Low-Rank Sparse Tensor Approximations for Large High-Resolution Videos

Xiang Liu, Huyunting Huang, Weitao Tang, Tonglin Zhang, Baijian Yang
Purdue University
{xiang35, huan1182, tang384, tlzhang, byang}@purdue.edu

Abstract*—Tensor decomposition techniques are becoming increasingly important in processing videos with large sizes and dimensions. Under the framework of CANDECOMP/PARAFAC decomposition (CPD), this work studies low-rank sparse tensor approximations (LRSTAs) to higher-order tensors. Both theoretical and practical properties are evaluated for LRSTAs to represent large high-resolution videos. The evaluation brings three major contributions of this work. Firstly, the theoretical connection between CPD for high-order tensors and traditional singular value decomposition (SVD) for matrices are established, and the tensor rank for traditional SVD is defined. This provides a theoretical basis to compare tensor-based approach against matrix-based approach under the framework of tensor decompositions. Secondly, the non-orthogonality of CPD and its implications are revealed. The solution set of an LRSTA can only be used as a whole. Thirdly, a computationally efficient algorithm is developed. Its practical properties are also investigated in object detection and recognition in high-resolution videos. The results of the experiments showed that the proposed algorithm can handle large high-resolution videos very efficiently in terms of memory allocation. Results also revealed that commonly used total variations may not be a good evaluation metric for real world applications in computer vision. LRSTAs should be evaluated using the end goal of the applications, such as the accuracy of object detection and recognition.

***Index Terms*—Tensor decomposition, CP decomposition, PCA**

I. Introduction

In this era of big data, the sizes of all kinds of data are increasing significantly and so does the images and videos. The sizes of frames and length of time are both increasing, which leads to the great needs of an efficient video analyzing tool. Since tensor or the multi-way array can naturally represent images or videos, tensor decomposition techniques can be directly applied to those. These techniques are often developed under the framework of CANDECOMP/PARAFAC decomposition (CPD) [1]–[3] or Tucker decomposition (TKD) [4]. In this work, we examine the theoretical and the practical properties of LRSTA under the framework of CPD for high-resolution videos. We find that the choice of the rank value in CPD cannot be small if it is applied to high-resolution videos. Violation of orthogonality in CPD can cause a serious bias if the rank-one tensors are not used together. Properties of LRSTAs should be evaluated based on the results of object detection and recognition given by neural networks rather than a criterion developed based on relative total variations.

Because of the sizes of tensor data, it is hard to apply machine learning methods directly. Tensor decompositions are highly desirable to reduce the sizes of data. Both CPD and TKD can be treated as extensions of traditional singular Value decomposition (SVD) for matrices. The aim of CPD is to decompose a given tensor into a sum of a number of rank-one tensors. The aim of TKD is to decompose a given tensor into a core tensor multiplied by orthogonal matrices along their modes (i.e., coordinates). Since the core tensor is not sparse in general, CPD is often used to develop LRSTAs for higher-order tensors.

This work has three main contributions. The first is the establishment of theoretical connection between traditional SVD for matrices and CPD for higher-order tensors. We provide a mathematical formulation to interpret tensor rank for traditional SVD if it is applied individually to frames of a video. This provides a basis to theoretically compare tensor-based and matrix-based methods under the framework of tensor decompositions. The second is the investigation of nonorthogonality. Orthogonality of rank-one tensors in an LRSTA is generally violated. Because of this, the set of rank-one tensors given by an LRSTA must be used as a whole, which is fundamentally different from the approach in traditional PCA for matrices [5], [6]. The third is the evaluation of the practical properties of tensor CPD. Our experiments show that it is inappropriate to use a small rank value to compute an approximation if the sizes of the objects are relatively small, comparing to the sizes of frames of videos. To detect and recognize small objects in a video, a moderate rank value (*e.g.* a few thousand) is needed to strike the balance of data reduction and detection accuracy.

The rest of the article is organized as follows. In Section II, we provide a brief review of related work. In Section III, we introduce tensor notations and definition of CPD. In Section IV, we introduce our method, including its theoretical properties and numerical algorithms. In Section V, the proposed algorithm is implemented. The performance of object detection and recognition are evaluated and compared against the commonly used PCA appraoch. In Section VI, we conclude this article.

II. Related Work

In the literature, researchers have acknowledged that both CPD and TKD can be used to construct low-rank tensor approximations [7]. Both methods recommend calculating the low-rank approximation by minimizing the Frobenius norm between the given and approximate tensors. The approximations in both methods can be computed by the Alternating Least Square (ALS) algorithm.

