Robust Online Tucker Dictionary Learning from Multidimensional Data Streams

Le Trung Thanh[*†], Tran Trong Duy[†], Karim Abed-Meraim[*], Nguyen Linh Trung[†], and Adel Hafiane[*]
[*] University of Orléans, INSA-CVL, PRISME, France
E-mails: {trung-thanh.le,karim.abed-meraim}@univ-orleans.fr and adel.hafiane@insa-cvl.fr
[†] VNU University of Engineerning and Technology, AVITECH, Vietnam
E-mails: {duytt,linhtrung}@vnu.edu.vn

***Abstract*—Big data streaming analytics has recently attracted much attention in the signal and information processing communities due to the fact that massive streaming datasets have been collected over the years. Among them, many modern data streams are represented as multidimensional arrays (aka tensors), and thus, streaming tensor decomposition or tensor tracking has become a promising tool to analyze such streaming data. In this paper, we propose a novel online algorithm called ROTDL for the problem of robust tensor tracking under the Tucker format. ROTDL is not only capable of tracking the underlying Tucker dictionary of multidimensional data streams over time, but also robust to sparse outliers. The proposed algorithm is specifically designed by using the alternating direction method of multipliers, block-coordinate descent, and recursive least-squares filtering techniques. Several experiments demonstrate the effectiveness of ROTDL for robust tensor tracking.**

I. Introduction

Tensor decomposition (TD) has recently gained much attention from the signal processing and machine learning communities [1]–[3]. TD allows to factorize a tensor which a multiway array into basic components (e.g., vectors, matrices, or simpler/smaller tensors). Accordingly, it has become a powerful processing tool for multivariate and multidimensional data analysis. In practice, TD has been successfully applied to various domains, from computer vision [4]–[6] and wireless communications [7]–[9] to neuroscience [10]–[12].

Recently, the demand for real-time and stream processing is significantly increasing as a huge number of streaming data has been acquired over the years [13]. Factorizing tensors from multidimensional data streams is non-trivial due to several inherent problems of real-time processing, such as the unbounded volume (data size), concept drift or time-dependent and varying models, and uncertainty and/or noise. Despite these challenges, there have been many research efforts devoted to streaming tensor decomposition or tensor tracking so far. We refer the readers to [14] for a comprehensive survey on the state-of-the-art tensor tracking algorithms.

In this study, we focus on one of the most well-known and widely-used types of TD, namely Tucker decomposition. This decomposition can be considered as a multiway extension of SVD for higher-order tensors. Particularly under the Tucker format, we can express a tensor as a multilinear product of a small core tensor and a set of loading matrices [15]–[17]. This format offers several appealing features to represent

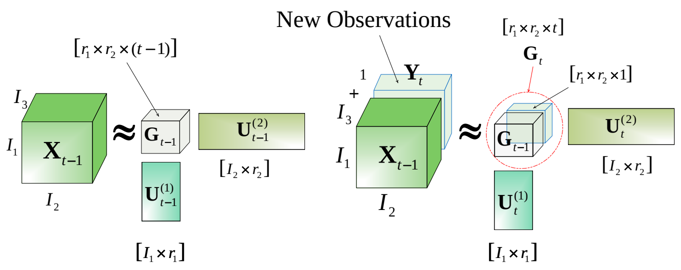


Fig. 1: Tracking the underlying Tucker representation of a 3rd-order streaming tensor $\boldsymbol{\mathcal{X}}_t$.

high-order tensors. For example, it allows to decompose any tensor of any order under a predefined error. Unlike the CP rank estimation which is known as a NP-hard problem, the Tucker rank (aka multilinear rank) can be determined in a stable and efficient way. Nowadays, gross corruptions are more and more ubiquitous in modern streaming datasets and systems [18]. They often have a pernicious effect on mining and knowledge discovery from data. A robust online variant of TD for streaming tensors called robust tensor tracking (RTT) has been emerging as a good approach. The main goal of this study is to propose an effective algorithm for RTT under the Tucker model.

In the literature, many adaptive (incremental) tensor algorithms have been proposed for streaming Tucker decomposition. They can be broadly categorized into three main classes: (i) tensor subspace tracking and (ii) online tensor dictionary learning, and (iii) multi-aspect streaming Tucker decomposition. The first class works under the assumptions that one of tensor modes/dimensions is increasing with time and the temporal slices (i.e., data streams) interact with the same core tensor of fixed size. Some notable algorithms belonging to this class are RPTucker [19], BASS-Tucker [20], and ATD [21]. In particular, RPTucker is an efficient Riemannian gradient-based algorithm and its stochastic variant can be used to track the underlying multilinear rank component of streaming tensors over time [19]. BASS-Tucker is a streaming Bayesian-based algorithm which is specifically designed for factorizing sparse streaming tensors [20]. ATD is a fast and effective adaptive Tucker decomposition which is capable of dealing with incomplete observations [21]. The second class, on the other hand, assumes that the underlying core of the streaming tensor has one dimension growing with time and each temporal slice of the core associates with a data stream. The very first algorithm belonging to this class is the so-called streaming tensor analysis (STA) in [22]. Particularly, STA is based on the incremental subspace learning on tensor unfolding matrices. Since then, several other algorithms following the same approach with STA were proposed, such as IRTSA [23], HO-RLSL [24], and RTSL [25]. Another good approach is based on online multimodal dictionary learning, such as OTDL [26], ORLTM [27], and D-L1-Tucker [28]. In particular, they apply a two-step learning procedure to track the tensor factors over time, including (i) inference of coefficients in the core tensor and (ii) dictionary update per each mode, see Fig. 1 for illustration. In this class, the core tensor is often supposed to be sparse, and hence, the former step is named as the tensor sparse coding. The third class is dedicated to tracking multi-aspect streaming tensors under the Tucker format over time. Such streaming tensors may evolve in multiple modes/dimensions over time. Two tensor algorithms in this class are SITTA in [29] and eOTD in [30]. SIITA provides an inductive framework for tracking the low-rank tensor approximation of multi-aspect streaming tensors and completing their missing entries with side information. eOTD adopts the divide and conquer framework to deal with multi-aspect streaming tensors. Despite having advantages, most of the existing streaming Tucker decomposition algorithms mentioned above are sensitive to sparse outliers. In the adaptive signal processing literature, there are some other streaming tensor methods robust to data corruptions such as in [31]–[34]. However, they are specifically designed under other tensor formats (i.e., CP/PARAFAC, tensor-train, and t-SVD). These drawbacks motivate us to design a new adaptive Tucker decomposition algorithm which is capable of dealing with sparse corruptions over time.

