10-2017

Tensor factorization for low-rank tensor completion

Pan ZHOU
Singapore Management University, panzhou@smu.edu.sg

Canyi LU

Zhouchen LIN

Chao ZHANG

Tensor Factorization for Low-Rank Tensor Completion

Pan Zhou, Canyi Lu, *Student Member, IEEE*, Zhouchen Lin, *Senior Member, IEEE*, and Chao Zhang, *Member, IEEE*

***Abstract*—Recently, a tensor nuclear norm (TNN) based method [1] was proposed to solve the tensor completion problem, which has achieved state-of-the-art performance on image and video inpainting tasks. However, it requires computing tensor singular value decomposition (t-SVD), which costs much computation and thus cannot efficiently handle tensor data, due to its natural large scale. Motivated by TNN, we propose a novel low-rank tensor factorization method for efficiently solving the 3-way tensor completion problem. Our method preserves the low-rank structure of a tensor by factorizing it into the product of two tensors of smaller sizes. In the optimization process, our method only needs to update two smaller tensors, which can be more efficiently conducted than computing t-SVD. Furthermore, we prove that the proposed alternating minimization algorithm can converge to a Karush-Kuhn-Tucker (KKT) point. Experimental results on the synthetic data recovery, image and video inpainting tasks clearly demonstrate the superior performance and efficiency of our developed method over state-of-the-arts including the TNN [1] and matricization methods [2]–[5].**

***Index Terms*—Tensor Factorization, Tensor Completion, Low-rank Factorization.**

I. Introduction

A tensor is a multi-dimensional array of numbers which is the multi-way (higher-order) generalization of vectors and matrices, thus can express more complex intrinsic structures of higher-order data. Actually, a tensor is a natural form of high-dimensional and multi-way real world data. For example, a color image can be regarded as a 3-way tensor due to its three channels; a grey scale video can also be viewed as a 3-way tensor indexed by two spatial variables and one temporal variable. So tensor analysis is of practical significance and benefits many applications in computer vision [2], [6], [7], data mining [8], collaborative filtering [9], *etc*. Low-rank tensor completion is one of the most important problems in tensor processing and analysis. It aims at filling in the missing entries of a partially observed low-rank tensor. Recent research [2], [3], [6], [7], [10]–[13] shows that high dimensional tensor data of interest, such as videos and image collections, are usually intrinsically low-rank or approximately so. Subsequently, many works utilize this property to recover tensors of incomplete observations [14]–[16] and apply it to real applications, *e.g.* hyperspectral data recovery [6], [7], [12], image/video inpainting [1]–[3], text analysis [17], [18], and multitask learning [19], *etc*.

The low-rank tensor completion problem can be regarded as an extension of the low-rank matrix completion problem [20] which aims at exactly recovering a low-rank matrix from an incomplete observation. Since the matrices of interest, *e.g.* images [13], [20], are usually known to be (approximately) low-rank, the matrix completion method minimizes the matrix rank to depict the low-rank structure of the data. Accordingly, its mathematical model is written as

$$\min_{\boldsymbol{C}} \text{rank}(\boldsymbol{C}), \quad \text{s.t.} P_\Omega(\boldsymbol{C}-\boldsymbol{M})=\boldsymbol{0}, \tag{1}$$

where the set Ω of locations corresponds to the observed entries, or more concretely, if $\boldsymbol{M}_{ij}$ is observed, then $(i,j)\in\Omega$. P_Ω is a linear operator that extracts entries in Ω and fills the entries not in Ω with zeros. The rank$(\cdot)$ is the matrix rank function. However, as directly optimizing problem (1) is NP-hard, many methods [20]–[22] approximate the rank function rank$(\boldsymbol{C})$ by its convex surrogate, *i.e.* the nuclear norm $\|\boldsymbol{C}\|_*$. Indeed, this approximation has performance guarantee. Candès *et al.* [20] proved that under certain incoherence conditions, the rank-r matrix $\boldsymbol{M}\in\mathbb{R}^{n\times n}$ with $O(n^{1.2}r\log n)$ observations can be recovered with high probability by minimizing the matrix nuclear norm. Then Chen *et al.* [23] improved the sample complexity of recovering a semidefinite matrix to $O(nr\log^2 n)$. But these nuclear norm minimization methods require computing singular value decomposition (SVD) of matrix data, which is very computationally expensive. To resolve this issue, the low-rank matrix factorization methods [24]–[26] have been proposed. They depict the low-rank structure of a matrix by factorizing it into the product of two smaller matrices and avoid computing SVD, thus are more efficient. Besides, by directly restricting the rank of the estimation, the low-rank factorization methods can also well recover the low-rank data and achieve state-of-the-art completion results in image/video inpainting [25], [27], [28], and collaborative filtering [26], [29].

It seems natural to directly extend matrix completion methods to the tensor completion problem. However, this is not suitable, as the numerical algebra of tensors is fraught with hardness results [16]. For instance, the CANDECOMP/PARAFAC (CP) rank [30] of a tensor $\boldsymbol{\mathcal{C}}\in\mathbb{R}^{n_1\times n_2\times\cdots\times n_k}$ which is defined as the minimum number of rank one decomposition, *i.e.*

$$\text{rank}_{\text{cp}}(\boldsymbol{\mathcal{C}})=\min\{r \mid \boldsymbol{\mathcal{C}}=\sum_{i=1}^{r}\boldsymbol{a}_1^{(i)}\otimes\boldsymbol{a}_2^{(i)}\otimes\cdots\otimes\boldsymbol{a}_k^{(i)}\}, \tag{2}$$

P. Zhou, Z. Lin, and C. Zhang are with Key Lab. of Machine Perception (MoE), School of EECS, Peking University, P. R. China. Z. Lin and C. Zhang are also with Cooperative Medianet Innovation Center, Shanghai, China. P. Zhou is now with Department of Electrical & Computer Engineering, National University of Singapore, Singapore. (e-mails: pzhou@pku.edu.cn, zlin@pku.edu.cn, and chzhang@cis.pku.edu.cn).

C. Lu is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore (e-mail: canyilu@gmail.com).

where the symbol $\otimes$ denotes the outer product and $\boldsymbol{a}_j^{(i)} \in \mathbb{R}^{n_j}$ $(\forall(i,j))$ is a vector, is generally NP-hard to compute [16], [31], [32]. To resolve this issue, other tensor rank definitions [14], [33] have been proposed, leading to different low-rank tensor completion methods [1], [5], [13], [34].

Matricization, *a.k.a.* unfolding or flattening, is the commonly used approach to solve the tensor completion problem [13]. It unfolds the tensor data into matrices and applies matrix completion methods, *e.g.* the matrix nuclear norm based methods [20]–[22] or the matrix factorization methods [24]–[26], to recover the low-rank tensor. The Tucker rank [14] defined on the unfolded matrices to depict the rank of a tensor is the basic of matricization methods, formulated as

$$\text{rank}_{\text{tc}}(\boldsymbol{\mathcal{C}}) = \left(\text{rank}(\boldsymbol{C}_{(1)}), \cdots, \text{rank}(\boldsymbol{C}_{(i)}), \cdots, \text{rank}(\boldsymbol{C}_{(k)})\right), \tag{3}$$

where $\boldsymbol{\mathcal{C}}$ is a k-way tensor and $\boldsymbol{C}_{(i)}$ is its mode-i matricization. Based on Tucker rank, Kasai *et al.* [5] further considered data structure and proposed a Riemannian manifold based tensor completion method (RMTC). Furthermore, as the minimizing rank function is complex due to its combinational nature [25], in [2], [7], [35], the authors used the sum of the nuclear norm (SNN) $\sum_{i=1}^{k} \|\boldsymbol{C}_{(i)}\|_*$ to approximate the rank of the tensor and the completion model is formulated as

$$\min_{\boldsymbol{\mathcal{C}}} \sum_{i=1}^{k} \|\boldsymbol{C}_{(i)}\|_*, \quad \text{s.t. } P_\Omega(\boldsymbol{\mathcal{C}} - \boldsymbol{\mathcal{M}}) = \boldsymbol{0}. \tag{4}$$

But Romera-Paredes *et al.* [4] proved that SNN is not a tight convex relaxation of $\sum_{i=1}^{k} \text{rank}(\boldsymbol{C}_{(i)})$ and presented an alternative convex relaxation, *i.e.* TenALS which is tighter than SNN. Moreover, the nuclear norm minimization problem (4) is generally solved iteratively in which SVD is involved at each iteration. So these nuclear norm minimization methods suffer from high computational cost of multiple SVDs. To efficiently handle large scale data, Xu *et al.* [3] utilized the matrix factorization method to preserve the low-rank structure of the unfolded matrices, *i.e.* factorizing each mode matricization $\boldsymbol{C}_{(i)}$ into the product of two smaller matrices $\boldsymbol{X}^i$ and $\boldsymbol{Y}^i$:

$$\min_{\{\boldsymbol{X}_i\},\{\boldsymbol{Y}_i\},\boldsymbol{\mathcal{C}}} \sum_{i=1}^{k} \boldsymbol{\alpha}_i \|\boldsymbol{X}^i\boldsymbol{Y}^i - \boldsymbol{C}_{(i)}\|_*, \text{ s.t. } P_\Omega(\boldsymbol{\mathcal{C}} - \boldsymbol{\mathcal{M}}) = \boldsymbol{0}, \tag{5}$$

where $\boldsymbol{\alpha}_i$ is a positive weight parameter which satisfies $\sum_{i=1}^{k} \boldsymbol{\alpha}_i = 1$. This approach has been widely applied to various applications [10], [11]. However, as pointed out by [32], [33], [36], [37], directly unfolding a tensor would destroy the original multi-way structure of the data, leading to vital information loss and degraded performance. Besides, matricization methods unfold a k-way tensor into k matrices among which each has the same number of entries as the tensor, and then recover each unfolded matrix and combine them into a tensor at each iteration, thus may be inefficient, as the number of entries of a tensor is usually very large.

Recently, based on the tensor-tensor product (t-product) and tensor singular value decomposition (t-SVD) [36]–[38], Kilmer *et al.* [33] proposed the tensor multi-rank and tubal rank definitions and Semerci *et al.* [34] developed a new tensor nuclear norm (TNN). Subsequently, Zhang *et al.* applied TNN to tensor completion [1], with state-of-the-art video inpainting results achieved, and further analyzed the exact completion conditions of the proposed model in [39]. Since t-SVD is based on an operator theoretic interpretation of 3-way tensors as linear operators on the space of oriented matrices, the tensor multi-rank and tubal rank can well characterize the inherent low-rank structure of a tensor while avoiding the loss of information inherent in matricization of the tensor [32], [33], [36], [37]. But TNN still requires computing t-SVD, which is very time consuming when the data scale is large.

