

# Streaming data preprocessing via online tensor recovery for large environmental sensor networks

YUE HU, Vanderbilt University, USA

AO QU, Vanderbilt University, USA

YANBING WANG, Vanderbilt University, USA

DANIEL B. WORK, Vanderbilt University, USA

Measuring the built and natural environment at a fine-grained scale is now possible with low-cost urban environmental sensor networks. However, fine-grained city-scale data analysis is complicated by tedious data cleaning including removing outliers and imputing missing data. While many methods exist to automatically correct anomalies and impute missing entries, challenges still exist on data with large spatial-temporal scales and shifting patterns. To address these challenges, we propose an online robust tensor recovery (OLRTR) method to preprocess streaming high-dimensional urban environmental datasets. A small-sized dictionary that captures the underlying patterns of the data is computed and constantly updated with new data. OLRTR enables online recovery for large-scale sensor networks that provide continuous data streams, with a lower computational memory usage compared to offline batch counterparts. In addition, we formulate the objective function so that OLRTR can detect structured outliers, such as faulty readings over a long period of time. We validate OLRTR on a synthetically degraded National Oceanic and Atmospheric Administration temperature dataset, with a recovery error of 0.05, and apply it to the Array of Things city-scale sensor network in Chicago, IL, showing superior results compared with several established online and batch-based low rank decomposition methods.

CCS Concepts: • **Information systems** → **Spatial-temporal systems**; *Data mining*; • **Computing methodologies** → **Anomaly detection**; **Factorization methods**.

Additional Key Words and Phrases: robust tensor recovery, tensor factorization, multilinear analysis, outlier detection, internet of things, urban computing

## 1 INTRODUCTION

### 1.1 Motivation

The United Nations established 17 Sustainable Development Goals that are to be achieved by 2030 [1]. One of the goals is to promote sustainable and resilient, inclusive and safe cities. To quantify the effects of the built environment on micro climate and other environmental impacts, many urban-scale environmental sensing initiatives exploiting emerging *internet of things* (IoT) technologies are being developed (e.g., [2, 3]). These projects measure block-by-block micro-climate quantities to inform better green infrastructure investment, transportation planning, and energy-saving designs.

Despite the high spatial resolution information provided by the low-cost sensors, data quality and data treatment still remain major concerns that hinders a wider adoption of these technologies [4, 5]. Outliers and missing data are amongst the challenges. Current approaches to clean the datasets prior to interpretation are often limited in functionality for which anomalies or missing data are independently addressed [6–8].

A promising direction to overcome these limitations involves tensor factorization methods, which have shown state-of-the-art performance in detecting outliers and imputing missing data [9, 10]. Because large-scale urban sensor networks usually produce higher-order data that contains spatial

---

Authors' addresses: Yue Hu, Vanderbilt University, 1025 16th Ave S, Suite 102, Nashville, USA, yue.hu@vanderbilt.edu; Ao Qu, Vanderbilt University, 1025 16th Ave S, Suite 102, Nashville, USA, ao.qu@vanderbilt.edu; Yanbing Wang, Vanderbilt University, 1025 16th Ave S, Suite 102, Nashville, USA, yanbing.wang@vanderbilt.edu; Daniel B. Work, Vanderbilt University, Nashville, USA, dan.work@vanderbilt.edu.

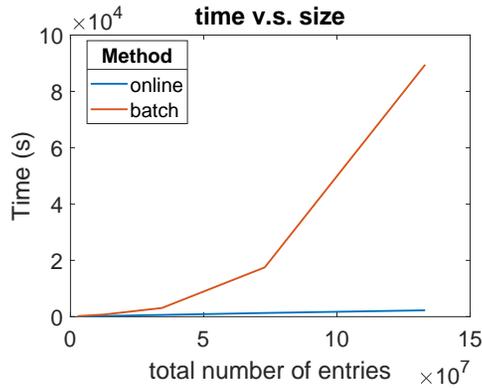


Fig. 1. Computation time as a function of input tensor size for batch-based method [10] and our online-based method (proposed). For batch-based tensor decomposition, computation time increases sharply with data size, thus impractical for large scale systems.

and temporal relationships, low-rank recovery can be naturally applied. Yet these batch-based methods rely on collecting the data samples for all time, and re-solving the problem when new data arrives, making them sub-ideal for streaming sensor networks deployed for continuous monitoring. As depicted in Fig. 1, computation time increases sharply with data size, posing challenge for the real-world applications.