II. Preliminaries

A. Notations

In this paper, we adopt the following notational conventions. Scalars, vectors, and matrices are denoted by lowercase letters (e.g., x), boldface lowercase letters (e.g., $\mathbf{x}$), and boldface capital letters, respectively. Blackboard bold letters and bold calligraphic letters are used to represent sets/subsets/supports (e.g., $\mathbb{R}$) and high-order tensors (e.g., $\boldsymbol{\mathcal{X}}$), respectively. We denote by $x_{i_1,i_2,\dots,i_N}$ the $(i_1,i_2,\dots,i_N)$-th element of $\boldsymbol{\mathcal{X}}$ and its mode-n unfolding matrix is written as $\underline{\mathbf{X}}^{(n)}$. Symbols $(.)^\top$ and $(.)^\#$ represent the transpose and pseudo-inverse operators. Symbol $\|.\|$ denotes the Euclidean norm of a vector, matrix, and tensor. Next, we summarize some algebraic operators on tensors that are frequently used in this paper.

Considering a N-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, its n-mode product with $\mathbf{U}\in\mathbb{R}^{J\times I_n}$, written as $\boldsymbol{\mathcal{X}}\times_n\mathbf{U}$, results in a new tensor $\boldsymbol{\mathcal{Y}}\in\mathbb{R}^{I_1\times\cdots\times I_{n-1}\times J\times I_{n+1}\times\cdots\times I_N}$ satisfying $\underline{\mathbf{Y}}^{(n)} = \mathbf{U}\underline{\mathbf{X}}^{(n)}$. Its product with a sequence of N matrices $\{\mathbf{U}^{(n)}\}_{n=1}^N$

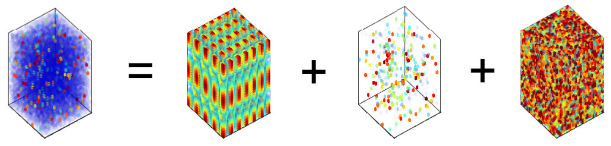


Fig. 2: Temporal tensor slice $\boldsymbol{\mathcal{Y}}_t$

along N modes is denoted by

$$\boldsymbol{\mathcal{X}}\prod_{n=1}^{N}\times_n\mathbf{U}^{(n)} = \boldsymbol{\mathcal{X}}\times_1\mathbf{U}^{(1)}\times_2\mathbf{U}^{(2)}\times_3\cdots\times_N\mathbf{U}^{(N)}. \quad (1)$$

The concatenation of $\boldsymbol{\mathcal{X}}$ with $\boldsymbol{\mathcal{Y}}\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_{N-1}}$, written as $\boldsymbol{\mathcal{X}}\boxplus\boldsymbol{\mathcal{Y}}$, results in $\boldsymbol{\mathcal{Z}}\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_N+1}$ with elements satisfying

$$z_{i_1,i_2,\dots,i_N} = \begin{cases} x_{i_1,i_2,\dots,i_N}, & \text{if } i_N \le I_N, \\ y_{i_1,i_2,\dots,i_{N-1}}, & \text{if } i_N = I_N+1. \end{cases} \quad (2)$$

The Kronecker product of a sequence of matrices in a reverse order is denoted by

$$\bigotimes_{n=1}^{N}\mathbf{U}^{(n)} = \mathbf{U}^{(N)}\otimes\mathbf{U}^{(N-1)}\otimes\cdots\otimes\mathbf{U}^{(1)}. \quad (3)$$

B. Tucker Decomposition

Tucker decomposition of a tensor $\boldsymbol{\mathcal{X}}\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ can be obtained by solving the following minimization

$$\underset{\boldsymbol{\mathcal{G}},\{\mathbf{U}^{(n)}\}_{n=1}^N}{\operatorname{argmin}} \Big\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}}\prod_{n=1}^{N}\times_n\mathbf{U}^{(n)}\Big\|_F^2, \quad (4)$$

where $\boldsymbol{\mathcal{G}}$ is the core tensor of size $r_1\times r_2\times\cdots\times r_N$, $\mathbf{r} = [r_1,r_2,\dots,r_N]$ is the desired low multilinear rank, and $\{\mathbf{U}^{(n)}\}_{n=1}^N$ with $\mathbf{U}^{(n)}\in\mathbb{R}^{I_n\times r_n}$ are the tensor factors (aka loading matrices) [1], [16]. This decomposition can be considered as a generalization of SVD for factorizing high-order tensors. Generally, the solution of (4) is not unique in the sense that we can rotate the columns of $\mathbf{U}^{(n)}$ by an orthogonal matrix $\mathbf{Q}^{(n)}\in\mathbb{R}^{r_n\times r_n}$ while still retaining the Tucker format. Interestingly, the column space covering the matrix $\mathbf{U}^{(n)}$ is unique, and hence, we often estimate subspaces of the tensor factors instead [1], [3]. Two widely-used algorithms for computing the Tucker decomposition are higher-order SVD (HOSVD) and higher-order orthogonal iteration (HOOI) [17].