Previous work of low-rank approximations to high-order tensors based on TKD can be found in [8]. The idea is to use a reduced size core tensor rather than the full-size core tensor to construct the approximation. Since the core tensor is generally dense, low-rank approximations are often constructed by CPD for high-order tensors. Statistical and computational properties of low-rank approximations were discussed in [7]. It is pointed out that the computation of the best low-rank approximation is NP-hard. Recently, [9] proposed a statistical model to construct rank-one approximations of high-order tensors and this problem is also NP-hard. More generally, it has been shown that most of the well-known tensor problems are NP-hard [10], including the best low-rank approximation problem for high-order tensors.A deeper dive into the computational complexity shows that NP hard issues are only associated with extreme cases. In fact, the ALS algorithm has been successfully applied to many real world problems [11]. The challenge is that directly applying ALS algorithm in FHD videos may cause out of memory error.

III. CPD

For simplicity, we limit the introduction of tensor notations and operations to the third-order tensors. Our approach can be easily extended to arbitrary-order tensors.

A. Notation and Operation

Third-order tensors are generally expressed as $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, where I, J, and K are the horizontal, lateral, and frontal dimensions of the given tensor. The (i, j, k)th entry of $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ is denoted by $\mathcal{X}_{ijk}$. The ith horizontal, the jth lateral, and the kth frontal slices are $J \times K$, $I \times K$, and $I \times J$ matrices denoted by $\mathcal{X}_{i::}$, $\mathcal{X}_{:j:}$, and $\mathcal{X}_{::k}$, respectively. The inner product of $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I\times J\times K}$ is defined as $\langle \mathcal{X}, \mathcal{Y}\rangle = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} \mathcal{X}_{ijk}\mathcal{Y}_{ijk}$. The Frobenius norm of $\mathcal{X}$ is $\|\mathcal{X}\| = \langle \mathcal{X}, \mathcal{X}\rangle^{1/2}$. Tensors $\mathcal{X}$ and $\mathcal{Y}$ are said orthogonal and denoted by $\mathcal{X} \perp \mathcal{Y}$ if $\langle \mathcal{X}, \mathcal{Y}\rangle = 0$. The outer product of vectors $\mathbf{a} = (a_1, \dots, a_I)^\top \in \mathbb{R}^I$, $\mathbf{b} = (b_1, \dots, b_J)^\top \in \mathbb{R}^J$, and $\mathbf{c} = (c_1, \dots, c_K)^\top \in \mathbb{R}^K$ is a third-order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ with $\mathcal{X}_{ijk} = a_i b_j c_k$. A third-order tensor is said rank-one if it can be expressed by an outer product of three vectors. The Kronecker product of vectors $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^J$ is given by $\mathbf{a} \otimes \mathbf{b} = (a_1\mathbf{b}^\top, \dots, a_I\mathbf{b}^\top)^\top$. It is an IJ-dimensional vector composed by products of all of the combinations of elements in $\mathbf{a}$ and $\mathbf{b}$. The Khatri-Rao product of matrices $\mathbf{A} \in \mathbb{R}^{I\times K}$ and $\mathbf{B} \in \mathbb{R}^{J\times K}$ is $(IJ) \times K$ matrix given by $\mathbf{A} \odot \mathbf{B} = \begin{bmatrix}\mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_K \otimes \mathbf{b}_K\end{bmatrix}$.

B. CPD for Third-Order Tensors

The concept of CPD can be traced back to about 93 years ago when [3] proposed to use the sum of rank-one tensors to express a given tensor. It became popular after its introduction to the psychometrics community in 1970 by [1], [2]. CPD is to factorize a given tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ into a sum of rank-one tensors, such that it can be expressed by

$$\mathcal{X} = [\![\boldsymbol{\lambda}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \qquad (1)$$

where R is a positive integer, λ_r is a positive real number, and $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, and $\mathbf{c}_r \in \mathbb{R}^K$ are unit vectors. In equation (1), $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_R)^\top \in \mathbb{R}^R$ is called the weight vector, and $\mathbf{A} \in \mathbb{R}^{I\times R}$, $\mathbf{B} \in \mathbb{R}^{J\times R}$, $\mathbf{C} \in \mathbb{R}^{K\times R}$ are called factor matrices. The rank of $\mathcal{X}$, denoted by $\text{rank}(\mathcal{X})$, is the minimum value of R such that (1) holds. The decomposition of (1) with $R = \text{rank}(\mathcal{X})$ is called the CP rank decomposition (CPRD) of $\mathcal{X}$. It has been pointed out that $\text{rank}(\mathcal{X}) \le \min\{IJ, IK, JK\}$ for any third-order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ [7]. Exact computation of the rank of a general third-order tensor is NP-hard [10].

IV. Low-Rank Sparse Tensor Approximation

As the rank of a third-order tensor is often large, full-rank CPD is rarely used in practice. Low-rank sparse tensor approximations (LRSTAs) are recommended. Rather than making the decomposition exactly equal to a given tensor $\mathcal{X}$, LRSTA provides an approximation.

