprint arXiv:2105.00105*, 2021.

Alexander Sebastian Johannes Wolf Wolf. Low rank tensor decompositions for high dimensional data approximation, recovery and prediction. 2019.

Kim Batselier, Wenjian Yu, Luca Daniel, and Ngai Wong. Computing low-rank approximations of large-scale matrices with the tensor network randomized SVD. *SIAM Journal on Matrix Analysis and Applications*, 39(3):1221–1244, 2018.

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, 2021.

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.

Jimeng Sun, Spiros Papadimitriou, Ching-Yung Lin, Nan Cao, Shixia Liu, and Weihong Qian. Multivis: Content-based social network exploration through multi-way visual analysis. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 1064–1075. SIAM, 2009.

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

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

Zhao Song, David P Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2772–2789. SIAM, 2019.

Abraham Traoré, Maxime Berar, and Alain Rakotomamonjy. Singleshot: a scalable Tucker tensor decomposition. *Advances in Neural Information Processing Systems*, 32, 2019.

Osman Asif Malik and Stephen Becker. Low-rank Tucker decomposition of large tensors using TensorSketch. In *Advances in Neural Information Processing Systems*, pages 10096–10106, 2018.

Yining Wang, Hsiao-Yu Tung, Alexander J Smola, and Anima Anandkumar. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems*, pages 991–999, 2015.

Alex Gittens, Kareem Aggour, and Bülent Yener. Adaptive sketching for fast and convergent canonical polyadic decomposition. In *International Conference on Machine Learning*, pages 3566–3575. PMLR, 2020.

Osman Asif Malik and Stephen Becker. Fast randomized matrix and tensor interpolative decomposition using countsketch. *Advances in Computational Mathematics*, 46(6):1–28, 2020.

Jalaj Upadhyay. The price of privacy for low-rank factorization. *Advances in Neural Information Processing Systems*, 31, 2018.

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

Sejoon Oh, Namyong Park, Sael Lee, and Uksong Kang. Scalable tucker factorization for sparse tensors-algorithms and discoveries. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1120–1131. IEEE, 2018.

Anton Tsitsulin, Marina Munkhoeva, Davide Mottin, Panagiotis Karras, Ivan Oseledets, and Emmanuel Müller. Frede: anytime graph embeddings. *Proceedings of the VLDB Endowment*, 14(6):1102–1110, 2021.

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

Marion F Baumgardner, Larry L Biehl, and David A Landgrebe. 220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3, Sep 2015. URL `https://purr.purdue.edu/publications/1947/1`.

Supplementary Material for Sample Efficient Learning of Factored Embeddings of Tensor Fields

A Randomized SketchyCore Tensor Decompositions

A.1 Randomized SketchyCoreTucker (R-SCT)

SkechySVD has been extended to compute low-rank Tucker decomposition of tensors. One can call this method SketchyTucker. Using similar ideas as in SketchyCoreSVD, we derive a Randomized SketchyCoreTucker (R-SCT) method.

Without loss of generality, we assume $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$. Its matricizations along the three dimensions are denoted by $\boldsymbol{A}_{(1)} \in \mathbb{R}^{N_1 \times (N_2 N_3)}$, $\boldsymbol{A}_{(2)} \in \mathbb{R}^{N_2 \times (N_3 N_1)}$, and $\boldsymbol{A}_{(3)} \in \mathbb{R}^{N_3 \times (N_1 N_2)}$, respectively.

Define $\boldsymbol{r} = [r_1, r_2, r_3], \boldsymbol{k} = [k_1, k_2, k_3], \boldsymbol{s} = [s_1, s_2, s_3], \boldsymbol{m} = [m_1, , m_2, m_3], \boldsymbol{n} = [n_1, n_2, n_3]$, where $\boldsymbol{r} \preceq \boldsymbol{k} \preceq \boldsymbol{s} \preceq \min\{\boldsymbol{m}, \boldsymbol{n}\}$ and "$\preceq$" means "$\leq$" for each entry. Define $\mathbf{p} = [p_1, p_2, p_3]$ and $\mathbf{q} = [q_1, q_2, q_3]$ where $p_1 = \frac{m_1}{N_2 N_3} \in (0,1), p_2 = \frac{m_2}{N_1 N_3} \in (0,1), p_3 = \frac{m_3}{N_1 N_2} \in (0,1)$, and $q_1 = \frac{n_1}{N_1} \in (0,1), q_2 = \frac{n_2}{N_2} \in (0,1), q_3 = \frac{n_3}{N_3} \in (0,1)$.