## 1.2 Contribution

**To overcome the challenges of batch tensor factorization methods, we develop an *online robust tensor recovery* (OLRTR) algorithm to pre-process streaming data from large-scale urban sensor networks.**

The main contribution of this work is to introduce OLRTR to automatically correct errors and impute missing data common to large distributed urban sensor networks. OLRTR computes and sequentially updates a small-sized dictionary that stores the underlying, time-varying patterns of the data, which significantly lowers the memory usage and can adapt to shifting patterns in the datasets.

Two real-world experiments demonstrate the effectiveness of OLRTR. The first uses a complete, high quality *National Oceanic and Atmospheric Administration* (NOAA) temperature dataset [11], which is artificially degraded by injecting known outliers and also by removing some entries to simulate missing data. We demonstrate that the proposed tensor factorization approach correctly identifies the outliers and recovers accurate values for the missing data. The second experiment applies the method to raw and incomplete temperature data from the state of the art IoT platform known as the *Array of Things* (AoT) urban sensing platform in Chicago, IL [12]. The recovered temperature data is validated by comparing to nearby NOAA readings. The experimental results show the superiority of OLRTR over several online and batch-based methods, as well as the potential of OLRTR to provide reliable data streams for real-world sensor networks.

## 1.3 Overview of the proposed method

We briefly summarize the batch-based tensor factorization approach to remove outliers and impute missing data, then give an overview how we adapt it to online settings.

The sensor observation data is organized in a tensor [13, 14], to exploit the spatial and temporal structures in the data. An example of a three-way tensor storing sensor data is shown in the first

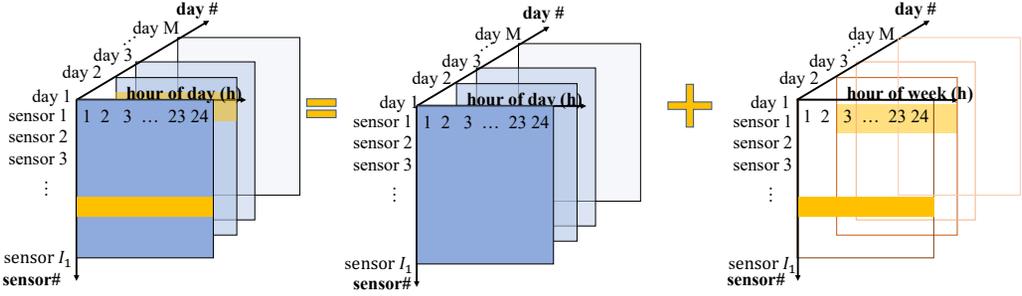


Fig. 2. Observation data decomposed into low rank tensor for clean data and fiber-sparse tensor for outlier data caused by malfunctioning sensors.

column of Fig. 2, where the first mode corresponds to each sensor, the second mode to each hour in a 24-hour period, and the third mode to each 24-hour period in the dataset. Tensor factorization approaches [13, 14] exploit the fact that such large, noisy, and incomplete datasets actually have low intrinsic dimensionality. Furthermore we assume that the outliers in the sensor network have a specific sparsity pattern, persisting across one of the orders of the tensor, as shown in the third column of Fig. 2. This outlier structure reflects the observation that some sensors degrade and produce faulty data for extended periods of time.