A. Tensor-Based versus Matrix-based Methods

Suppose that the estimator is given by CPD with rank R, such that it can be expressed by $\hat{\mathcal{X}} = [\![\boldsymbol{\lambda}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$, where $\boldsymbol{\lambda} \in \mathbb{R}^R$, $\mathbf{A} \in \mathbb{R}^{I\times R}$, $\mathbf{B} \in \mathbb{R}^{J\times R}$, and $\mathbf{C} \in \mathbb{R}^{K\times R}$ are defined similarly as those given by (1). Then,

$$\hat{\mathcal{X}} = \operatorname*{argmin}_{\boldsymbol{\lambda},\mathbf{A},\mathbf{B},\mathbf{C}} \|\mathcal{X} - [\![\boldsymbol{\lambda}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|, \qquad (2)$$

A pre-selected R, called the rank of the approximation and denoted by $\text{rank}(\hat{\mathcal{X}})$, is needed in (2). Since R is unknown, the relative total variation (for residuals) given by

$$\text{rtv}(\hat{\mathcal{X}}) = \frac{\|\mathcal{X} - \hat{\mathcal{X}}\|^2}{\|\mathcal{X}\|^2} \qquad (3)$$

is needed. It provides the best R by

$$R_\alpha = \operatorname*{argmin}_{\text{rank}(\hat{\mathcal{X}}):\text{rtv}(\hat{\mathcal{X}})\ge 1-\alpha} \{\text{rtv}(\hat{\mathcal{X}})\}, \qquad (4)$$

where $\alpha \in [0, 1]$ is an relative error value. In other words, R_α is the minimum rank of $\hat{\mathcal{X}}$ such that it can account for at least $(1-\alpha)100\%$ total variations of $\mathcal{X}$. Obviously, we have $R_\alpha \le \text{rank}(\mathcal{X})$ and the equality holds if $\alpha = 0$. We call this the total variation approach for LRSTA. It is similar to traditional PCA for matrices.

Traditional PCA can also be used to approximate $\mathcal{X}$ by applying SVD to $\mathcal{X}_{::k}$ for individual $k \in \{1, \dots, K\}$. The approximation is provided by minimizing the relative total variations based on each $\mathcal{X}_{::k}$. Let $\hat{\mathcal{X}}_{PCA} = [\![\mathbf{d}; \mathbf{U}, \mathbf{V}, \mathbf{E}]\!]$ with $\mathbf{U} = \begin{bmatrix}\mathbf{U}_1 & \cdots & \mathbf{U}_K\end{bmatrix}$, $\mathbf{V} = \begin{bmatrix}\mathbf{V}_1 & \cdots & \mathbf{V}_K\end{bmatrix}$, $\mathbf{E} = \text{diag}(\mathbf{1}_{R_1}^\top, \dots, \mathbf{1}_{R_K}^\top)$, and $\mathbf{d} = (\mathbf{d}_1^\top, \cdots, \mathbf{d}_K^\top)^\top$, where $\mathbf{d}_k = (d_{k1}, \cdots, d_{kR_k})^\top \in \mathbb{R}^{R_k}$ is a vector expression of $\mathbf{D}_k$ and $\mathbf{1}_{R_k} \in \mathbb{R}^{R_k}$ is a vector with all of its elements equal to 1. Then, $\mathbf{U} \in \mathbb{R}^{I\times \sum_{k=1}^K R_k}$, $\mathbf{V} \in \mathbb{R}^{J\times \sum_{k=1}^K R_k}$ are matrices spanned by columns of $\mathbf{U}_k$ and $\mathbf{V}_k$, $\mathbf{E} \in \mathbb{R}^{K\times \sum_{k=1}^K R_K}$ is constructed by treating $\mathbf{1}_{R_k}^\top$ as $1 \times R_k$ matrix and putting them to diagonals, and $\mathbf{d} \in \mathbb{R}^{\sum_{k=1}^{K} R_k}$ is a vector spanned by singular values given by diagonal elements of $\mathbf{D}_k$ for all $k \in \{1, \cdots, K\}$. If the numbers of columns of $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{E}$ are all equal to a preselected R, then we can define a tensor version of relative total variations for $\hat{\mathcal{X}}_{PCA}$ similar to (3). We denote it by $\mathrm{rtv}_{PCA}(\hat{\mathcal{X}}_{PCA})$. If R is not given, then we can use an equation similar to (4) to solve R for any relative error value $\alpha \in [0, 1]$. We denote it by $R_{\alpha,PCA}$.

Theorem 1. $R_\alpha \le R_{\alpha,PCA}$ *for any* $\alpha \in [0, 1]$.

Proof. The conclusion is obvious because the form given by $\hat{\mathcal{X}}_{PCA}$ is a special case of the form given by $\hat{\mathcal{X}}$. □

Both $\hat{\mathcal{X}}$ and $\hat{\mathcal{X}}_{PCA}$ are approximations. Their ranks are usually much lower than the rank of $\mathcal{X}$. Therefore, they are treated as reduced rank or low-rank approximations. Note that $\mathcal{X}$ is derived by a tensor-based method but $\hat{\mathcal{X}}_{PCA}$ is derived by a matrix-based method. We cannot compare a tensor rank with a matrix rank, because the former is based on the entire tensor but the latter is based on the slice matrices. Therefore, we need to compare R_α with $R_{\alpha,PCA}$. By Theorem 1, for a given α, the number of rank-one tensors given by $\hat{\mathcal{X}}$ is lower than that given by $\hat{\mathcal{X}}_{PCA}$, implying that the approximation given by CPD is more efficient than that given by PCA. However, PCA for matrices enjoys orthogonality. Its relative total variation can be completely determined by singular values. We can use the largest R_k singular values to formula the best matrix approximation for each $\mathcal{X}_{::k}$. This property also hold when it is applied to third-order tensors.