III. Problem Formulation

In this work, we consider a $(N+1)$-th order streaming tensor $\boldsymbol{\mathcal{X}}_t$ fixing all but one mode (dimension). Without loss of generality, we assume the last mode of $\boldsymbol{\mathcal{X}}_t$ is temporal. Hence, we can write $\boldsymbol{\mathcal{X}}_t\in\mathbb{R}^{I_1\times\cdots\times I_N\times I_{N+1}^t}$ where I_{N+1}^t is growing with time and $\{I_n\}_{n=1}^N$ are constant. In addition, the τ-th temporal slice $\boldsymbol{\mathcal{Y}}_\tau\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_N\times 1}$ of $\boldsymbol{\mathcal{X}}_t$ is supposed to be generated under the following model

$$\boldsymbol{\mathcal{Y}}_\tau = \boldsymbol{\mathcal{L}}_\tau + \boldsymbol{\mathcal{O}}_\tau + \boldsymbol{\mathcal{N}}_\tau,\ 1\le\tau\le I_{N+1}^t, \quad (5)$$

see Fig. 2 for an illustration. Here, $\mathcal{N}_\tau$ is a Gaussian noise tensor, $\mathcal{O}_\tau$ is a sparse outlier tensor, and $\mathcal{L}_\tau$ is the low multilinear-rank component

$$\mathcal{L}_\tau = \mathcal{G}_\tau \prod_{n=1}^{N} \times_n \mathbf{U}^{(n)}, \tag{6}$$

where $\mathcal{G}_\tau \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$ (with $r_n \leq I_n \ \forall n$) contains the tensor coding coefficients and $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r_n}$ is the n-th tensor factor. The underlying tensor $\mathcal{X}_t$ is derived from appending the new slice $\mathcal{Y}_t$ to the previous $\mathcal{X}_{t-1}$ along the time dimension, i.e., $\mathcal{X}_t = \mathcal{X}_{t-1} \boxplus \mathcal{Y}_t$ with $I^t_{N+1} = I^{t-1}_{N+1} + 1$.

On the arrival of $\mathcal{Y}_t$ at each time t, we aim to update the tensor dictionary $\mathcal{D}_t = \{\mathbf{U}_t^{(n)}\}_{n=1}^N$ such that they can provide a good low multilinear-rank approximation for $\mathcal{X}_t$. Specifically, we propose to minimize the following objective function:

$$\mathcal{D}_t = \underset{\mathcal{D}}{\operatorname{argmin}} \Big[f_t(\mathcal{D}) \triangleq \frac{1}{t} \sum_{\tau=1}^{t} \beta^{t-\tau} \ell(\mathcal{Y}_\tau, \mathcal{D}) \Big], \tag{7}$$

where $\beta \in (0,1]$ is the forgetting parameter aimed to discount the effect of past observations and the loss function $\ell(\cdot)$ with respect to the τ-th slice $\mathcal{Y}_\tau$ is defined as

$$\ell(\mathcal{Y}_\tau, \mathcal{D}) = \min_{\mathcal{G},\mathcal{O}} \Big\| \mathcal{Y}_\tau - \mathcal{O} - \mathcal{G} \prod_{n=1}^{N} \times_n \mathbf{U}^{(n)} \Big\|_F^2 + \rho_1 \|\mathcal{O}\|_1 + \rho_2 \|\mathcal{G}\|_1, \tag{8}$$

where the first term of $\ell(.)$ measures the difference between the observation and estimation; the second term $\|\mathcal{O}\|_1$ and $\|\mathcal{G}\|_1$ are to promote the sparsity on the outlier tensor and coding coefficients. To support our algorithm presented in the next section, we assume that (i) the rank $[r_1, r_2, \ldots, r_N]$ is given in advance and (ii) the tensor dictionary $\mathcal{D}_t$ is fixed or slowly varying with time.

IV. Proposed Method

In this section, we propose a novel adaptive Tucker decomposition algorithm called ROTDL which stands for Robust Online Tucker Dictionary Learning. Particularly on the arrival of the new data $\mathcal{Y}_t$, ROTDL performs two main stages: (i) tensor sparse coding and (ii) Tucker dictionary update. In what follows, we describe each stage of ROTDL in detail.

A. Tensor Sparse Coding

Under the assumption that $\mathcal{D}_t \approxeq \mathcal{D}_{t-1}$, we can determine the tensor coding coefficients in $\mathcal{G}_t$ and the sparse outlier $\mathcal{O}_t$ by minimizing the loss function

$$\{\mathcal{G}_t, \mathcal{O}_t\} = \underset{\mathcal{G},\mathcal{O}}{\operatorname{argmin}} \ \ell(\mathcal{Y}_t, \mathcal{D}_{t-1}). \tag{9}$$

Particularly, vectorizing (9) results in

$$\{\mathbf{g}_t, \mathbf{o}_t\} = \underset{\mathbf{g},\mathbf{o}}{\operatorname{argmin}} \ \|\mathbf{y}_t - \mathbf{o} - \mathbf{H}_{t-1}\mathbf{g}\|_2^2 + \rho_1 \|\mathbf{o}\|_1 + \rho_2 \|\mathbf{g}\|_1, \tag{10}$$

where $\mathbf{H}_{t-1} = \otimes_{n=1}^N \mathbf{U}_{t-1}^{(n)}$. In practice, we can set $\rho_1 = \rho_2 = \rho$ and hence recast (10) into the standard LASSO problem

$$\{\mathbf{g}_t, \mathbf{o}_t\} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \ \|\mathbf{y}_t - \mathbf{A}_{t-1}\boldsymbol{\alpha}\|_2^2 + \rho \|\boldsymbol{\alpha}\|_1, \tag{11}$$

where $\boldsymbol{\alpha} = [\mathbf{o}^\top, \mathbf{g}^\top]^\top$ contains unknown parameters of interest and $\mathbf{A}_{t-1} = [\mathbf{I}, \ \mathbf{H}_{t-1}]$ where $\mathbf{I}$ is the identity matrix. Accordingly, provable LASSO solvers can be used to minimize (11) effectively. For example, we can apply the following ADMM-based solver introduced in [35] whose the k-th ADMM step is given by