To reconstruct the underlying clean complete data and detect the outliers, we solve the following optimization problem for  $M$  days of data:

$$\begin{aligned} \min_{\mathcal{X}^M, \mathcal{E}^M} \quad & \text{rank}(\mathcal{X}^M) + \lambda \text{ sparsity}(\mathcal{E}^M) \\ \text{s.t.} \quad & \mathcal{B}^M_{i_1 i_2 \dots i_N} = (\mathcal{X}^M + \mathcal{E}^M)_{i_1 i_2 \dots i_N}, \\ & \text{where } (i_1, i_2, \dots, i_N) \text{ is an observed entry,} \end{aligned} \quad (1)$$

where tensor  $\mathcal{X}^M$  represents the clean complete data, tensor  $\mathcal{E}^M$  denotes the outliers, and  $\mathcal{B}^M$  denotes the observation data. The size of  $\mathcal{X}^M$ ,  $\mathcal{E}^M$  and  $\mathcal{B}^M$  grows with the number of days  $M$ . We regularize the low dimensionality of  $\mathcal{X}^M$  measured by the Tucker rank [15], and the sparsity of the outlier tensor  $\mathcal{E}^M$ , under the constraint that  $\mathcal{X}^M$  and  $\mathcal{E}^M$  adds up to the raw data  $\mathcal{B}^M$  in the observed entries.

In the batch-based approach [10], Problem (1) is solved by singular value thresholding [16, 17] based on the *alternating direction method of multipliers* (ADMM) framework [10, 14] in an iterative manner. However, singular value thresholding can only be computed after all samples are collected. This practically means that after we conduct the tensor decomposition at day  $M$ , when new data comes in on day  $M + 1$ , we have to solve Problem (1) from scratch. It is inefficient that we cannot reuse the results from earlier computations, and that we have to store all observation data in memory, which can grow large quickly. Moreover, in each iteration of solving (1), we need to compute a singular value decomposition, which is computationally expensive especially as the tensor size grows. Thus, the batch-based method is hardly scalable to large streaming systems.

To enable online processing and compute the data in a sequential manner, we develop OLRTR, which regularizes the rank of tensor  $\mathcal{X}$  in a new way. Namely, we keep a small dictionary forming a basis for the low-rank subspace, and find the corresponding coefficients to represent  $\mathcal{X}$  in terms of the basis. The size of the dictionary gives an upper bound on the rank of  $\mathcal{X}$ , and we aim to find the dictionary to best capture all the samples. In practice, we update the dictionary after each sample estimation so that the dictionary can also adapt to the shifting dynamics of the underlying

subspace. In this way, the computation for each sample is decoupled. Furthermore, we only need a small space to store the dictionary, and there is no need to store all observation data in memory. Thus, our approach enables online recovery for large-scale sensor networks.

The remainder of this article is as follows. Section 2 reviews the most related works. Section 3 introduces the basic tensor notations, and reviews batch-based tensor robust decomposition. Section 4 develops our proposed OLRTR method. Section 5 shows our experiments on both synthetic and real-world datasets. We finally conclude the work in Section 6.

## 2 RELATED WORK

### 2.1 Other data preprocessing efforts

Data preprocessing is fundamental to building a reliable and comprehensive understanding of the analysis tasks afterwards. In general, data preprocessing techniques can be categorized into three aspects [18]: (1) *data transformation*, such as data filtering and noise modeling, (2) *information gathering* such as visualization and feature extraction, and (3) *generation of new information* such as time series analysis, data fusion and simulation/creation of new features. These treatments help to solve the problems that hinders further analysis and provide us with meaningful understanding of the measurements.

### 2.2 Batch-based low-rank learning

Low rank learning has been widely used to exploit the correlations in the datasets, with application in video surveillance [17], link prediction [19], anomaly detection [20], and so on.

Two threads of works are most related with ours, namely robust matrix and tensors decomposition with gross corruption, and low rank matrix and tensor completion with missing data. For robust decomposition,  $l_1$  norm is usually used as a convex regularization for element-wise sparsity [14]. Cauchy distribution and the chi-squared distribution is also used to deal with the case when gross corruption and small noises co-exists [21, 22]. If the outlier is structured, for example grouped in columns, then  $l_{2,1}$  norm regularization is usually used [10, 23, 24]. Regarding missing data imputation in tensors, CANDECOMP/PARAFAC (CP) decomposition [25] is used in the works [26, 27], and Tucker decomposition [15] is used in the works [10, 28].