In this paper, we propose a novel low-rank tensor factorization method to solve the 3-way tensor completion problem. Similar to TNN [1], our method is also based upon the tensor multi-rank and tubal rank definitions. But instead of using the tensor nuclear norm, we factorize the low-tubal-rank tensor $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ into the product of two tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times r \times n_3}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{r \times n_2 \times n_3}$ of smaller sizes, where r is the tubal rank of $\boldsymbol{\mathcal{C}}$ and is usually much smaller than $\min(n_1, n_2)$. See the definition of tensor tubal rank in Definition 6. This factorization has performance guarantee, since by Lemma 2, a low-tubal-rank tensor can be factorized into the product of two tensors of much smaller sizes. In this way, we can characterize the low-rank property of the tensor $\boldsymbol{\mathcal{C}}$ and recover it. Since our low-rank factorization method avoids computing t-SVD, the computational cost at each iteration is $\mathcal{O}(r(n_1+n_2)n_3 \log n_3 + rn_1n_2n_3)$ and much more efficient than TNN [1] whose computational complexity is $\mathcal{O}(n_1n_2n_3 \log n_3 + n_1n_2n_3 \min(n_1, n_2))$. Compared with matricization based tensor completion methods, our method is based upon the tensor multi-rank and tubal rank and avoids losing the low-rank structure information. Experimental results also confirm this point. We also propose a rank-decreasing method to estimate the true tensor rank of a tensor whose effectiveness is verified by experimental results (see Fig. 3). At each iteration, once a dramatic change in the estimated rank of the variable $\boldsymbol{\mathcal{X}}$ is detected by the QR decomposition, we use PCA [40] to adjust the sizes of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ so that they can well reconstruct the original tensor. Finally, we present an efficient optimization method to solve our model and further prove its convergence. When we develop this method, based on tensor tubal rank and tensor nuclear norm, Liu *et al.* [40] also proposed a tensor factorization method for tensor completion, which is called "Tubal-Alt-Min". But these two concurrent methods essentially differ from each other due to the very different optimization algorithms and rank estimation strategies. See Sec. III-E for more detailed discussions on these differences. In summary, our main contributions include:

(1) We propose an efficient low-rank tensor factorization method for tensor completion problem. Our method characterizes the low-tubal-rank structure of a tensor by factorizing it into the product of two tensors of smaller sizes and only needs to update two smaller tensors at each iteration. In each iteration, the complexity of our method is $\mathcal{O}(r(n_1+n_2)n_3 \log n_3 + rn_1n_2n_3)$. Such a computational cost is much lower than TNN [1] which requires computing tensor SVD with the complexity $\mathcal{O}(n_1n_2n_3(\min(n_1, n_2) + \log n_3))$. It is obvious that

our method is much more efficient than TNN.
(2) We propose an adaptive method to estimate the tensor tubal rank in each iteration. By detecting the dramatic change of the rank of one factorization variable $\mathcal{X}$, we use PCA to decrease the sizes of the two factorization variables $\mathcal{X}$ and $\mathcal{Y}$ so that they can well reconstruct the original tensor. The validity of this rank estimation method is verified by the experimental results.
(3) We prove that the proposed alternating minimization algorithm can converge to a Karush-Kuhn-Tucker point.

Experimental results on synthetic data completion and image/video inpainting tasks verify the advantages of our method.

II. Notations and Preliminaries

In this section we first summarize some main notations and then introduce some definitions and a lemma used later.

A. Notations

We use boldface Calligraphy letters, *e.g.* $\mathcal{A}$, and boldface capital letters, *e.g.* $\boldsymbol{A}$, to denote tensors and matrices, respectively. We use boldface lowercase letters, *e.g.* $\boldsymbol{a}$, to represent vectors, and lowercase letters, *e.g.* a, to denote scalars. For a tensor $\mathcal{A}$, we use the Matlab notation $\mathcal{A}(i,:,:)$, $\mathcal{A}(:,i,:)$ and $\mathcal{A}(:,:,i)$ to denote its i-th horizontal, lateral and frontal slice, respectively. For brevity, let $\boldsymbol{A}^{(i)} = \mathcal{A}(:,:,i)$. $\mathcal{A}_{ijk}$ denote the (i,j,k)-th entry of $\mathcal{A}$. The Frobenius norm is defined as $\|\mathcal{A}\|_F = \sqrt{\sum_{ijk}|\mathcal{A}_{ijk}|^2}$. $\|\mathcal{A}\|_\infty$ represents the maximum absolute value in $\mathcal{A}$. $\boldsymbol{A}^*$ and $\boldsymbol{A}^\dagger$ represent the conjugate transpose and the pseudo-inverse of $\boldsymbol{A}$, respectively. $\boldsymbol{I}_n$ represents the identity matrix of size $n \times n$.

Next, we introduce the Discrete Fourier Transformation (DFT), which plays a core role in tensor-tensor product introduced later. Let $\bar{\mathcal{A}} \in \mathbb{C}^{n_1\times n_2\times n_3}$ represent the result of Discrete Fourier transformation (DFT) of $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ along the 3rd dimension. Define the *DFT matrix*:

$$\boldsymbol{F}_{n_3} = [\boldsymbol{f}_1, \cdots, \boldsymbol{f}_i, \cdots, \boldsymbol{f}_{n_3}] \in \mathbb{R}^{n_3\times n_3},$$

where $\boldsymbol{f}_i = [\omega^{0\times(i-1)}; \omega^{1\times(i-1)}; \cdots; \omega^{(n_3-1)\times(i-1)}] \in \mathbb{R}^{n_3}$ in which $\omega = e^{-(2\pi\mathfrak{b}/n_3)}$ and $\mathfrak{b} = \sqrt{-1}$. Then we have $\bar{\mathcal{A}}(i,j,:) = \boldsymbol{F}_{n_3}\mathcal{A}(i,j,:)$. Indeed, we can compute $\bar{\mathcal{A}}$ directly by the Matlab command $\bar{\mathcal{A}} = \texttt{fft}(\mathcal{A}, [], 3)$ and use the inverse DFT to obtain $\mathcal{A} = \texttt{ifft}(\bar{\mathcal{A}}, [], 3)$. We further define $\bar{\boldsymbol{A}} \in \mathbb{C}^{n_1n_3\times n_2n_3}$ as

$$\bar{\boldsymbol{A}} = \texttt{bdiag}(\bar{\mathcal{A}}) = \begin{bmatrix} \bar{\boldsymbol{A}}^{(1)} & & & \\ & \bar{\boldsymbol{A}}^{(2)} & & \\ & & \ddots & \\ & & & \bar{\boldsymbol{A}}^{(n_3)} \end{bmatrix}, \quad (6)$$

where $\texttt{bdiag}(\cdot)$ is an operator which maps the tensor $\bar{\mathcal{A}}$ to the block diagonal matrix $\bar{\boldsymbol{A}}$. We further define the block circulant matrix $\texttt{bcirc}(\mathcal{A}) \in \mathbb{R}^{n_1n_3\times n_2n_3}$ of $\mathcal{A}$ as

$$\texttt{bcirc}(\mathcal{A}) = \begin{bmatrix} \boldsymbol{A}^{(1)} & \boldsymbol{A}^{(n_3)} & \cdots & \boldsymbol{A}^{(2)} \\ \boldsymbol{A}^{(2)} & \boldsymbol{A}^{(1)} & \cdots & \boldsymbol{A}^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{(n_3)} & \boldsymbol{A}^{(n_3-1)} & \cdots & \boldsymbol{A}^{(1)} \end{bmatrix}. \quad (7)$$

The definitions of $\bar{\boldsymbol{A}}$ and $\texttt{bcirc}(\mathcal{A})$ are the basis of tensor rank and nuclear norm that will be introduced subsequently.

B. Basic Knowledge

We first introduce t-product between two 3-way tensors.

Definition 1. ***(T-product)*** *[36] The t-product between $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2\times n_4\times n_3}$ is defined as $\mathcal{A} * \mathcal{B} = \texttt{fold}(\texttt{bcirc}(\mathcal{A}) \cdot \texttt{unfold}(\mathcal{B})) \in \mathbb{R}^{n_1\times n_4\times n_3}$, where $\texttt{unfold}(\mathcal{A}) = [\boldsymbol{A}^{(1)}; \boldsymbol{A}^{(2)}; \cdots; \boldsymbol{A}^{(n_3)}] \in \mathbb{R}^{n_1n_3\times n_2}$ and its inverse operator $\texttt{fold}$ is defined as $\texttt{fold}(\texttt{unfold}(\mathcal{A})) = \mathcal{A}$.*

Indeed, t-product is equivalent to the matrix multiplication in the Fourier domain, *i.e.* $\mathcal{F} = \mathcal{A} * \mathcal{B}$ and $\bar{\boldsymbol{F}} = \bar{\boldsymbol{A}}\bar{\boldsymbol{B}}$ are equivalent [36] (also see Lemma 1). Based on the definition of t-product, we introduce the definition of t-SVD. Before that, we also need some other concepts.

Definition 2. ***(F-diagonal tensor)*** *[36] A tensor is called f-diagonal if each of its frontal slices is a diagonal matrix.*

Definition 3. ***(Conjugate transpose)*** *[36] The conjugate transpose of a tensor $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ is the tensor $\mathcal{A}^* \in \mathbb{R}^{n_2\times n_1\times n_3}$ obtained by conjugate transposing each of the frontal slices and then reversing the order of transposed frontal slices 2 through n_3.*

Definition 4. ***(Identity tensor)*** *[36] The identity tensor $\mathcal{I} \in \mathbb{R}^{n\times n\times n_3}$ is the tensor whose first frontal slice is the $n \times n$ identity matrix, and other frontal slices are all zeros.*

Definition 5. ***(Orthogonal tensor)*** *[36] A tensor $\mathcal{P} \in \mathbb{R}^{n\times n\times n_3}$ is orthogonal if it satisfies*

$$\mathcal{P}^* * \mathcal{P} = \mathcal{P} * \mathcal{P}^* = \mathcal{I}. \quad (8)$$

Now we introduce the recently proposed t-SVD, a new tensor decomposition framework.

Theorem 1. ***(T-SVD)*** *[36] Assume that $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ is a 3-way tensor. Then it can be factored as*

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^*, \quad (9)$$

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

T-SVD employs similar properties to matrix SVD, such as the orthogonal property of $\mathcal{U}$ and $\mathcal{V}$, and the diagonal structure of $\mathcal{S}$. Indeed, when $n_3 = 1$, t-SVD would degrade into matrix SVD. Now we introduce tensor multi-rank and tubal rank.

Definition 6. ***(Tensor multi-rank and tubal rank)*** *[33] For any $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$, its **multi-rank** $rank_m(\mathcal{A})$ is a vector defined as $\boldsymbol{r} = (rank(\bar{\boldsymbol{A}}^{(1)}); \cdots; \bar{\boldsymbol{A}}^{(n_3)}))$. The tensor **tubal rank** $rank_t(\mathcal{A})$ is defined as the number of nonzero singular tubes of $\mathcal{S}$, i.e.,*

$$rank_t(\mathcal{A}) = \#\{i : \mathcal{S}(i,i,:) \neq 0\} = \max(\boldsymbol{r}_1, \cdots, \boldsymbol{r}_{n_3}),$$

where $\mathcal{S}$ is from the t-SVD (see below) of $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^*$.

Then we can give the definition of tensor nuclear norm, which is the convex envelop of the tensor average rank. With this property, we use tensor nuclear norm to depict the low-rank structure of a tensor.

Definition 7. ***(Tensor nuclear norm)*** *[1], [6] The tensor nuclear norm $\|\mathcal{A}\|_*$ of a tensor $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ is defined*

as the sum of the singular values of all frontal slices of $\bar{\mathcal{A}}$, i.e., $\|\mathcal{A}\|_* = \frac{1}{n_3}\sum_{i=1}^{n_3}\|\bar{A}^{(i)}\|_*$.

Finally, we introduce a lemma which will be used for model simplification and theoretical analysis.