- $\boldsymbol{\alpha}^k = (\mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} + \eta \mathbf{I})^{-1} (\mathbf{A}_{t-1}^\top \mathbf{y}_t + \eta(\mathbf{z}^{k-1} - \mathbf{e}^{k-1}))$
- $\mathbf{z}^k = \mathcal{S}_{\rho/2\eta}(\boldsymbol{\alpha}^k + \mathbf{e}^{k-1})$
- $\mathbf{e}^k = \mathbf{e}^{k-1} + \boldsymbol{\alpha}^k - \mathbf{z}^k$

where $\mathbf{z}^k$ and $\mathbf{e}^k$ are auxiliary variables, $\eta > 0$ is a small regularization parameter, and $\mathcal{S}_\gamma(.)$ is the soft-thresolding operator defined as

$$\mathcal{S}_\gamma(x) = \begin{cases} x - \gamma & \text{if } x \geq \gamma \\ 0 & \text{if } \gamma > x > -\gamma \ . \\ x + \gamma & \text{if } x \leq -\gamma \end{cases} \tag{12}$$

It is worth noting that as all three terms of (10) are convex, we can apply several alternating optimization methods to minimize it efficiently, e.g., alternating direction method of multipliers and block-coordinate descent methods.

B. Tucker Dictionary Update

In this stage, we update each factor $\mathbf{U}_t^{(n)}$ of $\mathcal{D}_t$ by

$$\mathbf{U}_t^{(n)} = \underset{\mathbf{U}^{(n)}}{\operatorname{argmin}} \Big[\frac{1}{t} \sum_{\tau=1}^{t} \beta^{t-\tau} \Big\| \mathcal{Y}_\tau - \mathcal{O}_\tau - \mathcal{G}_\tau \prod_{m=1,\neq n}^{N} \times_m \mathbf{U}_{t-1}^{(m)} \times_n \mathbf{U}^{(n)} \Big\|_F^2 \Big]. \tag{13}$$

We can reformulate (13) as follows

$$\mathbf{U}_t^{(n)} = \underset{\mathbf{U}^{(n)}}{\operatorname{argmin}} \Big[\frac{1}{t} \sum_{\tau=1}^{t} \beta^{t-\tau} \Big\| \mathbf{Y}_\tau^{(n)} - \mathbf{O}_\tau^{(n)} - \mathbf{U}^{(n)} \mathbf{W}_\tau^{(n)} \Big\|_F \Big], \tag{14}$$

where $\mathbf{W}_\tau^{(n)} = \mathbf{G}_\tau^{(n)} \big[\otimes_{m=1, m\neq n}^N \mathbf{U}_{t-1}^{(m)} \big]^\top$. The factor $\mathbf{U}_t^{(n)}$ can be obtained by setting the first derivative of (14) to zero

$$\mathbf{U}_t^{(n)} \Big[\sum_{\tau=1}^{t} \beta^{\tau-1} \mathbf{W}_\tau^{(n)} \big(\mathbf{W}_\tau^{(n)}\big)^\top \Big] = \sum_{\tau=1}^{t} \beta^{\tau-1} \big(\mathbf{Y}_\tau^{(n)} - \mathbf{O}_\tau^{(n)}\big) \big(\mathbf{W}_\tau^{(n)}\big)^\top. \tag{15}$$

Now, let us denote the two auxiliary matrices

$$\mathbf{V}_t^{(n)} = \sum_{\tau=1}^{t} \beta^{\tau-1} \mathbf{W}_\tau^{(n)} \big(\mathbf{W}_t^{(n)}\big)^\top, \tag{16}$$

$$\mathbf{Z}_t^{(n)} = \sum_{\tau=1}^{t} \beta^{\tau-1} \big(\mathbf{Y}_\tau^{(n)} - \mathbf{O}_\tau^{(n)}\big) \mathbf{W}_\tau^{(n)\top}. \tag{17}$$

Specifically, $\mathbf{V}_t^{(n)}$ and $\mathbf{Z}_t^{(n)}$ can be recursively updated as

$$\mathbf{V}_t^{(n)} = \beta\mathbf{V}_{t-1}^{(n)} + \mathbf{W}_t^{(n)}\big(\mathbf{W}_t^{(n)}\big)^\top, \tag{18}$$

$$\mathbf{Z}_t^{(n)} = \beta\mathbf{Z}_{t-1}^{(n)} + \big(\mathbf{Y}_t^{(n)} - \mathbf{O}_t^{(n)}\big)\big(\mathbf{W}_t^{(n)}\big)^\top. \tag{19}$$

Accordingly, the right hand side (RHS) of (15) is given by

$$\begin{aligned}\text{RHS of (15)} &= \beta\mathbf{Z}_{t-1}^{(n)} + \big(\mathbf{Y}_t^{(n)} - \mathbf{O}_t^{(n)}\big)\big(\mathbf{W}_t^{(n)}\big)^\top \\ &= \beta\mathbf{U}_{t-1}^{(n)}\mathbf{V}_{t-1}^{(n)} + \big(\mathbf{Y}_t^{(n)} - \mathbf{O}_t^{(n)}\big)\big(\mathbf{W}_t^{(n)}\big)^\top \\ &= \mathbf{U}_{t-1}^{(n)}\big(\mathbf{V}_t^{(n)} - \mathbf{W}_t^{(n)}\big(\mathbf{W}_t^{(n)}\big)^\top\big) + \big(\mathbf{Y}_t^{(n)} - \mathbf{O}_t^{(n)}\big)\big(\mathbf{W}_t^{(n)}\big)^\top \\ &= \mathbf{U}_{t-1}^{(n)}\mathbf{V}_t^{(n)} + \big(\mathbf{Y}_t^{(n)} - \mathbf{O}_t^{(n)} - \mathbf{U}_{t-1}^{(n)}\mathbf{W}_t^{(n)}\big)\big(\mathbf{W}_t^{(n)}\big)^\top. \end{aligned} \tag{20}$$