However, the above mentioned methods are all batch-based, requiring all samples be collected before the low rank decomposition can be performed. This does not meet our need in sensor networks where we need the estimation to be performed continuously as new data comes in. Moreover, the memory and computation requirement of batch methods poses challenge for large-scale sensor networks.

### 2.3 Online low-rank learning

To adapt the low-rank learning to online settings, various attempts in matrix and tensor fields have been made. For matrix decomposition, Mairal et al. [29] develops an online dictionary learning method based on the assumption of sparse coding, i.e. the data vectors are linear combinations of a few of the basis vectors. Inspired by [29], Feng et al. [30] proposes matrix online robust PCA via stochastic optimization, which is provably robust to sparse corruption. Shen et al. [31] develops a *Low-Rank Representation* (LRR) based online algorithm that segments data generated from a union of subspaces with improved time complexity and memory footprint. He et al. [32] tracks the subspaces by gradient descent on Grassmannian, the manifold of all  $d$ -dimensional subspaces. For tensor decomposition, Sobral et al. [33] proposes an *online stochastic framework for tensor decomposition* (OSTD) for video sequence background subtraction. Li et al. [34] develops an *online robust low-rank tensor modeling* (ORLTM) method that can deal with streaming tensor data drawn from a mixture of

multiple subspaces effectively through dictionary learning. Our work is most inspired by [30], but we extend the decomposition from matrix to tensor to exploit the multi-dimensional correlations, and we also adapt to the scenario of structured outlier and missing data.

There are two major differences between our method and the methods mentioned above. First, we aim to find structured outliers grouped in tensor fibers, while the approaches [30–34] only deals with unstructured element-wise outliers. While there are online decomposition approaches to find column outliers [35], they deal with the case when an entire sample of data vector is an outlier. In comparison, in our case the structured outliers lies across multiple samples. For example, in sensor networks we aim at finding out malfunctioning sensors producing wrong recordings for a consecutive time, while approaches [35] can only find out abnormal time slots when all sensors deviate from normal recording. We solve this problem by conducting decomposition on minibatches instead of single samples. Second, we are able to deal with the case when outliers and missing data co-exists, whereas the approaches [30, 31, 33, 34] only considers outliers with full observations. Several online robust decomposition approaches handle missing data, including [32] via gradient descent on Grassmannian, and Kasai et al. [36] based on the Candecomp/PARAFAC (CP) decomposition, yet they both deal with element-wise outliers.

### 3 PRELIMINARIES

#### 3.1 Tensor basics

We briefly introduce our notation and define tensor operators, following a standard notation [10, 13, 14], (See also [13, 14] for a detailed discussion).

A tensor is denoted by an Euler script letter (e.g.,  $\mathcal{X}$ ); and a matrix by a boldface capital letter (e.g.,  $\mathbf{X}$ ); a vector by a boldface lowercase letter (e.g.,  $\mathbf{x}$ ); and a scalar by a lowercase letter (e.g.,  $x$ ). A tensor of order  $N$  has  $N$  dimensions. A *fiber* is a column vector formed by fixing all indices of a tensor but one.

The *unfolding* function flattens the tensor into a matrix to facilitate the computation. The unfolding of a tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  in the  $n^{\text{th}}$  mode is formed by rearranging the mode- $n$  fibers as its columns, resulting in a matrix  $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_{N \setminus n}}$ , where  $I_{N \setminus n} = \prod_{i \neq n, i=1}^N I_i = I_1 \times \dots \times I_{i-1} \times I_{i+1} \times \dots \times I_N$ . To convert the unfolding matrix back to original tensor, the *fold* function is applied:  $\text{fold}_n(\mathbf{X}_{(n)}) = \mathcal{X}$ .