Lemma 1. *[36] Suppose that $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$, $\mathcal{B} \in \mathbb{R}^{n_2\times n_4\times n_3}$ are two arbitrary tensors. Let $\mathcal{F} = \mathcal{A} * \mathcal{B}$. Then, the following properties hold.*

(1) $\|\mathcal{A}\|_F^2 = \frac{1}{n_3}\|\bar{A}\|_F^2$;

(2) $\mathcal{F} = \mathcal{A} * \mathcal{B}$ *and* $\bar{F} = \bar{A}\bar{B}$ *are equivalent to each other.*

Note that by utilizing Lemma 1, we can compute t-SVD in an efficient way. We know that (9) is equivalent to $\bar{A} = \bar{U}\bar{S}\bar{V}^*$, where $\bar{A}^{(i)} = \bar{U}^{(i)}\bar{S}^{(i)}(\bar{V}^{(i)})^*$ is the SVD of $\bar{A}^{(i)}$, in which $\bar{U}^{(i)}$, $\bar{S}^{(i)}$ and $\bar{V}^{(i)}$ are the i-th frontal slices of $\bar{\mathcal{U}}$, $\bar{\mathcal{S}}$ and $\bar{\mathcal{V}}$, respectively. Thus, we can compute the SVD of $\bar{A}^{(i)}$ $(i = 1, \cdots, n_3)$ to obtain the t-SVD of $\mathcal{A}$. However, computing t-SVD is still very computationally expensive.

III. Tensor Factorization for Low-Rank Tensor Completion

Here we first present the details of our low-rank tensor factorization method, and then introduce its optimization and analyze its convergence. After that, a rank-decreasing method is proposed to estimate the true rank of a tensor. Finally, we compare our method with prior work.

A. Formulation of Tensor Factorization

Tensor completion is to fill in the missing values of a tensor $\mathcal{M} \in \mathbb{R}^{n_1\times n_2\times n_3}$ under a given subset Ω of its entries $\{\mathcal{M}_{ijk} \mid (i,j,k) \in \Omega\}$. Since tensor data of high dimensional are usually underlying low-rank [13], the formulation of tensor completion can be written as

$$\min_{\mathcal{C}} \operatorname{rank_t}(\mathcal{C}), \quad \text{s.t. } P_\Omega(\mathcal{C} - \mathcal{M}) = \mathbf{0}, \tag{10}$$

where $\operatorname{rank_t}(\mathcal{C})$ denotes the tubal rank of $\mathcal{C}$ and P_Ω is the linear operator that extracts entries in Ω and fills the entries not in Ω with zeros.

Note that there are several definitions of tensor rank. The tensor tubal rank in Definition 6 is one of the commonly used tensor rank definitions. Since the tensor multi-rank and tubal rank are based on t-SVD, which is an operator theoretic interpretation of tensors as linear operators in the space of oriented matrices, applying the tensor multi-rank or tubal rank to depict the rank of a tensor avoids destroying the low-rank structure of a tensor [1], [33]. But since minimizing $\operatorname{rank_t}(\mathcal{C})$ is complex, it seems natural to replace $\operatorname{rank_t}(\mathcal{X})$ of the sum as the rank of all the frontal slices of $\bar{\mathcal{C}}$, *i.e.*, $\sum_{i=1}^{n_3}\operatorname{rank}(\bar{C}^{(i)})$. To solve it more easily, Zhang *et al.* [1] used the tensor nuclear norm $\sum_{i=1}^{n_3}\|\bar{C}^{(i)}\|_*$ defined in Definition 7 to approximate the rank function, with state-of-the-art completion results achieved [1]. However, as we mentioned, the nuclear norm problem has to be solved iteratively and involves SVD at each iteration. What is worse, tensor data are usually multi-way and high dimensional and hence are very large scale. Therefore, this approach for nuclear norm minimization problem suffers from high computation cost of multiple SVDs. Motivated by this work and matrix completion research that low-rank matrix factorization method can handle large scale data more efficiently [24]–[26], we utilize the low-rank tensor factorization method to recover the low-rank tensor and aim at a faster method for tensor completion. We first present two properties of the tensor tubal rank in Lemma 2.

Lemma 2. *Suppose that $\mathcal{F} \in \mathbb{R}^{n_1\times n_2\times n_3}$, $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$, and $\mathcal{B} \in \mathbb{R}^{n_2\times n_4\times n_3}$ are three arbitrary tensors. Then, the following properties hold.*

(1) *If $\operatorname{rank_t}(\mathcal{F}) = \hat{k}$, then $\mathcal{F}$ can be written into a tensor product form $\mathcal{F} = \mathcal{G} * \mathcal{H}$, where $\mathcal{G} \in \mathbb{R}^{n_1\times\hat{k}\times n_3}$ and $\mathcal{H} \in \mathbb{R}^{\hat{k}\times n_2\times n_3}$ are two tensors of smaller sizes and they meet $\operatorname{rank_t}(\mathcal{G}) = \operatorname{rank_t}(\mathcal{H}) = \hat{k}$;*

(2) *$\operatorname{rank_t}(\mathcal{A} * \mathcal{B}) \le \min(\operatorname{rank_t}(\mathcal{A}), \operatorname{rank_t}(\mathcal{B}))$.*

The proof of Lemma 2 can be found in the supplementary material. As we can see, based on t-product and t-SVD, the tensor tubal rank enjoys some similar interesting properties as the matrix rank. Actually, when the third dimension is 1, the properties in Lemma 2 can also be applicable to a matrix. Thus we can adopt a low-rank factorization strategy, which is similar to the matrix factorization method, to deal with the large scale tensor completion problem more efficiently. By utilizing Lemma 2, we characterize the low-rank property of a low-rank tensor by factorizing it into the product of two tensors with smaller sizes. That is, we can factorize any tensor $\mathcal{M}$ of a tensor tubal rank up to $\hat{r}$ into the tensor product form $\mathcal{M} = \mathcal{X} * \mathcal{Y}$ of two tensors $\mathcal{X} \in \mathbb{R}^{n_1\times\hat{r}\times n_3}$ and $\mathcal{Y} \in \mathbb{R}^{\hat{r}\times n_2\times n_3}$, which meet $\operatorname{rank_t}(\mathcal{X}) = \operatorname{rank_t}(\mathcal{Y}) = \hat{r}$. Thus, we can control the rank of the tensor $\mathcal{M}$ by controlling the sizes of two tensors $\mathcal{X}$ and $\mathcal{Y}$. Note that we can always adjust the shape of $\mathcal{M}$ so that it satisfies $n_1 = \max(n_1, n_2, n_3)$ or $n_2 = \max(n_1, n_2, n_3)$. For example, if a tensor $\mathcal{F} \in \mathbb{R}^{n_1\times n_2\times n_3}$ meets $n_3 = \max(n_1, n_2, n_3)$, we can reshape it by transposing each horizontal slice to obtain a new tensor $\mathcal{F}' \in \mathbb{R}^{n_1\times n_3\times n_2}$, and then deal with $\mathcal{F}'$. Besides, the tubal rank $\hat{r}$ is typically much smaller than $\min(n_1, n_2)$. Hence, updating and storing the two smaller tensors can be much more efficient. At the same time, following most matrix or tensor completion methods [3], [10], [11], [25], [26], we assume that the noise in observation data is Gaussian observation noise and we use the Frobenius norm to characterize it. Accordingly, our tensor factorization formulation can be written as follows:

$$\min_{\mathcal{X},\mathcal{Y}} \frac{1}{2}\|P_\Omega(\mathcal{X} * \mathcal{Y} - \mathcal{M})\|_F^2. \tag{11}$$

To solve problem (11) more conveniently, we introduce one auxiliary variable $\mathcal{C}$ and rewrite problem (11) as follows:

$$\min_{\mathcal{X},\mathcal{Y},\mathcal{C}} \frac{1}{2}\|\mathcal{X} * \mathcal{Y} - \mathcal{C}\|_F^2, \quad \text{s.t. } P_\Omega(\mathcal{C} - \mathcal{M}) = \mathbf{0}. \tag{12}$$

In this way, we can observe that in problem (12) only the variable $\mathcal{C}$ is involved in the linear operator P_Ω. So when updating $\mathcal{C}$, we only need to project $\mathcal{C}$ onto Ω which is easy. In contrast, in problem (11), P_Ω has constraints on the product of $\mathcal{X}$ and $\mathcal{Y}$ and hence updating $\mathcal{X}$ or $\mathcal{Y}$ is more challenging.

Now, we discuss problem (12) in detail. Assume that $\operatorname{rank}_m(\mathcal{C}) = \boldsymbol{r}$ and $\operatorname{rank}_t(\mathcal{C}) = \hat{r}$, where $\boldsymbol{r}_i = \operatorname{rank}(\bar{C}^{(i)})$

Algorithm 1 Tensor Completion by Tensor Factorization (TCTF)

Input: The tensor data $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, the observed set Ω, the initialized rank $\boldsymbol{r}^0 \in \mathbb{R}^{n_3}$, and $\varepsilon = 1e-5$.
Initialize: $\hat{\boldsymbol{X}}^0$, $\hat{\boldsymbol{Y}}^0$.
While not converge do
1. Fix $\hat{\boldsymbol{X}}^k$ and $\hat{\boldsymbol{Y}}^k$ to update $\boldsymbol{\mathcal{C}}^{k+1}$ by (14), and then compute $\bar{\boldsymbol{C}}$.
2. Fix $\hat{\boldsymbol{Y}}^k$ and $\boldsymbol{\mathcal{C}}^{k+1}$ to update $\hat{\boldsymbol{X}}^{k+1}$ by (15).
3. Fix $\hat{\boldsymbol{X}}^{k+1}$ and $\boldsymbol{\mathcal{C}}^{k+1}$ to update $\hat{\boldsymbol{Y}}^{k+1}$ by (16).
4. Adopt the rank decreasing scheme in Section III-D to adjust $\boldsymbol{r}^k$ and the sizes of $\hat{\boldsymbol{X}}^{k+1}$ and $\hat{\boldsymbol{Y}}^{k+1}$.
5. Check the convergence condition: $\| \hat{\boldsymbol{X}}^{k+1} - \hat{\boldsymbol{X}}^k \|_\infty < \varepsilon$, $\| \hat{\boldsymbol{Y}}^{k+1} - \hat{\boldsymbol{Y}}^k \|_\infty < \varepsilon$ and $\| \boldsymbol{\mathcal{C}}^{k+1} - \boldsymbol{\mathcal{C}}^k \|_\infty < \varepsilon$.
6. $k \leftarrow k+1$.
end while
Output: $\hat{\boldsymbol{X}}^{k+1}$, $\hat{\boldsymbol{Y}}^{k+1}$, and $\boldsymbol{\mathcal{C}}^{k+1}$.