The main steps of R-SCT are summarized in Algorithm A.1.

Algorithm A.1 Randomized SketchyCoreTucker (R-SCT)

Input: $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$, $\boldsymbol{r}, \boldsymbol{k}, \boldsymbol{s}, \boldsymbol{p}, \boldsymbol{q}$, uniformly sampled m_i column indices of $\boldsymbol{A}_{(i)}$, Θ_i, the map $\boldsymbol{\Omega}_i \in \mathbb{R}^{k_i \times m_i}$, uniformly sampled n_i row indices of $\boldsymbol{A}_{(i)}$, Δ_i, and the map $\boldsymbol{\Phi}_i \in \mathbb{R}^{s_i \times b_i}$ where $i = 1, 2, 3$.
Output: The near-optimal multi-linear $\boldsymbol{r}$ approximation,

$$[[\mathcal{A}]]_{\boldsymbol{r}} = [[\mathcal{C}]]_{\boldsymbol{r}} \times_1 \boldsymbol{Q}_1 \times_2 \boldsymbol{Q}_2 \times_3 \boldsymbol{Q}_3,$$

and the compact tensor sketch, i.e., a thumbnail of data tensor,

$$[[\mathcal{C}]]_{\boldsymbol{r}} \times_1 \boldsymbol{Q}_1^{(\Delta_1,:)} \times_2 \boldsymbol{Q}_2^{(\Delta_2,:)} \times_3 \boldsymbol{Q}_3^{(\Delta_3,:)}.$$

```
1: **Building randomized sketches**
2: $\boldsymbol{Y}_1 = \boldsymbol{A}_{(1)}^{(:,\Theta_1)} \boldsymbol{\Omega}_1^* \in \mathbb{R}^{N_1 \times k_1}$; $\boldsymbol{Y}_2 = \boldsymbol{A}_{(2)}^{(:,\Theta_2)} \boldsymbol{\Omega}_2^* \in \mathbb{R}^{N_2 \times k_2}$; $\boldsymbol{Y}_3 = \boldsymbol{A}_{(3)}^{(:,\Theta_3)} \boldsymbol{\Omega}_3^* \in \mathbb{R}^{N_3 \times k_3}$;
3: $\mathcal{Z} = \mathcal{A}^{(\Delta_1,\Delta_2,\Delta_3)} \times_1 \boldsymbol{\Phi}_1 \times_2 \boldsymbol{\Phi}_2 \times_3 \boldsymbol{\Phi}_3 \in \mathbb{R}^{s_1 \times s_2 \times s_3}$;
4: **Compute the QR decomposition**
5: $\boldsymbol{Y}_1 = \boldsymbol{Q}_1 \boldsymbol{R}_1$, where $\boldsymbol{Q}_1 \in \mathbb{R}^{N_1 \times k_1}$, and $\boldsymbol{R}_1 \in \mathbb{R}^{k_1 \times k_1}$;
6: $\boldsymbol{Y}_2 = \boldsymbol{Q}_2 \boldsymbol{R}_2$, where $\boldsymbol{Q}_2 \in \mathbb{R}^{N_2 \times k_2}$, and $\boldsymbol{R}_2 \in \mathbb{R}^{k_2 \times k_2}$;
7: $\boldsymbol{Y}_3 = \boldsymbol{Q}_3 \boldsymbol{R}_3$, where $\boldsymbol{Q}_3 \in \mathbb{R}^{N_3 \times k_3}$, and $\boldsymbol{R}_3 \in \mathbb{R}^{k_3 \times k_3}$;
8: $\mathcal{C} = \mathcal{Z} \times_1 (\boldsymbol{\Phi}_1 \boldsymbol{Q}_1^{(\Delta_1,:)})^\dagger \times_2 (\boldsymbol{\Phi}_2 \boldsymbol{Q}_2^{(\Delta_2,:)})^\dagger \times_3 (\boldsymbol{\Phi}_3 \boldsymbol{Q}_3^{(\Delta_3,:)})^\dagger \in \mathbb{R}^{k_1 \times k_2 \times k_3}$;
9: Compute the best multi-linear rank $\boldsymbol{r}$ approximation of $\mathcal{C}$, $[[\mathcal{C}]]_{\boldsymbol{r}}$, by an algorithm such as HOSVD or HOOI.
```

A.2 Randomized SketchyCoreTensorTrain (R-SCTT)

SketchyCoreTT computes the truncated SVDs in Step 3 and Step 7 in Algorithm A.2 using SketchyCoreSVD. Besides (k_1, s_1, k_2, s_2), there are two extra parameters (p_1, p_2) about sampling ratios.