From (15) and (20), we obtain the following update rule

$$\mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \boldsymbol{\Delta}\mathbf{Y}_t^{(n)}\mathbf{Q}_t^{(n)}, \text{ where} \tag{21}$$

$$\boldsymbol{\Delta}\mathbf{Y}_t^{(n)} = \mathbf{Y}_t^{(n)} - \mathbf{O}_t^{(n)} - \mathbf{U}_{t-1}^{(n)}\mathbf{W}_t^{(n)}, \tag{22}$$

$$\mathbf{Q}_t^{(n)} = \big(\mathbf{W}_t^{(n)}\big)^\top\big(\mathbf{V}_t^{(n)}\big)^{-1}. \tag{23}$$

In order to enable the update (21), we can initialize $\mathbf{V}_0^{(n)} = \delta^{(n)}\mathbf{I}_{r_n}$ with a small number $\delta^{(n)} > 0$.

V. Experimental Results

In this section, we conduct several experiments to demonstrate the tracking ability of ROTDL. Particularly, its performance is evaluated in the following aspects: (i) effect of the additive noise, (ii) effect of the sparse outliers, (iii) its tracking ability in nonstationary and time-varying environments, and (iv) performance comparisons with the state-of-the-art online (adaptive) Tucker decomposition algorithms. Our MATLAB codes are available online at `https://github.com/thanhtbt/ROTDL`.

Experiment Setup: At each time $t > 0$, we generate the t-th temporal slice $\boldsymbol{\mathcal{Y}}_t \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ of the underlying streaming tensor $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I_1\times\cdots\times I_N\times t}$ as follows

$$\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{G}}_t \prod_{n=1}^{N} \times_n \mathbf{U}_t^{(n)} + \boldsymbol{\mathcal{O}}_\tau + \boldsymbol{\mathcal{N}}_\tau. \tag{24}$$

Here, the tensor $\boldsymbol{\mathcal{G}}_t$ is given by $\boldsymbol{\mathcal{G}}_t = \boldsymbol{\mathcal{P}}_t \circledast \boldsymbol{\mathcal{Q}}_t \in \mathbb{R}^{r_1\times r_2\times\cdots\times r_N}$ where $\boldsymbol{\mathcal{P}}_t$ is a binary mask tensor whose entries are i.i.d. Bernoulli random variables with probability $\omega_{\texttt{sparsity}}$; $\boldsymbol{\mathcal{Q}}_t$ is generated from a Gaussian distribution with zero mean and unit variance; and both $\boldsymbol{\mathcal{P}}_t$ and $\boldsymbol{\mathcal{Q}}_t$ share the same size with $\boldsymbol{\mathcal{G}}_t$. $\boldsymbol{\mathcal{O}}_t \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ is a sparse outlier tensor whose entries are drawn uniformly from the range $[0, \text{fac}_{\texttt{outlier}}]$ and the locations of outliers follow a Bernoulli distribution with probability $\omega_{\texttt{outlier}}$. $\boldsymbol{\mathcal{N}}_t \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ is a Gaussian noise tensor whose entries are derived from $\mathcal{N}(0,\sigma_n^2)$. The n-th tensor factor $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n\times r_n}$ is varied under the following data model:

$$\mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \varepsilon\mathbf{M}_t^{(n)}, \tag{25}$$

where $\mathbf{M}_t^{(n)} \in \mathbb{R}^{I_n\times r_n}$ is a Gaussian noise of zero mean and unit variant and $\varepsilon > 0$ is to control the time variation of $\mathbf{U}^{(n)}$ over time.

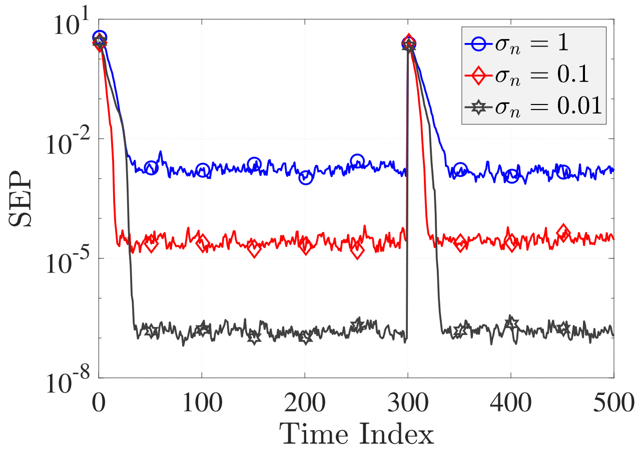


Fig. 3: Effect of the noise level σ_n

At $t = 0$, we draw $\boldsymbol{\mathcal{G}}_0$ and $\{\mathbf{U}_0^{(n)}\}_{n=1}^N$ from a Gaussian distribution with zero-mean and unit-variance.