$(i = 1, \cdots, n_3)$ and $\hat{r} = \max(\boldsymbol{r}_1, \cdots, \boldsymbol{r}_{n_3})$. Thus, $\bar{\boldsymbol{C}}^{(i)} \in \mathbb{C}^{n_1 \times n_2}$ can be factorized into the product of two matrices $\hat{\boldsymbol{X}}^{(i)}$ and $\hat{\boldsymbol{Y}}^{(i)}$ of smaller sizes, where $\hat{\boldsymbol{X}}^{(i)} \in \mathbb{C}^{n_1 \times \boldsymbol{r}_i}$ and $\hat{\boldsymbol{Y}}^{(i)} \in \mathbb{C}^{\boldsymbol{r}_i \times n_2}$ are the i-th block diagonal matrices of $\hat{\boldsymbol{X}} \in \mathbb{C}^{n_1 n_3 \times (\sum_{i=1}^{n_3} \boldsymbol{r}_i)}$ and $\hat{\boldsymbol{Y}} \in \mathbb{C}^{(\sum_{i=1}^{n_3} \boldsymbol{r}_i) \times n_2 n_3}$, respectively. Let $\bar{\boldsymbol{X}}^{(i)} = [\hat{\boldsymbol{X}}^{(i)}, \boldsymbol{0}] \in \mathbb{C}^{n_1 \times \hat{r}}$ and $\bar{\boldsymbol{Y}}^{(i)} = [\hat{\boldsymbol{Y}}^{(i)}; \boldsymbol{0}] \in \mathbb{C}^{\hat{r} \times n_2}$. Then, we have $\hat{\boldsymbol{X}}\hat{\boldsymbol{Y}} = \bar{\boldsymbol{X}}\bar{\boldsymbol{Y}}$. From Lemma 1, we know that $\bar{\boldsymbol{X}}\bar{\boldsymbol{Y}}$ and $\boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}}$ are equivalent and $\|\boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{C}}\|_F^2 = \frac{1}{n_3}\|\bar{\boldsymbol{X}}\bar{\boldsymbol{Y}} - \bar{\boldsymbol{C}}\|_F^2$. Thus, problem (12) is equivalent to

$$\min_{\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}, \boldsymbol{\mathcal{C}}} \frac{1}{2n_3} \sum_{i=1}^{n_3} \|\hat{\boldsymbol{X}}^{(i)}\hat{\boldsymbol{Y}}^{(i)} - \bar{\boldsymbol{C}}^{(i)}\|_F^2, \text{ s.t. } P_\Omega(\boldsymbol{\mathcal{C}} - \boldsymbol{\mathcal{M}}) = \boldsymbol{0}. \tag{13}$$

B. Optimization of Tensor Factorization

Problem (13) is not jointly convex with respect to $(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}, \boldsymbol{\mathcal{C}})$. Empirically, we can develop an iterative minimization method to update each variable alternately. Similar to [3], [25] which respectively use the alternating minimization algorithm to optimize matrix and tensor completion problem, here we also adopt it as our optimization method. We can update the three variables in turn as follows:

$$\begin{aligned} \boldsymbol{\mathcal{C}} &= \underset{P_\Omega(\boldsymbol{\mathcal{C}} - \boldsymbol{\mathcal{M}}) = \boldsymbol{0}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{C}}\|_F^2 \\ &= \boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}} + P_\Omega(\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}}). \end{aligned} \tag{14}$$

Then, we can obtain $\bar{\boldsymbol{C}}$ via $\boldsymbol{\mathcal{C}}$ and then update $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$ as follows:

$$\begin{aligned} \hat{\boldsymbol{X}}^{(i)} &= \underset{\hat{\boldsymbol{X}}^{(i)}}{\operatorname{argmin}} \frac{1}{2n_3}\|\hat{\boldsymbol{X}}^{(i)}\hat{\boldsymbol{Y}}^{(i)} - \bar{\boldsymbol{C}}^{(i)}\|_F^2 \\ &= \bar{\boldsymbol{C}}^{(i)}(\hat{\boldsymbol{Y}}^{(i)})^* \left(\hat{\boldsymbol{Y}}^{(i)}(\hat{\boldsymbol{Y}}^{(i)})^*\right)^\dagger, i = 1, \cdots, n_3, \end{aligned} \tag{15}$$

$$\begin{aligned} \hat{\boldsymbol{Y}}^{(i)} &= \underset{\hat{\boldsymbol{Y}}^{(i)}}{\operatorname{argmin}} \frac{1}{2n_3}\|\hat{\boldsymbol{X}}^{(i)}\hat{\boldsymbol{Y}}^{(i)} - \bar{\boldsymbol{C}}^{(i)}\|_F^2 \\ &= \left((\hat{\boldsymbol{X}}^{(i)})^*\hat{\boldsymbol{X}}^{(i)}\right)^\dagger (\hat{\boldsymbol{X}}^{(i)})^*\bar{\boldsymbol{C}}^{(i)}, i = 1, \cdots, n_3. \end{aligned} \tag{16}$$

After updating $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$, we can compute $\boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}}$ in an efficient way. Assume that $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}}$. Then we first compute $\bar{\boldsymbol{G}}^{(i)} = \hat{\boldsymbol{X}}^{(i)}\hat{\boldsymbol{Y}}^{(i)}$ $(i = 1, \cdots, n_3)$ and obtain $\boldsymbol{\mathcal{G}} = \texttt{ifft}(\bar{\boldsymbol{\mathcal{G}}}, [], 3)$. This is more efficient than directly computing $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}}$. Furthermore, all $\hat{\boldsymbol{X}}^{(i)}$ can be updated parallelly and all $\hat{\boldsymbol{Y}}^{(i)}$ can also be parallelly computed. But in this paper, we adopt the serial updating scheme in our code, since our serial updating scheme is also very fast. Please refer to Section IV. The detailed optimization procedure is presented in Algorithm 1.

Complexity analysis: Here we analyze the computational complexity of Algorithm 1. At each iteration, when updating $\boldsymbol{\mathcal{C}}$ by Eqn. (14), the computational cost for conducting the (inverse) DFT and matrix product is $\mathcal{O}(\hat{r}(n_1+n_2)n_3 \log n_3 + \hat{r}n_1n_2n_3)$ where $\hat{r}$ is the estimated tubal rank of $\boldsymbol{\mathcal{C}}$. Then the cost of updating $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ respectively by Eqn. (15) and (16) is $\mathcal{O}(\hat{r}(n_1+n_2)n_3 \log n_3 + \hat{r}n_1n_2n_3)$. In step 4, we use QR decomposition to estimate the target rank whose cost is also $\mathcal{O}(\hat{r}(n_1+n_2)n_3 \log n_3 + \hat{r}n_1n_2n_3)$. So the total cost at each iteration is $\mathcal{O}(\hat{r}(n_1+n_2)n_3 \log n_3 + \hat{r}n_1n_2n_3)$. By comparison, for matricization methods, the costs of SiLRTC [2] and TMac [3], which respectively employ nuclear norm minimization and matrix factorization strategy, are respectively $\mathcal{O}((n_1+n_2+n_3)n_1n_2n_3)$ and $\mathcal{O}((r_1+r_2+r_3)n_1n_2n_3)$ at each iteration, where r_1, r_2 and r_3 respectively denote the estimated rank of the three unfolded matrices. RMTC [5] further considers the Riemannian manifold and has the cost $\mathcal{O}(r_1r_2r_3n_1n_2n_3)$. Based on matricization, TenALS [4] is an alternative convex relaxation of Tucker rank and is tighter than TNN. But at each iteration, TenALS has a subproblem to solve and its cost at each outer iteration is $\mathcal{O}\left((n_1+n_2+n_3)n_1n_2n_3 + t(n_1^2+n_2^2+n_3^2)\right)$, where t is the inner loop iteration. Thus, at each iteration, the computational cost of our algorithm is much lower than those of matricization methods. Now we compare our method with TNN [1] and Tubal-Alt-Min [41] whose costs at each iteration are $\mathcal{O}(n_1n_2n_3 \log n_3 + n_1n_2n_3 \min(n_1, n_2))$ and $\mathcal{O}\left(\hat{r}(n_1+n_2)n_3 \log n_3 + \hat{r}n_1n_2n_3^2 \log^2(\max(n_1, n_2))\right)$, respectively. We can observe that our method is more efficient than TNN and Tubal-Alt-Min. Also the tensor tubal rank based methods, *e.g.* our method and TNN, usually have lower computational complexity than matricization methods, such as SiLRTC and TMac. Actually, we also find that the factorization based methods, *i.e.* our method and TMac, also outperform their corresponding nuclear norm minimization counterparts, *i.e.* TNN and SiLRTC, on computational efficiency.

C. Convergence Analysis

In this subsection, before proving the convergence of the proposed algorithm, we first present the first order optimality conditions for (13). By introducing a Lagrangian multiplier $\boldsymbol{\mathcal{Q}}$ for the constraint, we can write the Lagrangian function of problem (13):

$$\mathcal{L}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{Q}}) = \frac{1}{2n_3}\|\hat{\boldsymbol{X}}\hat{\boldsymbol{Y}} - \bar{\boldsymbol{C}}\|_F^2 + \langle \boldsymbol{\mathcal{Q}}, P_\Omega(\boldsymbol{\mathcal{C}} - \boldsymbol{\mathcal{M}}) \rangle. \tag{17}$$

Let $\nabla_{\hat{\boldsymbol{X}}}\mathcal{L}=\mathbf{0},\nabla_{\hat{\boldsymbol{Y}}}\mathcal{L}=\mathbf{0},\nabla_{\boldsymbol{\mathcal{C}}}\mathcal{L}=\mathbf{0}$ and $\nabla_{\boldsymbol{\mathcal{Q}}}\mathcal{L}=\mathbf{0}$. We can obtain the following KKT conditions:

$$\begin{cases}(\hat{\boldsymbol{X}}\hat{\boldsymbol{Y}}-\bar{\boldsymbol{C}})\hat{\boldsymbol{Y}}^*=\mathbf{0},\\ \hat{\boldsymbol{X}}^*(\hat{\boldsymbol{X}}\hat{\boldsymbol{Y}}-\bar{\boldsymbol{C}})=\mathbf{0},\\ P_{\Omega^c}(\boldsymbol{\mathcal{C}}-\boldsymbol{\mathcal{X}}*\boldsymbol{\mathcal{Y}})=\mathbf{0},\\ P_{\Omega}(\boldsymbol{\mathcal{C}}-\boldsymbol{\mathcal{M}})=\mathbf{0},\\ P_{\Omega}(\boldsymbol{\mathcal{C}}-\boldsymbol{\mathcal{X}}*\boldsymbol{\mathcal{Y}})+\boldsymbol{\mathcal{Q}}=\mathbf{0},\end{cases}\tag{18}$$

where Ω^c is the complement of Ω.

Now we establish Theorem 2 to prove that Algorithm 1 decreases the objective function value monotonically and it can converge to a KKT point of our optimization problem.