Algorithm A.2 Tensor Train Decomposition

Input: $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$.

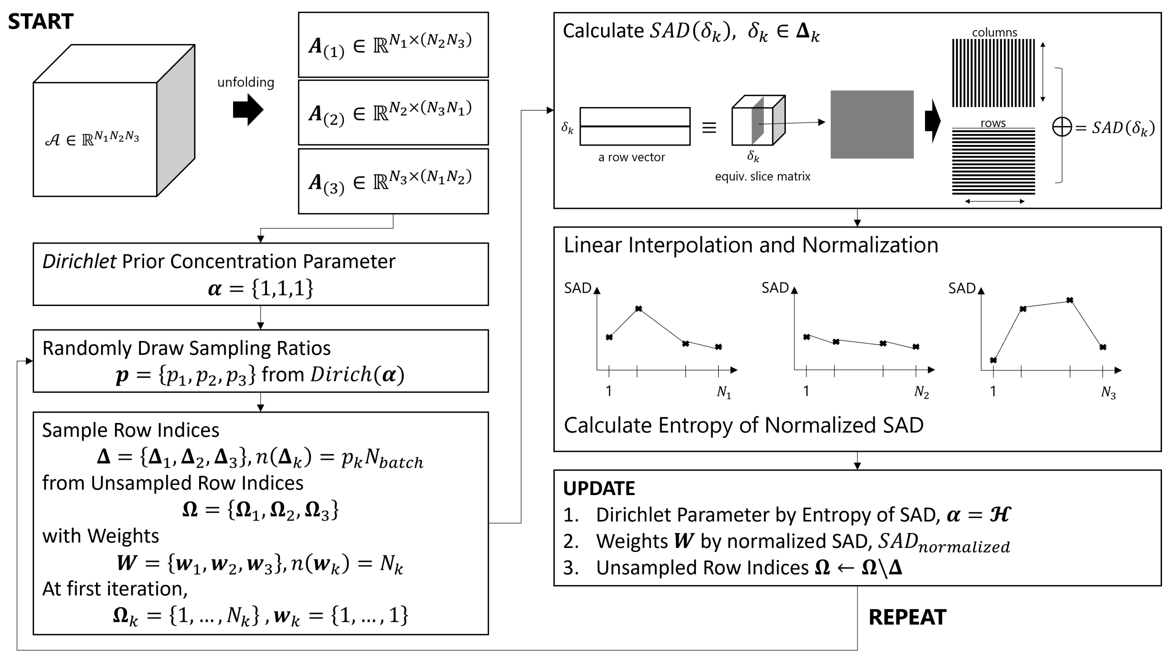
Output: Tensor $\mathcal{B}$ in TT format with factor matrices $\boldsymbol{G}_1 \in \mathbb{R}^{N_1 \times r_1}$ and $\boldsymbol{G}_3 \in \mathbb{R}^{r_2 \times N_3}$ and core $\mathcal{G}_2 \in \mathbb{R}^{r_1 \times N_2 \times r_2}$.

```
1: Initialize $\mathcal{C} = \mathcal{A}$.
2: Set $\boldsymbol{C} = \text{reshape}(\mathcal{C}, N_1, N_2 N_3)$;
3: Compute truncated SVD of $\boldsymbol{C}$ with rank $r_1$, denoted by $\boldsymbol{U}_1 \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^*$;
4: Set $\boldsymbol{G}_1 = \text{reshape}(\boldsymbol{U}_1, N_1, r_1)$;
5: Update $\boldsymbol{C} = \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^*$;
6: Set $\boldsymbol{C} = \text{reshape}(\boldsymbol{C}, r_1 N_2, N_3)$;
7: Compute truncated SVD of $\boldsymbol{C}$ with rank $r_2$, denoted by $\boldsymbol{U}_2 \boldsymbol{\Sigma}_2 \boldsymbol{V}_2^*$;
8: Set $\mathcal{G}_2 = \text{reshape}(\boldsymbol{U}_2, r_1, N_2, r_2)$;
9: Set $\boldsymbol{G}_3 = \boldsymbol{\Sigma}_2 \boldsymbol{V}_2^*$.
```

B Supplementary figures

B.1 P-SCT Algorithm

Figure B.1 illustrates the P-SCT algorithm.