We use the Subspace Estimation Performance (SEP) metric [36] to measure the tracking accuracy of algorithms:

$$\text{SEP} = \frac{1}{N}\sum_{n=1}^{N} \frac{\operatorname{tr}\Big\{\big(\mathbf{U}_{es}^{(n)}\big)^{\#}\big(\mathbf{I} - \mathbf{U}_{tr}^{(n)}\big(\mathbf{U}_{tr}^{(n)}\big)^{\#}\big)\mathbf{U}_{es}^{(n)}\Big\}}{\operatorname{tr}\Big\{\big(\mathbf{U}_{es}^{(n)}\big)^{\#}\big(\mathbf{U}_{tr}^{(n)}\big(\mathbf{U}_{tr}^{(n)}\big)^{\#}\big)\mathbf{U}_{es}^{(n)}\Big\}}, \tag{26}$$

where $\mathbf{U}_{tr}^{(n)}$ (resp. $\mathbf{U}_{es}^{(n)}$) refers to the ground truth (resp. estimation) of the n-th tensor factor $\mathbf{U}^{(n)}$ at each time t. Specifically, the denominator of (26) evaluates the sum of the squares of the cosines of the canonical angles between $\mathbf{U}_{es}^{(n)}$ and $\mathbf{U}_{tr}^{(n)}$. Meanwhile, its numerator measures the similar sum but for the estimation and the orthogonal complement of ground truth (see [36] for further details). Accordingly, the lower the value of SEP is, the better performance the algorithm has. In all experiments, we set the forgetting factor β to 0.5. The experimental results are averaged over 10 independent runs.

Effect of the noise level: We use a streaming tensor $\boldsymbol{\mathcal{X}}_t$ of size $10 \times 15 \times 20 \times 500$ whose temporal slices $\{\boldsymbol{\mathcal{Y}}_\tau\}_{\tau=1}^{500}$ are derived from the data model (24) with rank $[3,3,3]$. The sparsity level $\omega_{\texttt{sparsity}}$ in each core tensor $\boldsymbol{\mathcal{G}}_\tau$ is set to 0.3. To investigate the effect of noise, we vary the value of σ_n among $\{0.01, 0.1, 1\}$ while other parameters are kept constant (i.e., $\varepsilon = 0$ and $\omega_{\texttt{outlier}} = 0$). Also, an abrupt change is created at $t = 300$. The experimental results from Fig. 3 indicate that the noise level σ_n has an impact on both the convergence rate and estimation accuracy of ROTDL. The higher the value of σ_n is, the worse the performance of ROTDL is.

Effect of the time-varying factor: In this task, we reuse the underlying streaming tensor $\boldsymbol{\mathcal{X}}_t$ above. The noise level σ_n and the outlier density $\omega_{\texttt{outlier}}$ are set to 0.01 and 0, respectively. Here, we vary the value of ε among $\{0, 10^{-3}, 10^{-2}, 10^{-1}\}$ and then evaluate the SEP metric of ROTDL. As can be seen from Fig. 4 that ROTDL is fully capable of tracking the underlying Tucker dictionary of the streaming tensor in slowly time-varying environments. However, ROTDL does not work well

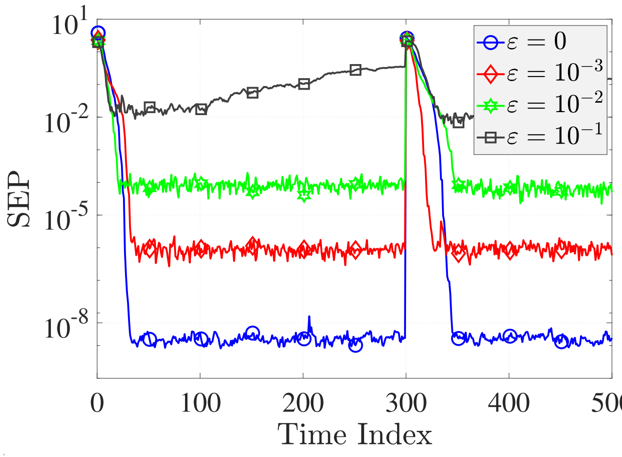
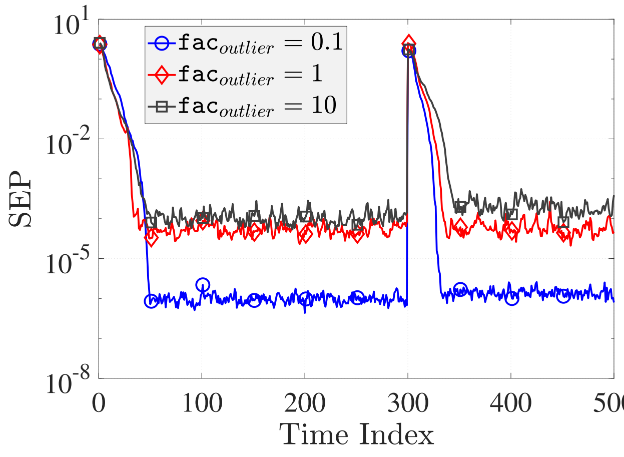
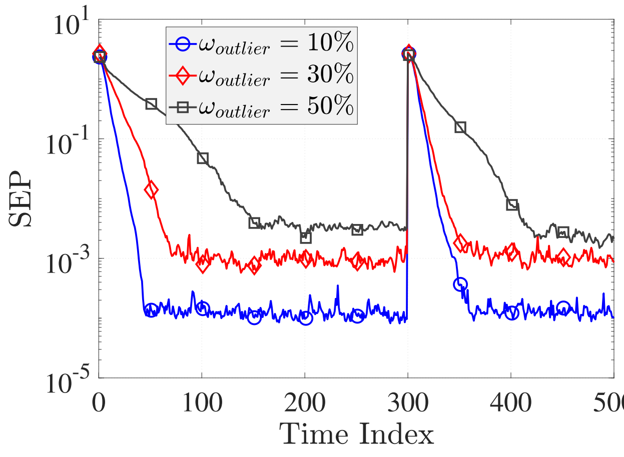


Fig. 4: Effect of the time-varying factor ε



(a) Outlier Intensity (with $\omega_{\texttt{outlier}}$ = 10%)



(b) Outlier density (with $\text{fac}_{\texttt{outlier}}$ = 10)

Fig. 5: Effect of Sparse Outliers.

when the time-varying factor is large (e.g., ε = 0.1).