Theorem 2. *Rank-one tensors given by* $\hat{\mathcal{X}}_{PCA}$ *are all orthogonal.*

Proof. Let $\mathbf{u}_r$, $\mathbf{v}_r$, and $\mathbf{e}_r$ be the rth columns of $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{E}$, respectively. We want to show $\langle \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{e}_r, \mathbf{u}_{r'} \circ \mathbf{v}_{r'} \circ \mathbf{e}_{r'} \rangle = \langle \mathbf{u}_r, \mathbf{u}_{r'} \rangle \langle \mathbf{v}_r, \mathbf{v}_{r'} \rangle \langle \mathbf{e}_r, \mathbf{e}_{r'} \rangle = 0$ for all $r \ne r'$. We consider two cases. In the first, we assume that there exists $k \in \{1, \ldots, K\}$ such that $\sum_{l=1}^{k-1} R_l < r, r' \le \sum_{l=1}^{k} R_l$. Then, $\mathbf{u}_r$ and $\mathbf{v}_r$ are the $(r - \sum_{l=1}^{k-1} R_l)$th columns of $\mathbf{U}_k$ and $\mathbf{V}_k$, respectively. We have $\langle \mathbf{u}_r, \mathbf{u}_{r'} \rangle = \langle \mathbf{v}_r, \mathbf{v}_{r'} \rangle = 0$ by orthogonality of $\mathbf{U}_k$ and $\mathbf{V}_k$. In the second, we assume that the condition is violated. For any $r < r'$, we can find $k \in \{1, \ldots, K-1\}$ such that $r \le \sum_{l=1}^{k} R_l < r'$. The positions of 1 in $\mathbf{e}_r$ and $\mathbf{e}_{r'}$ are inconsistent. We have implying that $\langle \mathbf{e}_r, \mathbf{e}_{r'} \rangle = 0$. □

We can easily derive the best tensor approximation in traditional PCA by examining the magnitudes of singular values because rank-one tensors given by $\hat{\mathcal{X}}_{PCA}$ are orthogonal. This idea does not apply to LRSTA because its rank-one tensors are not orthogonal. To overcome the difficulty, a numerical algorithm is needed.

B. Algorithm

The LRSTA given by (2) does not have a closed form solution. Numerical methods must be used. The most often used algorithm is the Alternating Least Squares (ALS). It is the standard algorithm for both CPD and TKD. Currently, ALS has been adopted by many software packages, including R, MATLAB, and Python. A well-known issue in ALS is the bottleneck caused by the matricized-tensor times Khatri-Rao product (MTTKRP) [12], [13]. We prove that this can be easily addressed. The following illustration uses third-order tensors only, but it can be extended to tensors of arbitrary-order.

ALS solves $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ in (2) individually in each of its iterations. In particular, suppose that ALS has been used to $\mathbf{C}$. For given $\mathbf{A}$ and $\mathbf{B}$, it solves $\mathbf{C}$ by $\mathbf{C} = \operatorname{argmin}_{\tilde{\mathbf{C}}} \|\mathbf{X} - [\![\mathbf{1}_R; \mathbf{A}, \mathbf{B}, \tilde{\mathbf{C}}]\!]\|^2$, leading to the solution as

$$\tilde{\mathbf{C}}(\mathbf{A}, \mathbf{B}) = \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})[(\mathbf{B}^\top \mathbf{B}) * (\mathbf{A}^\top \mathbf{A})]^\dagger, \tag{5}$$

where $\mathbf{X}_{(m)}$ (for $m = 1, 2, 3$) is the mode-m matricization (or unfolding) of $\mathcal{X}$ and $\mathbf{M}^\dagger$ represents the Moore-Penrose pseudoinverse of matrix $\mathbf{M}$. Similarly, we can define $\tilde{\mathbf{A}}(\mathbf{B}, \mathbf{C})$ and $\tilde{\mathbf{B}}(\mathbf{A}, \mathbf{C})$. By alternating $\mathbf{A} \leftarrow \tilde{\mathbf{A}}(\mathbf{B}, \mathbf{C})$, $\mathbf{B} \leftarrow \tilde{\mathbf{B}}(\mathbf{A}, \mathbf{C})$, and $\mathbf{C} \leftarrow \tilde{\mathbf{C}}(\mathbf{A}, \mathbf{B})$, the solution of $\hat{\mathcal{X}}$ is derived once the algorithm converges.