The inner product of  $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  is the sum of their element-wise product:  $\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \dots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \dots i_N} y_{i_1 i_2 \dots i_N}$ , where  $x_{i_1 i_2 \dots i_N}$  and  $y_{i_1 i_2 \dots i_N}$  denote the  $(i_1, i_2, \dots, i_N)$  element of  $\mathcal{X}$  and  $\mathcal{Y}$  respectively.

The *tensor Frobenius norm* follows naturally from the matrix Frobenius norm, and is defined as:  $\|\mathcal{X}\|_F = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}$ .

The mode- $n$  product of a tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  and a matrix  $\mathbf{A} \in \mathbb{R}^{J \times I_n}$  is denoted by  $\mathcal{X} \times_n \mathbf{A} = \mathcal{Y}$ , where  $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_{n-1} \times J \times I_{n+1} \times \dots \times I_N}$ .

The *Tucker decomposition* [13, 14] approximates a tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  as a core tensor  $\mathcal{G} \in \mathbb{R}^{c_1 \times c_2 \times \dots \times c_N}$  multiplied in each mode  $n$  by an appropriately sized matrix  $\mathbf{U}^{(n)}$ :  $\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times \dots \times_N \mathbf{U}^{(N)}$ . The matrices  $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times c_n}$  are factor matrices, which are usually assumed to be orthogonal.

#### 3.2 Robust tensor decomposition

In this section, we briefly summarize the batch-based higher-order tensor decomposition with fiber-wise corruption as posed in [10], which we adapt to online settings in Section 4.

The batch-based setup with complete observation is as follows. We are given a high dimensional data tensor  $\mathcal{B}$  that is corrupted in a few fibers. In other words, we have  $\mathcal{B} = \mathcal{X} + \mathcal{E}$ , where  $\mathcal{X}$  is

the low rank tensor, and  $\mathcal{E}$  is the sparse fiber outlier tensor.  $\mathcal{B}, \mathcal{X}, \mathcal{E} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ . Our goal is to reconstruct  $\mathcal{X}$  on the non-corrupted fibers, as well as identify the outlier location. Without loss of generality, we assume the fiber-wise corruption occurs along the first mode. The optimization problem goes as follows:

$$\begin{aligned} \min_{\mathcal{X}_i, \mathcal{E}} \quad & \sum_{i=1}^N \|\mathbf{X}_{i(i)}\|_* + \lambda \|\mathbf{E}_{(1)}\|_{2,1} \\ \text{s.t.} \quad & \mathcal{B} = \mathcal{X}_i + \mathcal{E}, i = 1, 2, \dots, N, \end{aligned} \quad (2)$$

where  $\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_N \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  are auxiliary variables split from the same variable  $\mathcal{X}$  to decouple the computation in different tensor modes, and  $\mathbf{X}_{i(i)}$  is the corresponding mode- $i$  unfolding for  $\mathcal{X}_i$ .  $\mathbf{E}_1$  is the unfolding of  $\mathcal{E}$  in the first mode. The  $N$  constraints  $\mathcal{B} = \mathcal{X}_i + \mathcal{E}$  ensure that  $\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_N$  are all equal to the original low rank tensor  $\mathcal{X}$ . The sum of nuclear norms  $\sum_i \|\mathbf{X}_{i(i)}\|_*$  is a convex relaxation for Tucker rank of  $\mathcal{X}$  [14], with the nuclear norm computed as  $\|\mathbf{X}\|_* := \sum_i \sigma_i$ , where  $\sigma_i$  denotes the  $i$ -th singular value of  $\mathbf{X}$ . The  $l_{2,1}$  norm of a matrix  $\mathbf{E} \in \mathbb{R}^{I_1 \times I_2}$  is used to encourage the column-wise sparsity, defined as  $\|\mathbf{E}\|_{2,1} = \sum_{j=1}^{I_2} \sqrt{\sum_{i=1}^{I_1} (e_{ij})^2}$ .