START
$\mathcal{A} \in \mathbb{R}^{N_1 N_2 N_3}$
unfolding
$\boldsymbol{A}_{(1)} \in \mathbb{R}^{N_1 \times (N_2 N_3)}$
$\boldsymbol{A}_{(2)} \in \mathbb{R}^{N_2 \times (N_3 N_1)}$
$\boldsymbol{A}_{(3)} \in \mathbb{R}^{N_3 \times (N_1 N_2)}$
Dirichlet Prior Concentration Parameter $\boldsymbol{\alpha} = \{1,1,1\}$
Randomly Draw Sampling Ratios $\boldsymbol{p} = \{p_1, p_2, p_3\}$ from $Dirich(\boldsymbol{\alpha})$
Sample Row Indices $\boldsymbol{\Delta} = \{\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2, \boldsymbol{\Delta}_3\}, n(\boldsymbol{\Delta}_k) = p_k N_{batch}$
from Unsampled Row Indices $\boldsymbol{\Omega} = \{\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2, \boldsymbol{\Omega}_3\}$
with Weights $\boldsymbol{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}, n(\boldsymbol{w}_k) = N_k$
At first iteration, $\boldsymbol{\Omega}_k = \{1, \ldots, N_k\}, \boldsymbol{w}_k = \{1, \ldots, 1\}$
Calculate $SAD(\delta_k)$, $\delta_k \in \boldsymbol{\Delta}_k$
δ_k
a row vector
$\equiv$
δ_k
equiv. slice matrix
columns
rows
$\bigoplus = SAD(\delta_k)$
Linear Interpolation and Normalization
SAD
1
N_1
SAD
1
N_2
SAD
1
N_3
Calculate Entropy of Normalized SAD
UPDATE
1. Dirichlet Parameter by Entropy of SAD, $\boldsymbol{\alpha} = \boldsymbol{\mathcal{H}}$
2. Weights $\boldsymbol{W}$ by normalized SAD, $SAD_{normalized}$
3. Unsampled Row Indices $\boldsymbol{\Omega} \leftarrow \boldsymbol{\Omega} \backslash \boldsymbol{\Delta}$
REPEAT

Figure B.1: Flowchart of P-SCT algorithm. We draw sampling ratios from Dirichlet prior to balancing the exploration/exploitation of each row space of the matrix unfolding. The sum of Absolute Difference (SAD) is computed for each sampled tensor slice. The concentration parameter of Dirichlet prior and sampling weights are updated based on the SAD score and SAD entropy. By repeating this Thompson sampling scheme, P-SCT selectively samples informative regions in the input tensor.

B.2 Scree plots

The proper multilinear ranks, $\boldsymbol{r}$, for each dataset are identified from the scree plots for the modes calculated by HOSVD. At rank r for mode k, scree plot can be computed by

$$scree_k(r) = \frac{1}{\|\boldsymbol{A}_{(k)}\|_F^2} \sum_{i=r+1} \sigma_i^2(\boldsymbol{A}_{(k)}). \tag{6}$$

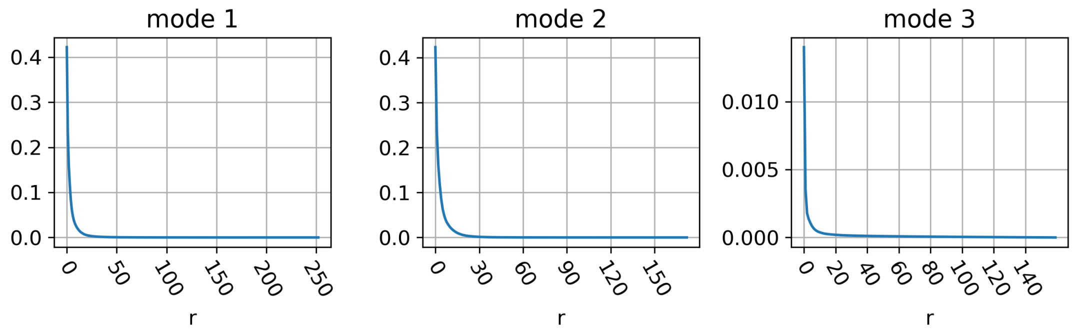


Figure B.2: Scree plots for the modes of Cardiac MRI dataset. $\boldsymbol{r} = (20, 20, 5)$ has been selected for the low rank approximation.