The bottleneck of MTTKPR appears in the computation of $\mathbf{B} \odot \mathbf{A}$, the Khatri-Rao product of $\mathbf{B}$ and $\mathbf{A}$, in (5). The term $\mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})$ in (5) is called the MTTKPR of $\mathbf{X}_{(3)}$ and $\mathbf{B} \odot \mathbf{A}$. The storage of the corresponding results needs a large amount of memory in size even if R is only moderately large. To overcome the difficulty, we propose the modified ALS method.

Let $\mathbf{A} = (a_{ir})_{I \times R}$, $\mathbf{B} = (b_{jr})_{J \times R}$, and $\tilde{\mathbf{C}} = (\tilde{c}_{kr})_{K \times R}$. Suppose that we want to solve $\tilde{c}_{kr}$ for all $k \in \{1, \cdots, K\}$ and $r = \{1, \cdots, R\}$ by (2). Then, we can use the squared error loss function as

$$L(\tilde{\mathbf{C}}) = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} \left(\mathcal{X}_{ijk} - \sum_{r=1}^{R} a_{ir} b_{jr} \tilde{c}_{kr} \right)^2. \tag{6}$$

Since the right side of (6) is a quadratic form, we can analytically minimize $L(\tilde{\mathbf{C}})$. In the end, we obtain

$$\tilde{\mathbf{C}}_{mod} = \mathbf{Z}\mathbf{Q}^\dagger \tag{7}$$

where $\mathbf{Z} = (z_{kr})_{K \times R}$ with $z_{kr} = \sum_{i=1}^{I} \sum_{j=1}^{J} \mathcal{X}_{ijk} a_{ir} b_{jr}$ and $\mathbf{Q} = (q_{r_1 r_2})_{R \times R}$ with $q_{r_1 r_2} = \sum_{i=1}^{I} \sum_{j=1}^{J} a_{ir_1} a_{jr_2} b_{jr_1} b_{jr_2}$. Similarly, we can derive $\tilde{\mathbf{A}}_{mod}$ and $\tilde{\mathbf{B}}_{mod}$. The proposed Modified ALS is illustrated in Algorithm 1.

Algorithm 1 Modified ALS

```
Input: Tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ and rank R
Output: $\hat{\mathcal{X}} = [\![\lambda; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ for (2)
Initialize A, B, C
repeat
    $\mathbf{A} \leftarrow \tilde{\mathbf{A}}_{mod}$, $\mathbf{B} \leftarrow \tilde{\mathbf{B}}_{mod}$, and $\mathbf{C} \leftarrow \tilde{\mathbf{C}}_{mod}$
until convergence requirements meet
Normalize columns of A, B, and C for $\lambda$
return $\lambda$, and the normalized A, B, and C.
```

Theoretically, $\tilde{\mathbf{C}}_{mod}$ and $\tilde{\mathbf{C}}$ are identical, but the computation of $\tilde{\mathbf{C}}_{mod}$ needs less memory. For instance, for FHD videos, we have $I = 1920$ and $K = 1080$. If 50 frames are considered, then $K = 50$ and the size of $\mathcal{X}$ for the video is about 0.78GB. The possible value of R can be as

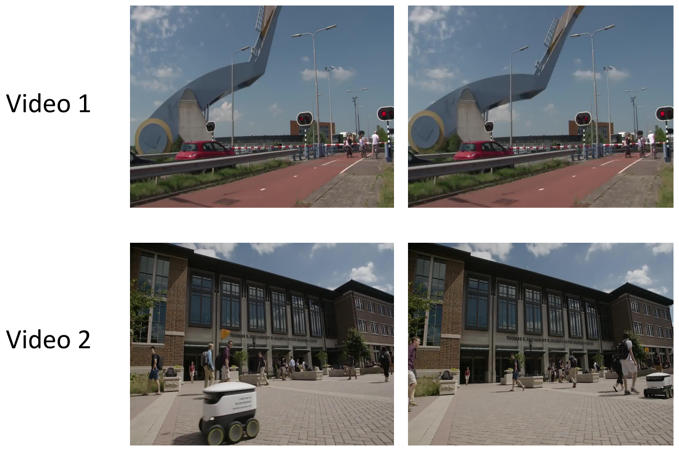


Figure 1. Dataset Overview

large as $1080 \times 50 = 5.4 \times 10^4$. Suppose that we choose $R = 1000$ in (2). Then, it is a small number, but $\mathbf{B} \odot \mathbf{A}$ contains $1920 \times 1080 \times 1000 = 2.0736 \times 10^9$ real numbers. A computer needs about 15GB memory to store the corresponding results. The size of memory needed in the computation can be several times higher. Because $\mathbf{Q}$ and $\mathbf{Z}$ only contain a few million real numbers, the implementation of our modified ALS needs much less memory (about $2 \times 0.78 = 1.56$GB) and can be carried out even by a PC.