We note that the mode along which to unfold  $\mathcal{E}$  in (2) depends on what kind of outliers we want to detect. For example, in sensor network data as shown in Fig. 2, unfolding  $\mathcal{E}$  along first mode corresponds to abnormal hours when records from all sensors deviates from normal; second mode corresponds to abnormal sensors that records wrong value for a consecutive period— which is our goal in this paper. To simplify the notations, the rest of the paper is based on unfolding along the first mode, but the method for other modes follows the same line.

In addition to observation data being grossly corrupted, we might have only partial observations of  $\mathcal{B}$ , and we seek to complete the decomposition nevertheless. In this case, we force the decomposition to match the observation data only at the available entries. This is done by introducing a compensation tensor  $\mathcal{O} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ , which is zero for entries in the observation set  $\Omega \subset [I_1] \times [I_2] \times \dots \times [I_N]$ , and can take any value outside  $\Omega$ . Thus using the same auxiliary variables technique as in (2), the problem is formulated as

$$\begin{aligned} \min_{\mathcal{X}_i, \mathcal{E}} \quad & \sum_{i=1}^N \|\mathbf{X}_{i(i)}\|_* + \lambda \|\mathbf{E}_{(1)}\|_{2,1} \\ \text{s.t.} \quad & \mathcal{B} = \mathcal{X}_i + \mathcal{E} + \mathcal{O}, i = 1, 2, \dots, N, \\ & \mathcal{O}_\Omega = 0, \end{aligned} \quad (3)$$

where  $\mathcal{O}_\Omega$  denotes the entries in  $\mathcal{O}$  that are observed. Since  $\mathcal{O}$  compensates for whatever the value is in the unobserved entries of  $\mathcal{B}$ , we only need to keep track of the indices of the unobserved entries, and can simply set the unobserved entries of  $\mathcal{B}$  to zero.

Problem (2) and (3) are usually solved via *Alternating direction method of multipliers* (ADMM) methods or Accelerated Proximal Gradient (APG) methods in an iterative manner [10, 14]. However, in each iteration, to optimize the term  $\|\mathbf{X}_{i(i)}\|_*$  we need to compute the *singular value decomposition* (SVD) of  $\mathbf{X}_{i(i)}$  composed of all samples. This both limits the ability for stream processing of the data, and is computationally expensive. In the next section, we develop an online algorithm to address these limitations.

## 4 METHODS

In this section, we first pose the online objective function for online higher-order tensor decomposition problem in the presence of fiber outliers, and then provide an efficient algorithm to solve

it in the streaming settings. The online algorithm under partial-observation settings follows the same line as the full observation case. We provide the formulation and the algorithm for partial observations in the Appendix.

#### 4.1 Problem formulation

We now develop the objective function for online setting, starting with the batch-based objective function (2). We note that in streaming data settings, for tensors  $\mathcal{E}, \mathcal{B}, \mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ , the size of the last dimension  $I_N$  grows with time, but we drop the explicit dependency of time in the notation. As we have seen, the major challenge in a batch-based method (2) is that the nuclear norm term  $\|\mathbf{X}_{i(i)}\|_*$  keeps all samples tightly coupled. In this section we show how we substitute the nuclear norm term with an equivalent form, based on which we can derive an empirical cost function that separates out the loss incurred by each sample, thus allowing computation sequentially in time.

First, in order to facilitate data online processing, we relax the decomposition constraint in (2) into a Frobenius norm penalty in the objective function. Thus, (2) becomes

$$\min_{\mathcal{X}_1, \dots, \mathcal{X}_N, \mathcal{E}} \frac{1}{2} \sum_{i=1}^N \|\mathcal{X}_i + \mathcal{E} - \mathcal{B}\|_F^2 + \lambda_1 \sum_{i=1}^N \|\mathbf{X}_{i(i)}\|_* + \lambda_2 \|\mathbf{E}_{(1)}\|_{2,1}, \quad (4)$$

where  $\lambda_1$  and  $\lambda_2$  are weight parameters balancing the costs of low rank and sparsity respectively.