Theorem 2. *Assume that* $f(\hat{\boldsymbol{X}},\hat{\boldsymbol{Y}},\boldsymbol{\mathcal{C}})=\frac{1}{2n_3}\|\hat{\boldsymbol{X}}\hat{\boldsymbol{Y}}-\bar{\boldsymbol{C}}\|_F^2=\frac{1}{2n_3}\sum_{i=1}^{n_3}\|\hat{\boldsymbol{X}}^{(i)}\hat{\boldsymbol{Y}}^{(i)}-\bar{\boldsymbol{C}}^{(i)}\|_F^2$ *is the objective function and the sequence* $\{(\hat{\boldsymbol{X}}^k,\hat{\boldsymbol{Y}}^k,\boldsymbol{\mathcal{C}}^k)\}$ *generated by Algorithm 1 is bounded. Then, the sequence* $\{(\hat{\boldsymbol{X}}^k,\hat{\boldsymbol{Y}}^k,\boldsymbol{\mathcal{C}}^k)\}$ *satisfies the following properties:*

(1) $f(\hat{\boldsymbol{X}}^k,\hat{\boldsymbol{Y}}^k,\boldsymbol{\mathcal{C}}^k)$ *is monotonically decreasing. Actually, it satisfies the following inequality:*

$$\begin{aligned}&f(\hat{\boldsymbol{X}}^k,\hat{\boldsymbol{Y}}^k,\boldsymbol{\mathcal{C}}^k)-f(\hat{\boldsymbol{X}}^{k+1},\hat{\boldsymbol{Y}}^{k+1},\boldsymbol{\mathcal{C}}^{k+1})\\ &\geq\frac{1}{2n_3}\|\hat{\boldsymbol{X}}^{k+1}\hat{\boldsymbol{Y}}^{k+1}-\hat{\boldsymbol{X}}^k\hat{\boldsymbol{Y}}^k\|_F^2\geq 0.\end{aligned}\tag{19}$$

(2) *Any accumulation point* $(\hat{\boldsymbol{X}}_\star,\hat{\boldsymbol{Y}}_\star,\boldsymbol{\mathcal{C}}_\star)$ *of the sequence* $\{(\hat{\boldsymbol{X}}^k,\hat{\boldsymbol{Y}}^k,\boldsymbol{\mathcal{C}}^k)\}$ *is a KKT point of problem (13).*

Therefore, the theoretical convergence of Algorithm 1 can be guaranteed. It is worth mentioning that $\hat{\boldsymbol{X}}^0$ and $\hat{\boldsymbol{Y}}^0$ are randomly initialized. Besides, when a tensor reduces to a matrix, *i.e.*, $n_3=1$, the whole method and algorithm can still be applied and the convergence of the algorithm also holds. Thus, matrix completion is one special case that our method can deal with. We adopt similar proof sketch proposed in [3] to prove Theorem 2 in the supplementary material. But there are also differences since our convergence analysis couples two spaces, the original space and the Fourier space, and hence it is needed to further consider the properties of Fourier transformation and the block diagonal structure of the analyzed matrices.

D. Rank Estimation

In most cases, we do not know the true rank of tensor data. Thus, it is necessary to develop a method for estimating the rank of data. Similar to [25] and [3], we adopt the rank-decreasing method to estimate the true rank of a tensor. In this paper, to characterize the rank of data more accurately and efficiently, we estimate the rank for each block diagonal matrix $\bar{\boldsymbol{C}}^{(i)}$. Suppose that the multi-rank of $\bar{\boldsymbol{C}}^{(i)}$ $(i=1,\cdots,n_3)$ is $\boldsymbol{r}^k=[\boldsymbol{r}_1^k,\cdots,\boldsymbol{r}_{n_3}^k]$ at the k-th iteration. We compute the eigenvalues of $(\hat{\boldsymbol{X}}^{(i)})^*\hat{\boldsymbol{X}}^{(i)}$ $(i=1,\cdots,n_3)$ and then sort all these eigenvalues, and we can obtain $\boldsymbol{\lambda}_1\geq\boldsymbol{\lambda}_2\geq\cdots\geq\boldsymbol{\lambda}_{n_k}$, where $n_k=\sum_{i=1}^{n_3}\boldsymbol{r}_i^k$. Finally, we compute the quotients $\hat{\boldsymbol{\lambda}}_i=\boldsymbol{\lambda}_i/\boldsymbol{\lambda}_{i+1}$ $(i=1,\cdots,n_k-1)$. Assume that $t^k=\underset{1\leq i\leq n_k-1}{\operatorname{argmax}}\ \hat{\boldsymbol{\lambda}}_i$ and $\tau^k=(t^k-1)\hat{\boldsymbol{\lambda}}_{t^k}/\sum_{i\neq t^k}\hat{\boldsymbol{\lambda}}_i$. If $\tau^k\geq 10$, *i.e.*, there being a large drop in the magnitude of the eigenvalues, we should reduce $\boldsymbol{r}^k$. Similar to PCA [40], we find $\boldsymbol{\lambda}_{s^k}$ such that it meets $\sum_{i=1}^{s^k}\boldsymbol{\lambda}_i/\sum_{i=1}^{n_k}\boldsymbol{\lambda}_i\geq 95\%$. Assume there are $\boldsymbol{m}_i^k$ eigenvalues of $(\hat{\boldsymbol{X}}^{(i)})^*\hat{\boldsymbol{X}}^{(i)}$ which belong to $\{\boldsymbol{\lambda}_{s^k+1},\cdots,\boldsymbol{\lambda}_{n_k}\}$. Then we set $\boldsymbol{r}_i^k=\boldsymbol{r}_i^k-\boldsymbol{m}_i^k$. Suppose $\boldsymbol{U}^{(i)}\boldsymbol{\Sigma}^{(i)}(\boldsymbol{V}^{(i)})^T$ is the skinny SVD of $\hat{\boldsymbol{X}}^{(i)}\hat{\boldsymbol{Y}}^{(i)}$. We can update $\hat{\boldsymbol{X}}^{(i)}=\boldsymbol{U}^{(i)}_{\boldsymbol{r}_i^k}\boldsymbol{\Sigma}^{(i)}_{\boldsymbol{r}_i^k}$ and $\hat{\boldsymbol{Y}}^{(i)}=(\boldsymbol{V}^{(i)})^T_{\boldsymbol{r}_i^k}$, where $\boldsymbol{U}^{(i)}_{\boldsymbol{r}_i^k}$ consists of the first $\boldsymbol{r}_i^k$ columns of $\boldsymbol{U}^{(i)}$ and $\boldsymbol{V}^{(i)}_{\boldsymbol{r}_i^k}$ consists of the first $\boldsymbol{r}_i^k$ rows of $\boldsymbol{V}^{(i)}$. $\boldsymbol{\Sigma}^{(i)}_{\boldsymbol{r}_i^k}$ is a diagonal matrix whose diagonal entries are the largest $\boldsymbol{r}_i^k$ eigenvalues of $\boldsymbol{\Sigma}^{(i)}$. In this way, we can adjust the rank $\boldsymbol{r}^k$ and estimate the true rank of the tensor data.

E. Differences from Prior Work

Since both TNN [1] and our method are based on tensor tubal rank, we first explain the differences between these two methods.

(1) TNN [1] is a tensor nuclear norm minimization approach. It uses the tensor nuclear norm to approximate the tensor tubal rank and its model is formulated as

$$\min_{\boldsymbol{\mathcal{C}}}\ \sum_{i=1}^{n_3}\|\bar{\boldsymbol{C}}^{(i)}\|_*,\quad\text{s.t. } P_\Omega(\boldsymbol{\mathcal{C}}-\boldsymbol{\mathcal{M}})=\mathbf{0}.\tag{20}$$

In this way, TNN preserves the low-rank structure of the tensor. In contrast, our method employs low-rank tensor factorization method to characterize the low-rank property of the tensor. Thus, they are two different kinds of methods for tensor completion.

(2) Since these two methods adopt two different strategies to deal with the tensor rank issue, their algorithms are also very different. To solve the nuclear norm minimization problem (20), TNN has to compute t-SVD at each iteration and thus suffers from high computational cost. In contrast, our method only needs to update two tensors of smaller sizes and avoids t-SVD computation, leading to higher computational efficiency. Since tensor data are usually large scale, the algorithm efficiency becomes more important.

Now we compare our method with a concurrent similar work, *i.e.* Tubal-Alt-Min [41]. This work is also based on tensor tubal rank and tensor nuclear norm, and adopts the tensor factorization method for tensor completion. But there are also essential differences between these two concurrent methods.

(1) These two methods adopt very different alternating minimization algorithms. The optimization of Tubal-Alt-Min is more complex (see Algorithm 2 in [41]), which involves median least squares minimization and smooth QR decomposition at each iteration and has higher computational complexity (see discussion in Sec. III-B). By comparison, our algorithm is much simpler. At each iteration, our algorithm only needs to compute the closed form solutions of the variables (see Algorithm 1).

(2) Their rank estimation strategies also differ. Tubal-Alt-Min needs to know the target rank in advance, but mostly the true rank of a tensor of incomplete observations is unknown. So manually tuning the target rank or using

other methods (*e.g.* TNN [1]) to estimate the target rank is unavoidable. But both are time-consuming, especially for large-scale tensor data. Besides, the performance of Tubal-Alt-Min may be limited by the rank estimation methods, *e.g.* TNN [1]. For example, for tensor data of high tubal rank, TNN cannot provide a good estimated rank (see Fig. 1), leading to performance degradation of Tubal-Alt-Min. By comparison, our method integrates the factorization process with our rank estimation strategy and can adaptively estimate the target rank at each iteration. Fig. 1 demonstrates the validity of our rank estimation method which indeed outperforms TNN when the rank of a tensor is relatively high. So our method is actually more efficient, and our experiments have verified our better performance than Tubal-Alt-Min.

Now we compare our method with other factorization based methods.

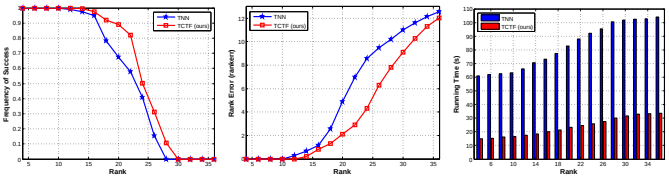
(1) The CP decomposition based methods, such as [30], compute the smallest number of rank one tensor decomposition and their optimization problems are generally NP hard [16], [31], [32], while matricization methods, such as TMac [3], directly unfold the tensor into matrices and then apply matrix factorization method to portray the low-rank structure of the tensor. In contrast, based on the tensor tubal rank, our method factorizes the tensor of interest into the product of two tensors of smaller sizes, which is different from other factorization based methods. Besides, the experimental results (*e.g.* Fig. 5 and 7) show our superior efficiency.

(2) Our model, optimization method, and the convergence analysis couple two spaces, the original space and the Fourier space, while the other factorization based methods only focus on the original space.

IV. Experiments

We conduct extensive experiments to evaluate our method, first testing it on synthetic data and then comparing it with other state-of-the-arts, including TMac [3], SiLRTC [2], TenALS [4], RMTC [5], TNN [1], and Tubal-Alt-Min [41] on image and video inpainting applications. Note that TMac has two versions, TMac-dec and TMac-inc, with the former using the rank-decreasing scheme to adjust its rank while the latter using the rank-increasing scheme. Please refer to [3]. The codes of TMac[1], SiLRTC[2], TenALS[3], RMTC[4] and TNN[5] are provided by their corresponding authors. As Eqns. (15) and (16) show, we can parallelly update all $\hat{\boldsymbol{X}}^{(i)}$ and $\hat{\boldsymbol{Y}}^{(i)}$ $(i = 1, \cdots, n_3)$, but for fair comparison of algorithm running time, we still employ the serial updating scheme in our code. In all experiments, our method randomly initializes $\hat{\boldsymbol{X}}^{(i)} = \texttt{randn}(n_1, \boldsymbol{r}_i^0)$ and $\hat{\boldsymbol{Y}}^{(i)} = \texttt{randn}(\boldsymbol{r}_i^0, n_2)$ $(i = 1, \cdots, n_3)$. For all methods, including ours, the stopping criteria are $\|\boldsymbol{V}_1^{k+1} - \boldsymbol{V}_1^k\|_{+\infty} \leq \texttt{tol}, \|\boldsymbol{V}_2^{k+1} - \boldsymbol{V}_2^k\|_{+\infty} \leq \texttt{tol}, \cdots, \|\boldsymbol{V}_l^{k+1} - \boldsymbol{V}_l^k\|_{+\infty} \leq \texttt{tol}$, where $\boldsymbol{V}_1, \boldsymbol{V}_2, \cdots, \boldsymbol{V}_l$ are l variables, and $\boldsymbol{V}_i^k$ and $\boldsymbol{V}_i^{k+1}$ denote the i-th updated variables at the k-th and the $(k+1)$-th iterations, respectively. In all experiments, we set $\texttt{tol} = 10^{-5}$. The platform is Matlab 2013a under Windows 8 on a PC of a 3.4GHz CPU and 8GB memory. Our code will be released online upon the acceptance of this paper.