Effect of the sparse outliers: Next, we study the robustness of ROTDL against sparse corruptions. The noise level σ_n and time-varying factor ε are both set to 10^{-3}. The values of the outlier intensity $\text{fac}_{\texttt{outlier}}$ and outlier density $\omega_{\texttt{outlier}}$ are, respectively, chosen in the set $\{0.1, 1, 10\}$ and $\{10\%, 30\%, 50\%\}$. Experimental results are illustrated in Fig. 5. We can see that ROTDL is robust to sparse outliers, even when they are strong (e.g., $\text{fac}_{\texttt{outlier}}$ = 10) and/or the number of corrupted entries is huge (e.g., $\omega_{\texttt{outlier}}$ = 50%).

Performance comparisons with the state-of-the-art online Tucker decomposition algorithms: Finally, the tracking ability of ROTDL is compared to that of DTA [22], STA [22], and ATD [21]. We consider two cases of sparse outliers (i.e., 5% and 20% corruptions) and two levels of the noise and time-varying factors (i.e., 10^{-2} and 10^{-3}). Fig. 6 indicates that ROTDL outperforms DTA, STA, and ATD completely.

VI. Conclusions

In this paper, we have considered the problem of robust tensor tracking under the Tucker format. A new effective online Tucker decomposition algorithm called ROTDL was proposed to track the underlying Tucker dictionary of streaming tensors over time. ROTDL can deal with imperfect multidimensional data streams from noisy and slowly time-varying environments. Experimental results show that ROTDL outperforms the state-of-the-art online Tucker decomposition algorithms, especially when streaming tensors are corrupted by outliers.

Acknowledgment

This work is supported by Project CN.22.xx granted by the University of Engineering and Technology, Vietnam National University, Hanoi.

References

[1] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.

[2] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and A. H. Phan, “Tensor decompositions for signal processing applications: From two-way to multiway component analysis,” *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, 2015.

[3] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, “Tensor decomposition for signal processing and machine learning,” *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, 2017.

[4] M. A. O. Vasilescu and D. Terzopoulos, “Multilinear analysis of image ensembles: Tensorfaces,” in *Eur. Conf. Comput. Vision*, 2002, pp. 447–460.

[5] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, “MPCA: Multilinear principal component analysis of tensor objects,” *IEEE Trans. Neural Netw.*, vol. 19, no. 1, pp. 18–39, 2008.

[6] Y. Panagakis, J. Kossaifi, G. G. Chrysos, J. Oldfield, M. A. Nicolaou, A. Anandkumar, and S. Zafeiriou, “Tensor methods in computer vision and deep learning,” *Proc. IEEE*, vol. 109, no. 5, pp. 863–890, 2021.

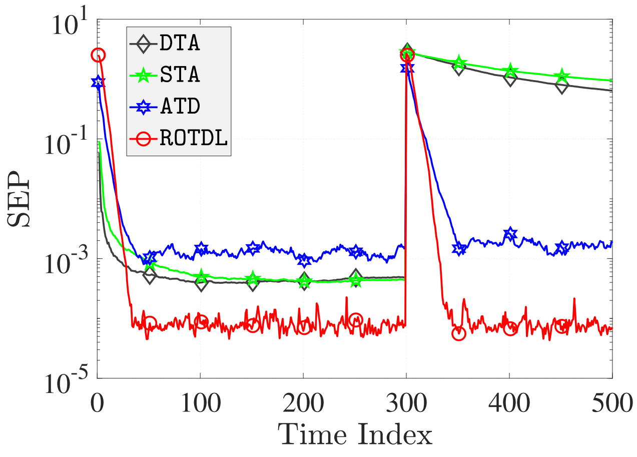
[7] G. Favier and A. L. de Almeida, “Tensor space-time-frequency coding with semi-blind receivers for MIMO wireless communication systems,” *IEEE Trans. Signal Process.*, vol. 62, no. 22, pp. 5987–6002, 2014.

[8] J. Feng, L. T. Yang, X. Liu, and R. Zhang, “Privacy-preserving tensor analysis and processing models for wireless internet of things,” *IEEE Wirel. Commun.*, vol. 25, no. 6, pp. 98–103, 2018.

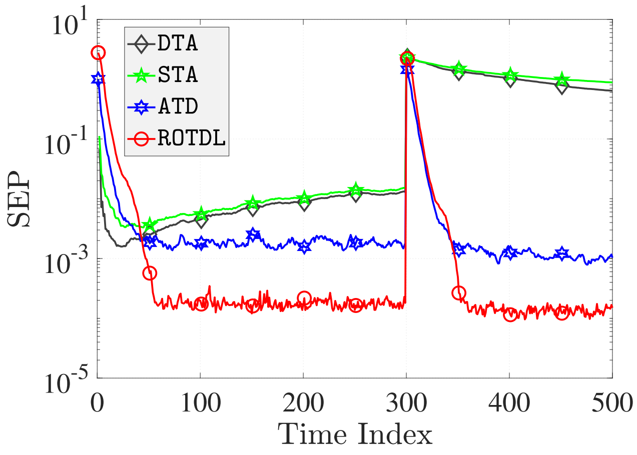
[9] H. Chen, F. Ahmad, S. Vorobyov, and F. Porikli, “Tensor decompositions in wireless communications and MIMO radar,” *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 3, pp. 438–453, 2021.

[10] E. Karahan, P. A. Rojas-Lopez, M. L. Bringas-Vega, P. A. Valdés-Hernández, and P. A. Valdes-Sosa, “Tensor analysis and fusion of multimodal brain images,” *Proc. IEEE*, vol. 103, no. 9, pp. 1531–1559, 2015.