V. Experiments

Four experiments were conducted in this work. The first experiment is to study the memory bottleneck of LRSTA. The second experiment is to compare the performance of LRSTA against PCA. The third experiment is to evaluate the impact of object detection and recognition problems on tensor decomposition. The fourth experiment is to relate the video resolutions with the numbers of rank-one tensors. All experiments were performed on a machine with Intel i7-8700K @ 3.70GHz CPU, 32GB RAM and 1T SSD. For color videos, the LRSTA and PCA are applied channel-wise. We employed scikit-learn for PCA algorithm [14]. We implemented LRSTA in python given by our Algorithm 1. The tolerance of PCA and LRSTA are set to 1e-4 and 1e-3 for all experiments. The maximum iterations for LRSTA is 50.

A. Dataset

Two videos were evaluated in our experiments, denoted as Video 1 and Video 2. An overview is shown in Figure 1. The resolution of Video 1 and 2 is 1280×720 and 1920×1080 respectively.

B. Experiments

Experiment I. The first experiment is to verify the memory bottleneck in algorithms for LRSTA based on the traditional ALS and our modified ALS. The memory bottleneck arises from the MTTKRP, which is mainly caused by the Khatri-Rao product. This issue is not contained in our modified ALS. The computation of Khatri-Rao product contains memory inflation for large-size tensor with many rank-one tensors, which is avoided in our method.

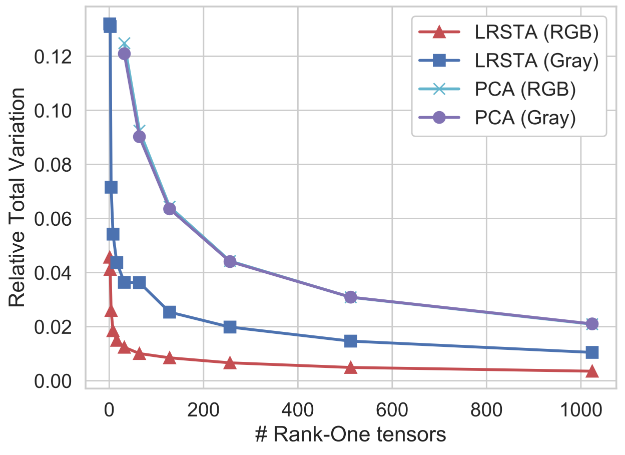
To reproduce the out-of memory (OOM) error of LRSTA, we used python package Tensorly and R package rTensor. We used the default settings for both Tensorly and rTensor. We generated $1280 \times 720 \times 32$ and $1280 \times 720 \times 64$ tensors sampling from the standard normal distribution. We chose number of rank-one tensors as $[1, 2, 4, \ldots, 1024]$. The OOM error occurred in both methods when the number of rank-tensors is larger than 128 for $1280 \times 720 \times 32$ and 64 for $1280 \times 720 \times 64$, respectively. This was caused by Khatri-Rao product, because the OOM did not appear in our proposed method. Our method successfully avoided the memory bottleneck caused by Khatri-Rao product in the computation of LRSTA. It worked well even when the rank-one tensor was 1024 for both $1280 \times 720 \times 32$ and $1280 \times 720 \times 64$ tensors. Moreover, we found that the modified ALS could save up to 99% memory. For $1280 \times 720 \times 32$ and 64 rank-one tensors, the computation of MTTKRP requires 14 gigabytes, but our proposed method only require approximately 500 megabyte. And for $1280 \times 720 \times 64$ and 32 rank-one tensors, the computation of MTTKRP also requires 14 gigabytes, but our proposed method only require approximately 250 megabytes.

Experiment II. The second experiment is to compare and contrast the performance of LRSTA and PCA by changing the number of rank-one tensors. We extracted 2 clips from each video. The first clip contained 32 frames and the second clip contains 64 frames. The experiments were performed on both color and gray-scale versions of the clips. The following number of rank-one tenors were evaluated: $\{1, 2, 4, 8, \ldots, 1024\}$. The metrics used in this experiment was the relative total variation, which is defined by equation (3) in Section IV-A.

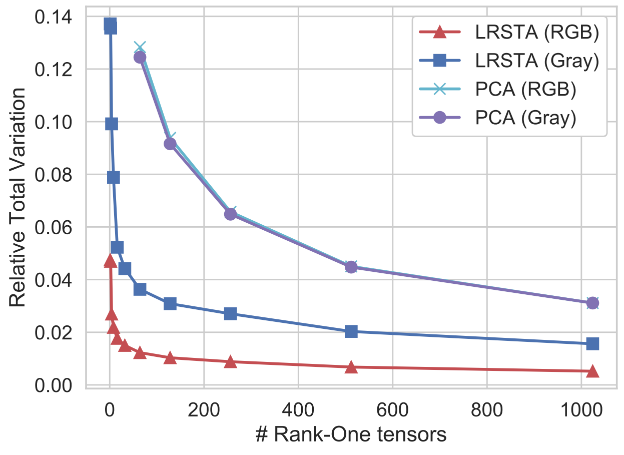
Figure 2 depicts the relationship of relative total variation with the number of rank-one tensors. It clearly shows that the increase of the number of rank-one tensors leads to the decrease of the relative total variation (for residuals) for both LRSTA and PCA. Results clearly show that the more rank-one tensors, the better tensor decomposition performance (less relative total variation). The curves for PCA are above the curves for LRSTA, indicating that the performance of LRSTA is better than that of PCA. Similar phenomena was also observed in the Video 1. Due to orthogonality, the curves for PCA are almost overlapped in both plots. In contrast, the curves for LRSTA are separated apart in both plots, with colored versions having lower relative total variations. This is caused by the non-orthogonality of CPD.