[1]http://www.caam.rice.edu/~yx9/TMac/
[2]http://www.cs.rochester.edu/~jliu/publications.html
[3]http://web.engr.illinois.edu/~swoh/software/optspace/code.html
[4]https://bamdevmishra.com/codes/tensorcompletion/
[5]http://www.ece.tufts.edu/~shuchin/software.html



(a) (b) (c)

Fig. 1. Comparison of tensor completion on synthetic data. (a), (b), and (c) report the frequency of success, the rank error, and the algorithm running time (seconds), respectively. **Best viewed in $\times 2$ sized color pdf file.**

A. Synthetic Experiments

Both TNN [1] and our method TCTF are based on the tensor multi-rank and tubal rank definitions, depicting the inherent low-rank structure of a tensor by characterizing the low-rank property of their Fourier transform result. Here we conduct experiments to compare them in detail on synthetic data. Other methods are not compared here since they use different tensor rank definitions. Tubal-Alt-Min [41] is also excluded in comparison, since it needs to know the target rank in advance while the experiments here aim at investigating the tensor rank estimation ability and exact recovery performance of the algorithms.

We generate a low-rank tensor $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and Ω by the following method. First, we use Matlab command $\texttt{randn}(n_1, \bar{r}, n_3)$ and $\texttt{randn}(\bar{r}, n_2, n_3)$ to produce two tensors $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \bar{r} \times n_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{\bar{r} \times n_2 \times n_3}$. Then, let $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}$. Finally, we uniformly select $pn_1n_2n_3$ positions of $\boldsymbol{\mathcal{M}}$ to construct Ω, where p is the sampling ratio. In the experiments, we just set $n_1 = n_2 = n_3 = 100$ and $p = 0.6$. Suppose $\hat{\boldsymbol{\mathcal{C}}}$ is the recovered tensor of $\boldsymbol{\mathcal{M}}$. We adopt the relative error $\texttt{relerr} = \|\hat{\boldsymbol{\mathcal{C}}} - \boldsymbol{\mathcal{M}}\|_F^2 / \|\boldsymbol{\mathcal{M}}\|_F^2$ and the average algorithm running time as evaluation metrics. If $\texttt{relerr} \leq 10^{-2}$, $\hat{\boldsymbol{\mathcal{C}}}$ is regarded as a successful recovery to $\boldsymbol{\mathcal{M}}$. For fairness, we run these methods 50 times with the rank $\bar{r}$ varying from 4 to 36 with increment 2. Following common experiment settings [3], [25], we set the initialized rank $\boldsymbol{r}^0 = 1.5[\bar{r}, \cdots, \bar{r}] \in \mathbb{R}^{n_3}$ in our method.

In Fig. 1 (a), we report the frequency of success of these two methods. Our method performs a little better than TNN. Note the rank of each frontal slice of the Fourier transform result $\bar{\boldsymbol{M}}$ of the generated tubal rank $\bar{r}$ tensor $\boldsymbol{\mathcal{M}}$ is also $\bar{r}$. Thus, in Fig. 1 (b), we report the rank error $\texttt{rankerr} = \sum_{i=1}^{n_3} |\hat{\boldsymbol{r}}_i - \bar{r}| / n_3$, where $\hat{\boldsymbol{r}}_i$ is the estimated rank of the i-th frontal slice of $\bar{\boldsymbol{M}}$. We find that when the rank increases, the difficulty for accurate recovery also increases, since the degrees of freedom of the data are also increasing. But compared with TNN, our method could estimate a more accurate rank of

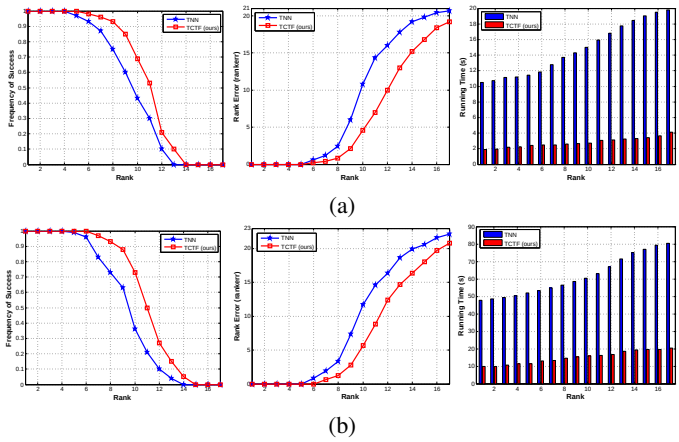


Fig. 2. Effects of parameters (the size of testing tensors and the sampling rate p) to the two methods. (a) $n_1 = 80, n_2 = 50, n_3 = 20$ and $p = 0.7$. (b) $n_1 = 50, n_2 = 80, n_3 = 100$ and $p = 0.8$. The left, middle, and right figures in (a) and (b) report the frequency of success, the rank error, and the algorithm running time (seconds), respectively. **Best viewed in $\times 2$ sized color pdf file.**

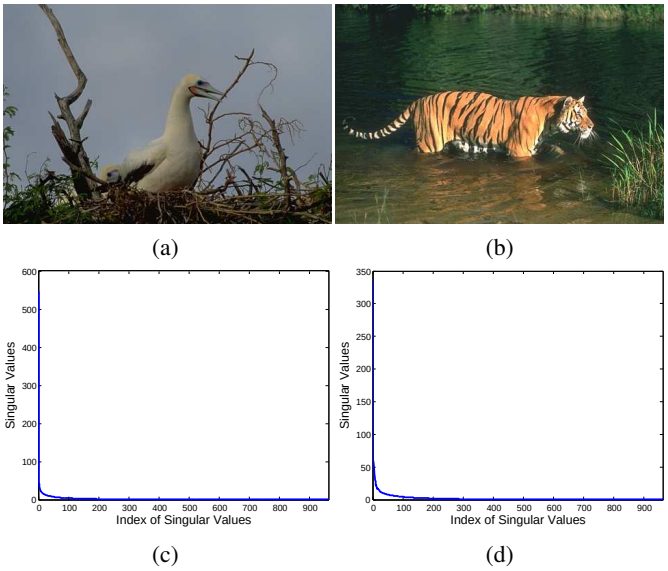


Fig. 3. Illustration of the low tubal rank property of the images in Berkeley Segmentation database. (a) and (b) are two images randomly selected from the database. (c) and (d) display the singular values of (a) and (b), respectively.

data. Therefore, our method achieves better recovery results. Fig. 1 (c) reports the average algorithm running time. Our method is about three times faster than the TNN method, since TNN needs to minimize the tensor nuclear norm and has to compute t-SVD at each iteration which is of high computation cost, especially when the data scale is large, while our method updates two tensors of much smaller sizes which is much more efficient. Actually, this result is also consistent with matrix completion conclusion that low-rank factorization method runs much faster than the matrix nuclear norm minimization methods [24]–[28].

To verify the robustness of our method to the parameters, *i.e.* the size of testing tensors and the sampling rate p, we conduct another two experiments. In the first experiment, we set $n_1 = 80, n_2 = 50, n_3 = 20$ and $p = 0.7$. Similar to the above experiment, we also report the frequency of successful recovery, the rank estimation error, and the average algorithm running time in Fig. 2 (a). In the second experiment, we set $n_1 = 50, n_2 = 80, n_3 = 100$ and $p = 0.8$ and report the experimental results in Fig. 2 (b). Although the size of the testing tensors and the sampling rate are very different, we can observe similar improvement of our method over TNN. Namely, when the tubal rank is relatively small, both methods can exactly recover the original low-rank tensor; and when faced with tensors of a relatively large tubal rank, our method works while TNN fails. For the estimation of the true tubal rank, our method can also gain better accuracy than TNN when handling tensors of a relatively small tubal rank, demonstrating its advantages. As for the running time, our method is always much faster than TNN. All these results are consistent with each other and those in Fig. 1. Thus, it is verified that our method is very robust to the parameters.

B. Real Data Experiments

We evaluate our method and other state-of-the-art methods on color image inpainting and gray scale video inpainting tasks. Color image and gray scale video are 3-way tensors and the inpainting task is to fill in the missing pixel values of a partially observed image or video. Assume that $\hat{\mathcal{C}}$ is the recovered tensor of $\mathcal{M} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. We employ the peak signal-to-noise ratio (PSNR), defined as

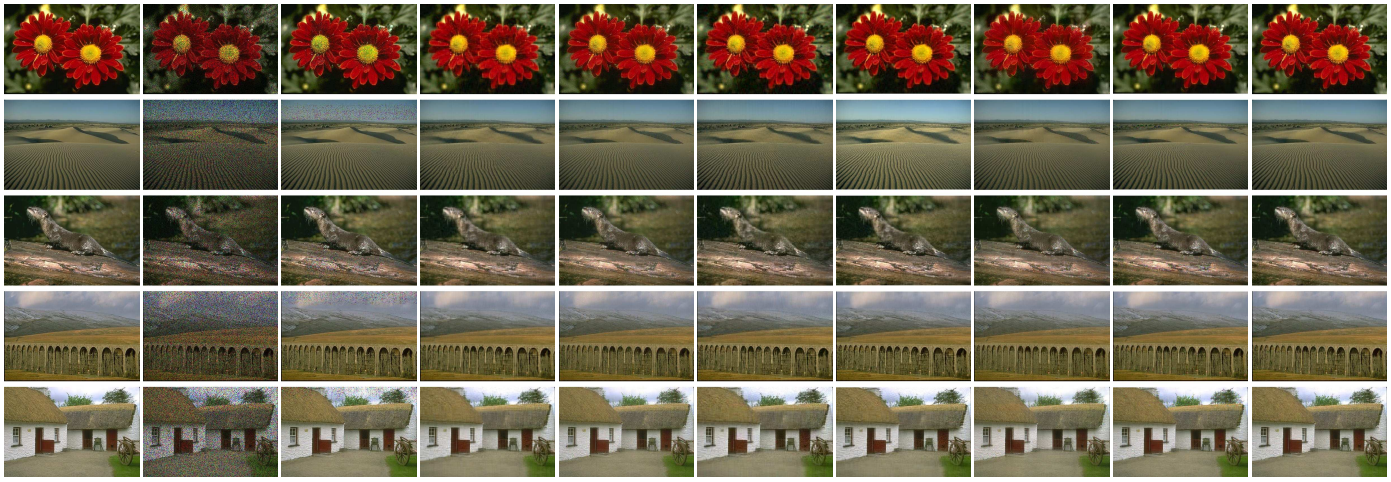
$$\text{PSNR} = 10 \log_{10} \left(\frac{n_1 n_2 n_3 \|\mathcal{M}\|_\infty^2}{\|\hat{\mathcal{C}} - \mathcal{M}\|_F^2} \right), \tag{21}$$

and the algorithm running time to evaluate these methods.

1) Image Inpainting: We use the Berkeley Segmentation database[6] [42] to evaluate our method for image inpainting. This database contains a wide variety of natural scenes. It has a total of 200 color images, each with size $321 \times 481 \times 3$. As pointed out by many works [20], [43]–[50], when arranging image or video data into matrices, they approximately lie on a union of low-rank subspaces, indicating the low-rank structure of the visual data. This is also true for tensor data. Actually, in Fig. 3 we plot the singular values of two images randomly selected from the dataset, most of which are very close to 0, and much smaller than the first several larger singular values. Indeed, in Fig. 3 (c) and (d), the numbers of singular values larger than 3% of the corresponding largest ones are 14 and 34, respectively, accounting for 1.5% and 3.5% of the total number 963 of the singular values, respectively. Thus, these images can be well approximated by the low tubal rank tensors.