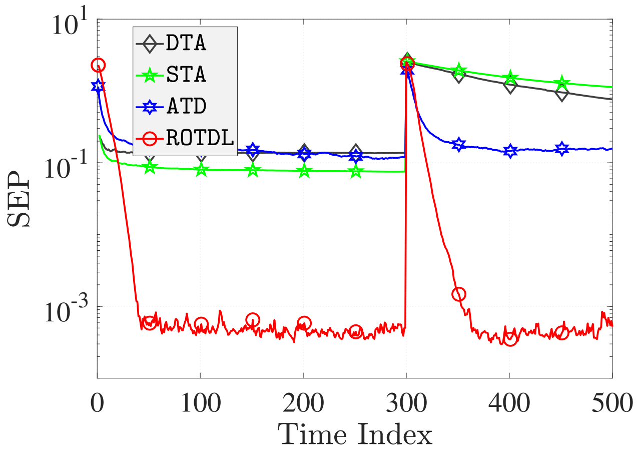
[11] A. G. Mahyari, D. M. Zoltowski, E. M. Bernat, and S. Aviyente, “A



(a) $\omega_{\texttt{outlier}} = 5\%$ and $\sigma_n = \varepsilon = 10^{-3}$



(b) $\omega_{\texttt{outlier}} = 5\%$ and $\sigma_n = \varepsilon = 10^{-2}$



(c) $\omega_{\texttt{outlier}} = 20\%$ and $\sigma_n = \varepsilon = 10^{-3}$

Fig. 6: Performance comparisons between the state-of-the-art online Tucker decomposition algorithms.

tensor decomposition-based approach for detecting dynamic network states from EEG," *IEEE Trans. Biomed. Eng.*, vol. 64, no. 1, pp. 225–237, 2016.

[12] N. T. A. Dao, N. V. Dung, N. L. Trung, K. Abed-Meraim *et al.*, "Multi-channel EEG epileptic spike detection by a new method of tensor decomposition," *J. Neural Eng.*, vol. 17, no. 1, p. 016023, 2020.

[13] T. Kolajo, O. Daramola, and A. Adebiyi, "Big data stream analysis: A systematic literature review," *J. Big Data*, vol. 6, no. 1, pp. 1–30, 2019.

[14] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "A contemporary and comprehensive survey on streaming tensor decomposition," *Techrxiv*, 2022. [Online]. Available: https://doi.org/10.36227/techrxiv.20105966.

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

[16] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2000.

[17] ——, "On the best rank-1 and rank-(r_1,r_2,...,r_n) approximation of higher-order tensors," *SIAM J. Matrix Anal. Appl.*, vol. 21, pp. 1324–1342, 2000.

[18] T. Akidau, S. Chernyak, and R. Lax, *Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing*, 2018.

[19] H. Kasai and B. Mishra, "Low-rank tensor completion: A Riemannian manifold preconditioning approach," in *Int. Conf. Mach. Learn.*, 2016, pp. 1012–1021.

[20] S. Fang, R. M. Kirby, and S. Zhe, "Bayesian streaming sparse Tucker decomposition," in *Conf. Uncertain. Artif. Intell.*, 2021, pp. 558–567.

[21] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "Tracking online low-rank approximations of incomplete high-order streaming tensors," *Techrxiv*, 2022. [Online]. Available: https://doi.org/10.36227/techrxiv.19704034.

[22] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, "Incremental tensor analysis: Theory and applications," *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 3, pp. 1–37, 2008.

[23] W. Hu, X. Li, X. Zhang, X. Shi, S. Maybank, and Z. Zhang, "Incremental tensor subspace learning and its applications to foreground segmentation and tracking," *Int. J. Comput. Vis.*, vol. 91, no. 3, pp. 303–327, 2011.

[24] A. Ozdemir, E. M. Bernat, and S. Aviyente, "Recursive tensor subspace tracking for dynamic brain network analysis," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 4, pp. 669–682, 2017.

[25] J. Li, G. Han, J. Wen, and X. Gao, "Robust tensor subspace learning for anomaly detection," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 89–98, 2011.

[26] A. Traore, M. Berar, and A. Rakotomamonjy, "Online multimodal dictionary learning," *Neurocomput.*, vol. 368, pp. 163–179, 2019.

[27] P. Li, J. Feng, X. Jin, L. Zhang, X. Xu, and S. Yan, "Online robust low-rank tensor modeling for streaming data analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1061–1075, 2019.

[28] D. G. Chachlakis, M. Dhanaraj, A. Prater-Bennette, and P. P. Markopoulos, "Dynamic L1-norm Tucker tensor decomposition," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 3, pp. 587–602, 2021.

[29] M. Nimishakavi, B. Mishra, M. Gupta, and P. Talukdar, "Inductive framework for multi-aspect streaming tensor completion with side information," in *ACM Int. Conf. Inf. Knowl. Manag.*, 2018, pp. 307–316.

[30] H. Xiao, F. Wang, F. Ma, and J. Gao, "eOTD: An efficient online tucker decomposition for higher order tensors," in *IEEE Int. Conf. Data Min.*, 2018, pp. 1326–1331.

[31] L. Dongjin and S. Kijung, "Robust factorization of real-world tensor streams with patterns, missing values, and outliers," in *IEEE Int. Conf. Data Eng.*, 2021, pp. 840–851.

[32] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "Robust tensor tracking with missing data and outliers: Novel adaptive CP decomposition and convergence analysis," *IEEE Trans. Signal Process.*, pp. 1–16, 2022.

[33] ——, "Robust tensor tracking with missing data under tensor-train format," in *Eur. Signal Process. Conf.*, 2022, pp. 832–836.

[34] M. M. Salut and D. V. Anderson, "Online tensor robust principal component analysis," *IEEE Access*, vol. 10, pp. 69 354–69 363, 2022.

[35] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[36] L. T. Thanh, N. V. Dung, N. L. Trung, and K. Abed-Meraim, "Robust subspace tracking with missing data and outliers: Novel algorithm with convergence guarantee," *IEEE Trans. Signal Process.*, vol. 69, pp. 2070–2085, 2021.