Experiment III. The third experiment studied object detection and recognition problems. Because the sizes of tensors are often large, tensor decompositions must be used to reduce the sizes of tensors. The contents of decomposed tensors should be evaluated by object detection and recognition methods. Therefore, we evaluated the impacts of object detection and recognition problems on both LRSTA and PCA for tensors.

We used a pre-trained YOLOv3 [15] model to measure our results. Figure 3 shows the detection results of Video



(a) Video 2 (32 frames)



(b) Video 2 (64 frames)

Figure 2. Relative total variation vs number of rank-one tensors for Video 2. In the legends of the plots, RGB stands for the colored version of Video 2 and Gray stands for the gray-scale version. The tensor size is 1920 × 1080 × 32 for (a) and 1920 × 1080 × 64 for (b).

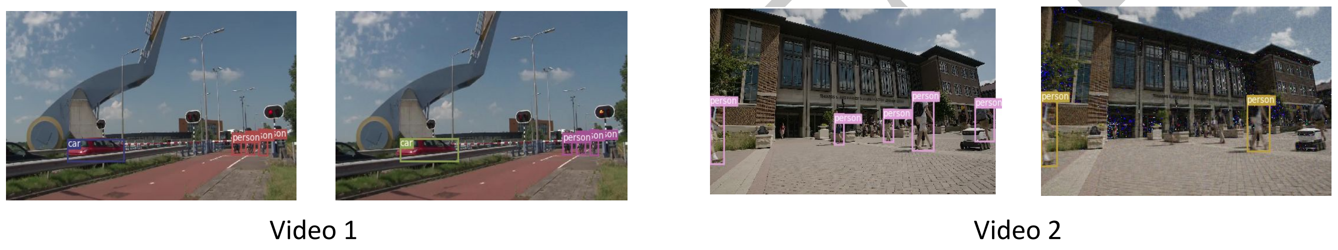


Figure 3. Detection results of LRSTA for Video 1 and 2 with 64 frames. The number of rank-one tensors is 1024. The left sub-figure for each video is the detection results of the original videos. The right sub-figures are the results of the reconstructed videos from LRSTA.

TABLE I
PRECISION, RECALL, F1 AND AP FOR VIDEO 2 WHEN THE NUMBER OF RANK-ONE TENSORS IS 4096.

METRIC	LRSTA			
R1 REMOVED	0	2	4	8
# OBJECTS	118	110	106	97
PRECISION	1.0	0.991	1.0	0.970
RECALL	0.620	0.582	0.561	0.513
F1	0.769	0.733	0.719	0.671
AP (PERSON)	0.624	0.582	0.561	0.513
METRIC	PCA			
R1 REMOVED	0	2	4	8
# OBJECTS	92	97	92	93
PRECISION	0.990	1.0	1.0	1.0
RECALL	0.487	0.513	0.487	0.492
F1	0.652	0.678	0.655	0.660
AP (PERSON)	0.487	0.513	0.487	0.492

1 and 2. The number of rank-one tensors is set to 1024. The confidence threshold for detection is set to 0.8 and the threshold of non-maximum merge is 0.4. In each panel, the left sub-figure is the detection results over the original video while the right sub-figures are the results of the reconstructed videos by LRSTA. The detection results of PCA reconstructed videos were poor because images were too blurry. More details can be found in supplementary materials. For Video 1, the detection performance of LRSTA is close to that of the original videos.

We then evaluated the impact of non-orthogonality. Because of the orthogonality of PCA, the rank-one tensors with smaller singular values can be safely removed without significantly impact the performance. However, orthogonality is not observed in LRSTA. This was verified in Experiment II by contrasting the performance before and after removing a few of rank-one

TABLE II
PRECISION, RECALL, F1 AND AP FOR VIDEO 1 AND VIDEO 2.