In experiments, we randomly select 55 images from this database for testing. Then we randomly sample all testing images by the sampling ratio $p = 0.7$. We set the initialized rank $\boldsymbol{r}^0 = [30, 30, 30]$ in TMac-dec and $\boldsymbol{r}^0 = [3, 3, 3]$ with increment 2 in TMac-inc, and set the weights $\alpha_1 = \alpha_2 = \alpha_3 = 1/3$ for both versions as suggested in [3]. Following [2], the weight parameter $\boldsymbol{\alpha} = \boldsymbol{\lambda}/|\boldsymbol{\lambda}|_1$ for SiLRTC, where $\boldsymbol{\lambda} = [1, 1, 10^{-3}]$, and the penalty coefficient β is

[6]https://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

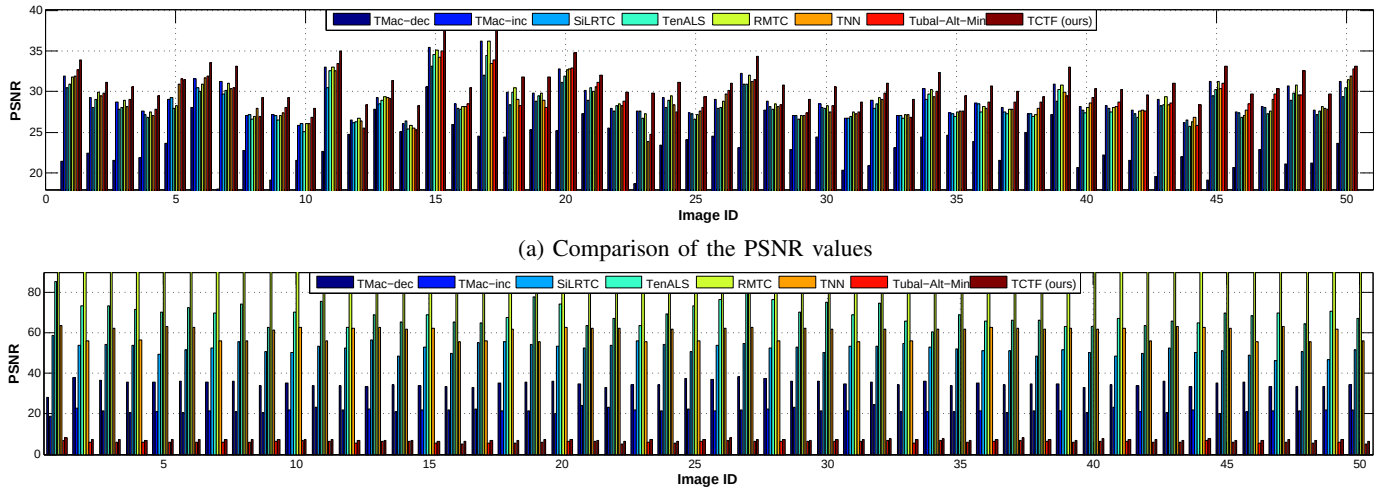


(a) Original (b) Observation (c) TMac-dec (d) TMac-inc (e) SiLRTC (f) TenALS (g) RMTC (h) TNN (i) Tubal-Alt-Min (j) TCTF

Image	TMac-dec		TMac-inc		SiLRTC		TenALS		RMTC		TNN		Tubal-Alt-Min		TCTF (ours)	
	PSNR	time	PSNR	time	PSNR	time	PSNR	time	PSNR	time	PSNR	time	PSNR	time	PSNR	time
Flower	22.86	23.83	28.63	18.03	28.14	37.91	27.39	68.96	28.83	1139.0	28.42	62.20	29.13	5.35	**30.80**	6.21
Desert	22.17	24.35	28.71	16.33	27.89	34.82	27.96	64.59	30.22	1230.5	29.35	58.20	26.18	5.01	**30.74**	5.49
River otter	22.96	24.77	26.94	19.76	27.76	37.38	26.41	66.54	28.41	1145.3	27.45	57.62	28.38	4.67	**29.10**	5.13
Viaduct	22.59	23.63	30.45	18.62	29.47	39.19	29.76	72.67	30.23	1109.9	31.53	60.37	30.07	4.94	**32.63**	5.88
House	20.64	23.49	26.76	17.85	26.62	36.30	25.85	77.21	26.48	1241.2	27.06	69.83	27.68	4.93	**28.92**	5.44

(k) PSNR and running time (seconds) on the above three images

Fig. 4. Examples of image inpainting. (a) is the original image. (b) is the observed image. (c)-(j) are the inpainting results of TMac-dec, TMac-inc, SiLRTC, TenALS, RMTC, TNN, Tubal-Alt-Min and our TCTF, respectively. (k) summaries the PSNR values and the running time (seconds) of all methods on the five testing images. **Best viewed in $\times 2$ sized color pdf file.**



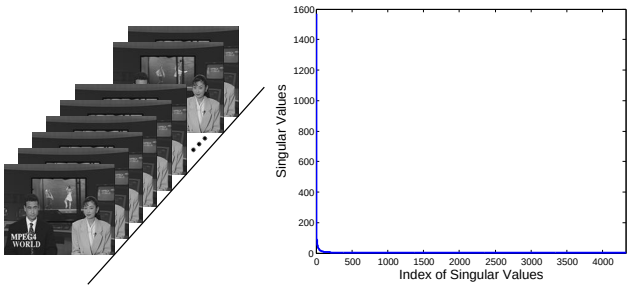
(a) Comparison of the PSNR values

(b) Comparison of the algorithm running time (seconds)

Fig. 5. Comparison of the PSNR values and the algorithm running time (seconds) on the randomly selected 50 images.

tuned by the method provided in [2]. In TenALS, the hyper-parameters α and γ are tuned by the methods provided in Section Experiment [4]. TNN is parameter free [1]. Since RMTC and Tubal-Alt-Min respectively need to know the true Tucker rank and the tensor tubal rank in advance, we just manually tune them. For fairness, in our method, we also set the initialized rank $\boldsymbol{r}^0 = [30, 30, 30]$ for all testing images.

We display the inpainting results of the five testing images in Fig. 4. Our method outperforms other methods on all of them. These methods, *i.e.* TMac, SiLRTC, TenALS and RMTC, directly unfold the tensor data into matrices and apply matrix nuclear norm or factorization method to preserve the low-rank structure of the tensor, thus may destroy multi-data structure and lead to performance degradation [1], [32], [33]. TNN, Tubal-Alt-Min and our method are based upon recent results on decomposition of a tensor and avoid the loss of structure information of the tensor [1], [33], thus can obtain better inpainting results. From the recovery results, our method recovers the details much better and it can well preserve the water-drops on the flowers, the edges of sand dune, the beards



(a) (b)

Fig. 6. Illustration of the low tubal rank property of the videos in YUV Video Sequences. (b) displays the singular values of the video (a) of 30 frames.

of the river otter, the leaves of trees, the textures of the viaducts and the mountains, and chimney above the house. It can be seen that our method is more superior.

We also report the PSNR values and the algorithm running time in Fig. 4 (k). Our method is the second fastest method, about three times faster than the third fastest method, TMac-inc, and at least ten times faster than TenALS and TNN. Since TNN has to compute t-SVD and DFT at each iteration, it runs slower. TMac, SiLRTC, TenALS and RMTC belong to matricization based tensor completion methods, which need to unfold the tensor data and recover each unfolded matrice. TMac-inc and TMac-dec run faster than SiLRTC and RMTC, since TMac-inc and TMac-dec use the low-rank factorization method, while SiLRTC adopts the nuclear norm and requires computing SVD and RMTC considers Riemannian manifold leading to very high computational cost. Besides, TenALS adopts a more complex convex relaxation instead of SNN to characterize the low-rank property of the tensor, so its optimization method is complex and time-consuming. Note, our method is only a little slower than Tubal-Alt-Min. There are two reasons. 1) Here we follow the experimental setting in [41] and use the simplified Tubal-Alt-Min algorithm for the inpainting task. The simplified algorithm has the same per-iteration complexity as ours, but it has no exact recovery performance guarantee. 2) Tubal-Alt-Min knows the target rank in advance which brings two advantages: less iterations required for convergence and much cheaper computation for updating variables thanks to small size of factorization tensors in Tubal-Alt-Min at the beginning. But the target rank is usually unknown in advance, and manually tuning or using other methods (*e.g.* TNN [1]) to estimate it will be very time-consuming, especially for large-scale data. Indeed, in the experiments, we first use TNN to estimate an accurate target rank and then tune it around the estimated rank. Note that the running time of Tubal-Alt-Min does not include that of TNN for estimating an accurate rank. Thus, our method is actually more efficient than Tubal-Alt-Min.

In Fig. 5, we report the PSNR values and the algorithm running time of all methods on the remaining 50 images. Our method performs the best with at least 1.2 dB improvement upon the PSNR metric on all 50 images, further verifying its advantages and robustness. From Fig. 5 (b), our method TCTF is much faster than other compared methods, except Tubal-Alt-Min. In conclusion, it not only achieves the best inpainting results but also runs very fast.

2) Video Inpainting: We evaluate our method on the widely used YUV Video Sequences[7]. Each sequence contains at least 150 frames. In the experiments, we test our method and other methods on three videos. The frame sizes of the first two videos both are 144×176 pixels and that of the last one is 146×120 pixels. We also plot the singular values of a video of size 146×120 and 30 frames in Fig. 6. Its number of singular values larger than 1% of the largest ones is 64, *i.e.* 1.5% of the total number 4,320 of singular values. It is worth mentioning that there are so many singular values which are very close to 0. Therefore, compared with images in Fig. 3, the video in Fig. 6 has much more redundant information because of its similar contents within and between frames, and thus its low tubal rank structure is more notable.

Due to the computational limitation, we only use the first 30 frames of the three sequences. We set the sampling ratio $p = 0.7$ and uniformly sample the videos to construct Ω. The weight parameter $\boldsymbol{\alpha} = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ in SiLRTC is suggested for video inpainting [2]. All other parameters of other four methods are set the same as above. Our method sets the initialized rank $\boldsymbol{r}^0 = [30, \cdots, 30] \in \mathbb{R}^{30}$.

As shown in Fig. 7, we display the 15th frame of the three testing videos, respectively. From the recovery results, our method performs better in filling the missing values of the three testing sequences. It can deal with the details of the frames better. On the PSNR metric, our method also achieves the best, consistent with the observation in Fig. 7. From time consumption, our method uses similar running time as Tubal-Alt-Min and is the second fastest method. It is about two times faster than the third fastest method TMac-inc and at least five times faster than the slowest method TenALS. We have discussed the reasons above. The video inpainting results are also consistent with the image inpainting results and all these demonstrate that our method can perform tensor completion better and runs more efficiently.

V. Conclusion

We propose a novel low-rank tensor factorization method for tensor completion. Based on the property of tensor tubal rank, our method depicts the low-rank structure of a tensor by factorizing the tensor into the product of two tensors of smaller sizes rather than minimizing tensor nuclear norm. Therefore, it avoids computing t-SVD and only needs to update and store two tensors of smaller sizes, leading to a higher algorithm efficiency. Compared with matricization methods that apply matrix nuclear norm or low-rank matrix factorization to each mode unfolding of the tensor, our method avoids destroying the inherent low-rank structure of the tensor. Besides, we prove that our proposed optimization method can converge to a KKT point. Experimental results demonstrate that our method not only obtains better performance but also runs faster than most compared state-of-the-art tensor completion methods.