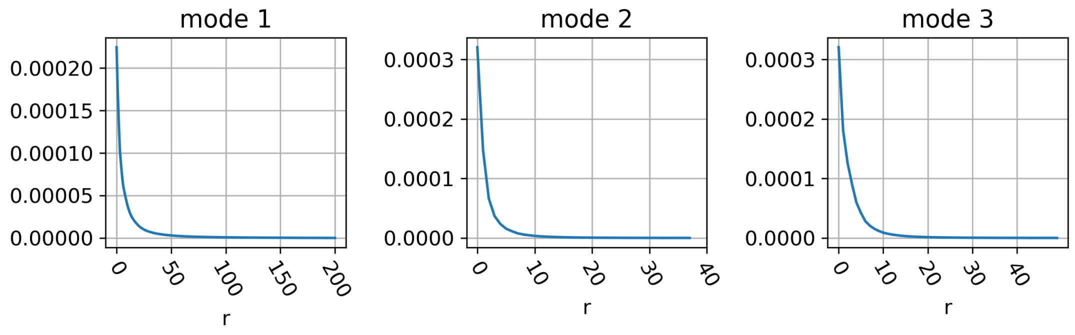


Figure B.3: Scree plots for the modes of NARR air temperature dataset. $\boldsymbol{r} = (50, 10, 20)$ has been selected for the low rank approximation.

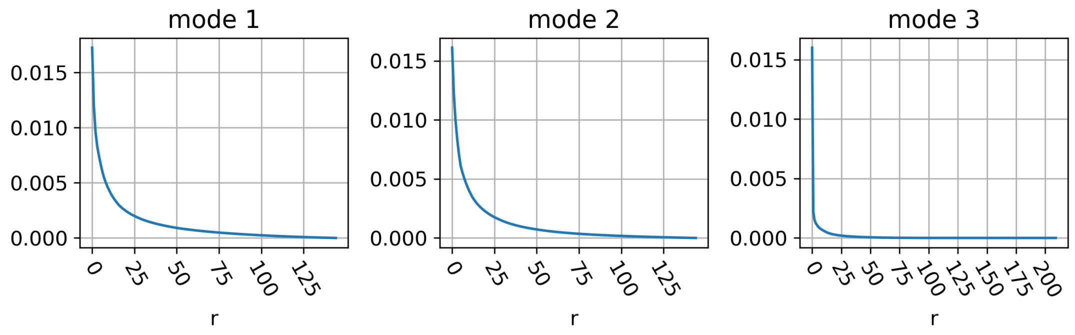


Figure B.4: Scree plots for the modes of hyperspectral image. $\boldsymbol{r} = (25, 25, 5)$ has been selected for the low rank approximation.

B.3 Snapshots

Figures B.5, B.6, and B.7 show progressive visual comparison of the snapshot of reconstructed data tensors. Low rank approximation results from R-SCT and P-SCT using the number of samples indicated in each subtitle. For all cases, P-SCT reveals the data structure faster than R-SCT.