METRIC	VIDEO 1 (1280 × 720 × 64)			
R1 TENSORS	128	256	512	1024
# OBJECTS	128	128	135	135
PRECISION	1.0	1.0	0.993	0.993
RECALL	0.914	0.914	0.964	0.964
F1	0.955	0.955	0.978	0.978
AP (PERSON)	0.914	0.914	0.964	0.492
METRIC	VIDEO 2 (1920 × 1080 × 64)			
R1 TENSORS	128	256	512	1024
# OBJECTS	0	0	5	30
PRECISION	0	0	1.0	0.811
RECALL	0	0	0.013	0.771
F1	0	0	0.025	0.141
AP (PERSON)	0	0	0.013	0.070

tensors with smaller weight values. The object detection results of YOLOv3 from the original videos were used as the ground truth. Evaluation metrics include the number of objects (true positives), precision, recall, F1 scores, and average precision (AP) for the category of person (Table I). $0, 2, 4$ and 8 represent that the metrics are computed given the last $0, 2, 4$ and 8 of rank-one tensors removed. # OBJECTS denotes the number of true positives. This indicates that rank-one tensors, even those with the smallest weight values, are important in LRSTA. Removing rank-one tensors with smaller weight values can affect the performance of object detection.

Experiment IV. The fourth experiment is to study the impact of resolutions of videos on the choices of numbers of rank-one tensors. Table II displays how the metrics vary with the number of rank-one tensors. $128, 256, 512$ and 1024 represent that the metrics are computed given number of rank-one tensor equal $128, 256, 512$ and 1024. Note that the resolution is 1280×720 in Video 1 and 1920×1080 in Video 2. The detection performance increases with the number of rank-one tensors for both videos. For Video 2, the detection performance was so poor in the case of 128 and 256 that no objects were detected. The performance becomes slightly better when the number of rank-one tensor increases to 512 and significantly better in the case of 1024. Likewise for Video 1, the detection performance with 1024 rank-one tensors was also better than that of 128, 256 and 512 rank-one tensors. Our results given by Table II show that 1) for the videos with the same resolutions, the more rank-one tensors the better the object detection results; and 2) the higher the resolutions, the more rank-one tensors are needed to achieve better results in object detection.

In the experiments, we reproduced the memory bottle issue in the computation of LRSTA for large-size tensosr with a large number of rank-one tensors. We demonstrated that our modified memory-efficient ALS algorithm can overcome the difficulty. The second experiment compares the performance of LRSTA with PCA via changing the number of rank-one tensors. It points out that the relative total variation of LRSTA is less than that of PCA, but the results given by LRSTA are not orthogonal, leading to that they can only be used as a whole. The third experiment studies the object detection and recognition problems. It shows the impact of non-orthogonality in LRSTA by removing the rank-one tensors with smaller weights. The forth experiment studies the impact of resolutions of videos on the choices of number of rank-one tensors. For higher the video resolutions, we should choose larger number of rank-one tensors to achieve better reconstructed quality and detection performance.

VI. Conclusions and Future Work

This paper discusses how tensor decompositions can be used to reduce dimensions of HD videos for the deep learning purpose. A concept of *tensor rank for traditional SVD* is provided to connect CPD with SVD for matrices. Memory bottleneck in previous ALS for LRSTA is investigated. A memory-efficient method is proposed. The impact of non-orthogonality in CPD is discussed. This work also provides theoretical frameworks for low-rank approximations with the comparison of traditional SVD. Experiments further validate and reinforce the theoretical analyses via object detection.

Although our research focuses on CPD and PCA, similar issues also appear in TKD. It is not enough to only study relative total variations explained by TKD for tensor decomposition. Image detection and recognition problems must also be investigated. This is left to future research. To the best of our knowledge, this is the first work that links SVD and CPD based on object detection but not total variations. The deep learning method investigated in this work is currently limited to YOLOv3. We also plan to further optimize the ALS algorithms and integrate it with more deep learning methods and applications. As tensors are most used to represent images and videos, image detection and recognition issues should be involved in understanding properties of tensor decompositions. This is more important than their mathematical properties.

References

[1] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[2] R. A. Harshman *et al.*, "Foundations of the parafac procedure: Models and conditions for an "explanatory" multimodal factor analysis," 1970.

[3] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

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

[5] I. T. Jolliffe, "Principal components in regression analysis," in *Principal component analysis*. Springer, 1986, pp. 129–155.

[6] T. Zhang and B. Yang, "Big data dimension reduction using pca," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2016, pp. 152–157.

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

[8] M. Imaizumi and K. Hayashi, "Tensor decomposition with smoothness," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1597–1606.

[9] E. Richard and A. Montanari, "A statistical model for tensor pca," in *Advances in Neural Information Processing Systems*, 2014, pp. 2897–2905.

[10] C. J. Hillar and L.-H. Lim, "Most tensor problems are np-hard," *Journal of the ACM (JACM)*, vol. 60, no. 6, pp. 1–39, 2013.

[11] M. Baskaran, T. Henretty, B. Pradelle, M. H. Langston, D. Bruns-Smith, J. Ezick, and R. Lethin, "Memory-efficient parallel tensor decompositions," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2017, pp. 1–7.

[12] G. Ballard, K. Hayashi, and K. Ramakrishnan, "Parallel nonnegative cp decomposition of dense tensors," in *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*. IEEE, 2018, pp. 22–31.

[13] K. Hayashi, G. Ballard, Y. Jiang, and M. J. Tobia, "Shared-memory parallelization of mttkrp for dense tensors," in *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2018, pp. 393–394.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.