[7]http://trace.eas.asu.edu/yuv/

(a) Original frame (b) Observation (c) TMac-dec (d) TMac-inc (e) SiLRTC (f) TenALS (g) RMTC (h) TNN (i) Tubal-Alt-Min (j) TCTF

Video	TMac-dec		TMac-inc		SiLRTC		TenALS		RMTC		TNN		Tubal-Alt-Min		TCTF (ours)	
	PSNR	time	PSNR	time	PSNR	time	PSNR	time	PSNR	time	PSNR	time	PSNR	time	PSNR	time
Hall Monitor	17.03	29.58	22.90	27.28	20.76	41.22	22.03	80.80	20.02	4216.2	23.65	75.85	24.15	12.41	**24.74**	14.65
Akiyo	17.46	29.80	22.11	27.22	20.81	39.31	21.91	83.92	18.56	4119.3	23.83	76.87	23.91	11.72	**25.42**	14.33
Bus	17.08	27.21	17.87	25.99	17.12	36.20	16.89	63.09	12.43	3402.8	18.66	62.72	18.36	9.29	**20.67**	11.72

(k) PSNR and running time (seconds) on the above three videos

Fig. 7. Examples of video inpainting. (a) Original frame. (b) Observed frame. (c)-(j) Inpainting results of TMac-dec, TMac-inc, SiLRTC, TenALS, RMTC, TNN, Tubal-Alt-Min and our TCTF. (k) PSNR values and running time (sec) of all methods on three testing videos. **Best viewed in $\times 2$ sized color pdf file.**

ACKNOWLEDGMENT

This research is partially supported by National Key Basic Research Project of China (973 Program) (grant no.s 2015CB352303 and 2015CB352502) and National Nature Science Foundation (NSF) of China (grant no.s 61671027 , 61625301, 61731018, and 61231002).

REFERENCES

[1] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer, "Novel methods for multilinear data completion and de-noising based on tensor-SVD," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2014, pp. 3842–3849.

[2] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 208–220, 2013.

[3] Y. Xu, R. Hao, W. Yin, and Z. Su, "Parallel matrix factorization for low-rank tensor completion," *Inverse Problems and Imaging*, vol. 9, no. 2, pp. 208–220, 2013.

[4] B. Romera-Paredes and M. Pontil, "A new convex relaxation for tensor completion," *Proc. Conf. Neutral Information Processing Systems*, pp. 2967–2975, 2013.

[5] H. Kasai and B. Mishra, "Low-rank tensor completion: a Riemannian manifold preconditioning approach," in *Int'l. Conf. Machine Learning*, 2016.

[6] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, "Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016.

[7] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low-n-rank tensor recovery via convex optimization," *Inverse Problems*, vol. 27, no. 2, pp. 25 010–25 028(19), 2010.

[8] M. Mørup, "Applications of tensor (multiway array) factorizations and decompositions in data mining," *Wiley Interdisciplinary Reviews Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 24–40, 2011.

[9] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering," in *Proc. ACM Conference on Recommender Systems*, 2010, pp. 79–86.

[10] Y. Liu and F. Shang, "An efficient matrix factorization method for tensor completion," *IEEE Signal Processing Letters*, vol. 20, no. 4, pp. 307–310, 2013.

[11] H. Tan, B. Cheng, W. Wang, Y. Zhang, and B. Ran, "Tensor completion via a multi-linear low-n-rank factorization model," *Neurocomputing*, vol. 133, no. 8, pp. 161–169, 2014.

[12] P. Zhou and J. Feng, "Outlier-robust tensor PCA," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017.

[13] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[14] L. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[15] R. Tomioka, T. Suzuki, K. Hayashi, and H. Kashima, "Statistical performance of convex tensor decomposition," in *Proc. Conf. Neutral Information Processing Systems*, 2011, pp. 972–980.

[16] C. Hillar and L. Lim, "Most tensor problems are NP-hard," *Journal of the ACM*, vol. 60, no. 6, p. 45, 2013.

[17] M. Collins and S. Cohen, "Tensor decomposition for fast parsing with latent-variable PCFGs," in *Proc. Conf. Neutral Information Processing Systems*, 2012, pp. 2519–2527.

[18] P. Liu, X. Qiu, and X. Huang, "Learning context-sensitive word embeddings with neural tensor skip-gram model," in *Proc. Int'l Joint Conf. on Artificial Intelligence*, 2015, pp. 1284–1290.

[19] B. Romera-Paredes, H. Aung, N. Bianchi-Berthouze, and M. Pontil, "Multilinear multitask learning," in *Proc. Int'l Conf. Machine Learning*, 2013, pp. 1444–1452.

[20] E. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2008.

[21] J. Cai, E. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[22] B. Recht, M. Fazel, and P. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM review*, vol. 52, no. 3, pp. 471–501, 2010.

[23] Y. Chen, "Incoherence-optimal matrix completion," *IEEE Trans. on Information Theory*, vol. 61, no. 5, pp. 2909–2923, 2013.

[24] R. Keshavan, S. Oh, and A. Montanari, "Matrix completion from a few entries," *IEEE Trans. on Information Theory*, vol. 56, no. 6, pp. 2980–2998, 2010.

[25] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *Mathematical Programming Computation*, vol. 4, no. 4, pp. 333–361, 2012.

[26] B. Recht and C. Ré, "Parallel stochastic gradient algorithms for large-scale matrix completion," *Mathematical Programming Computation*, vol. 5, no. 2, pp. 201–226, 2013.

[27] Y. Shen, Z. Wen, and Y. Zhang, "Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization," *Optimization Methods and Software*, vol. 29, no. 2, pp. 1–25, 2012.

[28] J. Tanner and K. Wei, "Low rank matrix completion by alternating steepest descent methods," *Applied and Computational Harmonic Analysis*, vol. 40, no. 2, pp. 417–429, 2015.

[29] N. Boumal and P. Absil, "RTRMC: A riemannian trust-region method for low-rank matrix completion," *Proc. Conf. Neutral Information Processing Systems*, pp. 406–414, 2011.
[30] H. Kiers, "Towards a standardized notation and terminology in multiway analysis," *Journal of Chemometrics*, vol. 14, no. 3, pp. 105–122, 2000.
[31] J. Landsberg, *Tensors: Geometry and Applications*. American Mathematical Society, 2012.
[32] C. Mu, B. Huang, J. Wright, and D. Goldfarb, "Square deal: Lower bounds and improved relaxations for tensor recovery," in *Proc. Int'l Conf. Machine Learning*, 2013, pp. 73–81.
[33] M. Kilmer, K. Braman, N. Hao, and R. Hoover, "Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 148–172, 2013.
[34] O. Semerci, N. Hao, M. Kilmer, and E. Miller, "Tensor-based formulation and nuclear norm regularization for multienergy computed tomography," *IEEE Trans. on Image Processing*, vol. 23, no. 4, pp. 1678–1693, 2014.
[35] M. Signoretto, Q. Dinh, L. Lathauwer, and J. Suykens, "Learning with tensors: A framework based on convex optimization and spectral regularization," *Machine Learning*, vol. 94, no. 3, pp. 303–351, 2014.
[36] M. Kilmer and C. Martin, "Factorization strategies for third-order tensors," *Linear Algebra and its Applications*, vol. 435, no. 3, pp. 641–658, 2011.
[37] C. Martin, R. Shafer, and B. LaRue, "An order-p tensor factorization with applications in imaging," *SIAM Journal on Scientific Computing*, vol. 35, no. 1, pp. 474–490, 2013.
[38] I. Oseledets, "Tensor-train decomposition." *SIAM Journal on Scientific Computing*, vol. 33, no. 33, pp. 2295–2317, 2011.
[39] Z. Zhang and S. Aeron, "Exact tensor completion using t-SVD," *IEEE Trans. on Signal Processing*, 2016.
[40] I. Jolliffe, "Principal Component Analysis," *Springer Berlin*, vol. 87, no. 100, pp. 41–64, 2010.
[41] X. Liu, S. Aeron, V. Aggarwal, and X. Wang, "Low-tubal-rank tensor completion using alternating minimization," in *arXiv:1610.01690*, 2016.
[42] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE Int'l. Conf. Computer Vision*, 2001, pp. 416–423.
[43] P. Zhou, Z. Lin, and C. Zhang, "Integrated low-rank-based discriminative feature learning for recognition," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 27, no. 5, pp. 1080–1093, 2016.
[44] P. Zhou, C. Zhang, and Z. Lin, "Bilevel model based discriminative dictionary learning for recognition," *IEEE Trans. on Image Processing*, vol. 26, no. 3, pp. 1173–1187, 2017.
[45] H. Zhang, Z. Lin, C. Zhang, and E. Chang, "Exact recoverability of robust pca via outlier pursuit with tight recovery bounds." in *AAAI*, 2015, pp. 3143–3149.
[46] Y. Wang, C. Xu, C. Xu, and D. Tao, "Beyond RPCA: Flattening complex noise in the frequency domain." in *AAAI*, 2017, pp. 2761–2767.
[47] P. Zhou, C. Fang, Z. Lin, C. Zhang, and E. Chang, "Dictionary learning with structured noise," *Neurocomputing*, 2017.
[48] Y. Wang, C. Xu, S. You, C. Xu, and D. Tao, "DCT regularized extreme visual recovery," *IEEE Trans. on Image Processing*, vol. 26, no. 7, pp. 3360–3371, 2017.
[49] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 210–227, 2009.
[50] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 171–184, 2013.

Pan Zhou received Master Degree in computer science from Peking University in 2016. Now he is a Ph.D. candidate at the Department of Electrical and Computer Engineering (ECE), National University of Singapore, Singapore. His research interests include computer vision, machine learning, and optimization.

Canyi Lu received the bachelor degree in mathematics from the Fuzhou University in 2009, and the master degree in the pattern recognition and intelligent system from the University of Science and Technology of China in 2012. He is currently a Ph.D. student with the Department of Electrical and Computer Engineering at the National University of Singapore. His current research interests include computer vision, machine learning, pattern recognition and optimization. He was the winner of the Microsoft Research Asia Fellowship 2014.

Zhouchen Lin (M'00-SM'08) received the Ph.D. degree in applied mathematics from Peking University, in 2000. He is currently a Professor with the Key Laboratory of Machine Perception, School of Electronics Engineering and Computer Science, Peking University. His research interests include computer vision, image processing, machine learning, pattern recognition, and numerical optimization. He is an area chair of CVPR 2014/2016, ICCV 2015, NIPS 2015, and a senior committee member of AAAI 2016/2017 and IJCAI 2016. He is an Associate Editor of the IEEE TRANSACTIONS ON PATTERNAN ALYSIS AND MACHINE INTELLIGENCE and the International Journal of Computer Vision. He is an IAPR Fellow.

Chao Zhang received the Ph.D. degree in electrical engineering from the Beijing Jiaotong University, Beijing, China in 1995. From 1995 to 1997, he was a postdoctoral research fellow at the National Laboratory on Machine Perception, Peking University. Since 1997 he has been an associate professor at Key Laboratory of Machine Perception (MOE), School of Electronics Engineering and Computer Science, Peking University. His research interests include image processing, statistical pattern recognition and visual recognition.