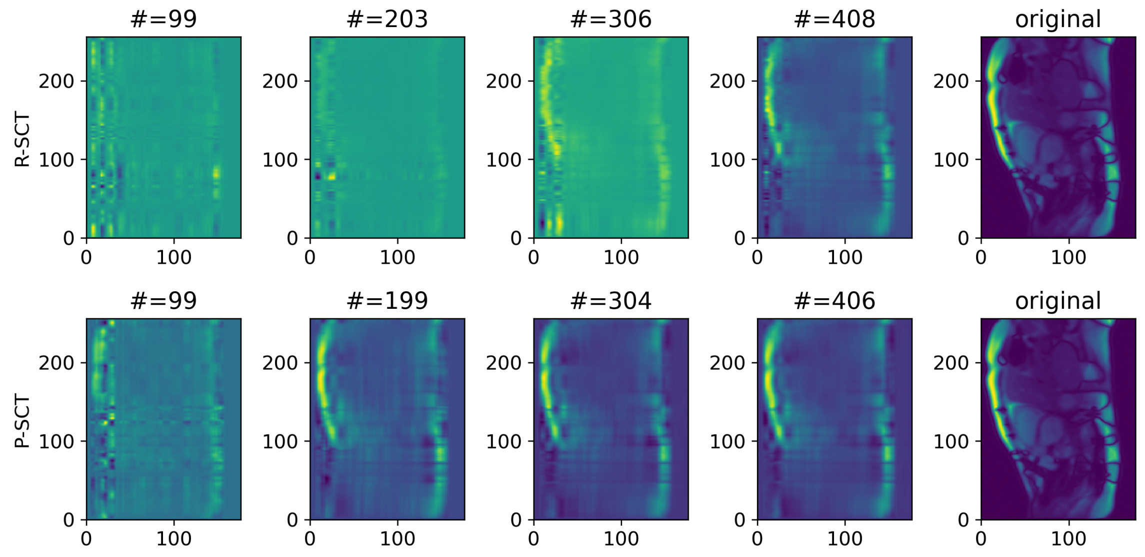


Figure B.5: Cardiac MRI snapshots at the first timestamp.

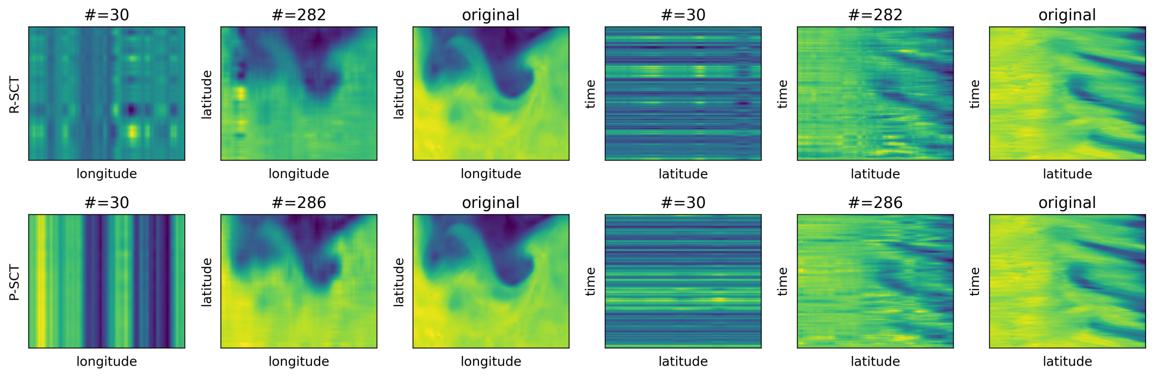


Figure B.6: NARR air temperature snapshots at the first timestamp.

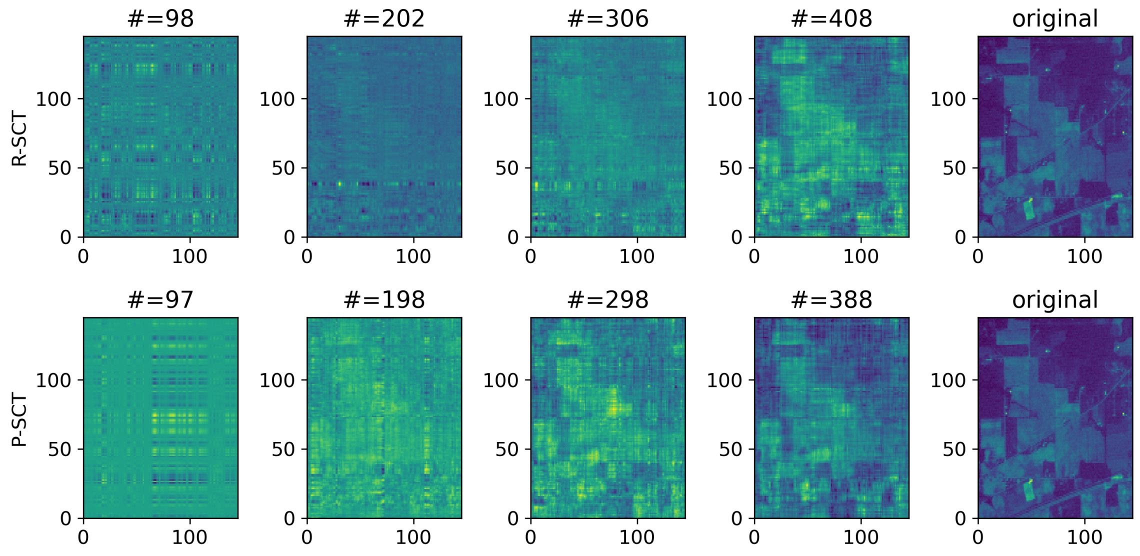


Figure B.7: Hyperspectral image snapshots of the fourth band.