Next, we deal with the the nuclear norm term  $\|\mathbf{X}_{i(i)}\|_*$  that couples all samples and prohibits processing the data sequentially. Namely, for each mode  $i = 1, 2, \dots, N$ , we substitute  $\|\mathbf{X}_{i(i)}\|_*$  with an equivalent form:

$$\begin{aligned} \|\mathbf{X}_{i(i)}\|_* &= \inf_{\mathbf{L}_i, \mathbf{R}_i} \left\{ \frac{1}{2} \|\mathbf{L}_i\|_F^2 + \frac{1}{2} \|\mathbf{R}_i\|_F^2 \right\} \\ \text{s.t.} \quad &\mathbf{X}_{i(i)} = \mathbf{L}_i \mathbf{R}_i^T. \end{aligned} \quad (5)$$

where we explicitly factorize  $\mathbf{X}_{i(i)}$  into  $\mathbf{L}_i \in \mathbb{R}^{I_i \times r}$ ,  $\mathbf{R}_i \in \mathbb{R}^{I_N \setminus i \times r}$ , and upper bound the rank of  $\mathbf{X}_{i(i)}$  by  $r$ , with  $r \ll \min(I_i, I_N \setminus i)$ .  $\mathbf{L}_i$  can be seen as a dictionary, where each column represents a basis vector in the mode- $i$  unfolding, and  $\mathbf{R}_i$  are the corresponding coefficients of the basis for the samples. Such nuclear norm substitution (5) is well established in works including [37, 38]. In this way, Problem (4) becomes:

$$\begin{aligned} \min_{\mathcal{X}_i, \mathcal{E}} \quad &\frac{1}{2} \sum_{i=1}^N \|\mathcal{X}_i + \mathcal{E} - \mathcal{B}\|_F^2 + \frac{\lambda_1}{2} \sum_{i=1}^N (\|\mathbf{L}_i\|_F^2 + \|\mathbf{R}_i\|_F^2) + \lambda_2 \|\mathbf{E}_{(1)}\|_{2,1} \\ \text{s.t.} \quad &\mathbf{X}_{i(i)} = \mathbf{L}_i \mathbf{R}_i^T, i = 1, 2, \dots, N. \end{aligned} \quad (6)$$

Substitution of  $\mathbf{X}_{i(i)}$  with  $\mathbf{L}_i$  and  $\mathbf{R}_i$  and removing the constraint in (6), we arrive at:

$$\min_{\mathbf{L}_i, \mathbf{R}_i, \mathcal{E}} \frac{1}{2} \sum_{i=1}^N \|\mathbf{L}_i \mathbf{R}_i^T + \mathbf{E}_{(i)} - \mathbf{B}_{(i)}\|_F^2 + \frac{\lambda_1}{2} \sum_{i=1}^N (\|\mathbf{L}_i\|_F^2 + \|\mathbf{R}_i\|_F^2) + \lambda_2 \|\mathbf{E}_{(1)}\|_{2,1}. \quad (7)$$

For the first term in (7), we change the Frobenius norm of a tensor in (6) into the Frobenius norm of its mode- $i$  unfolding, which does not change the value of the norm. Problem (7) is not jointly convex in terms of  $\mathbf{L}_i$  and  $\mathbf{R}_i$ , but a locally minimizing solution for (7) provides a good global solution for the original problem (2), as theoretically proven in [30] in the matrix case, and empirically shown in tensor cases in Section 5.

In online settings, we divide the overall observation tensor into a series of minibatches along the last dimension,  $\mathcal{B} = [\mathcal{B}^1, \mathcal{B}^2, \dots, \mathcal{B}^M]$ ,  $\mathcal{B}^t \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_{N-1} \times I'_N}$ , where  $I'_N = \lfloor I_N/M \rfloor$  is the size of last dimension for each sample. In sensor networks, this amounts to processing the data every day when  $I'_N = 1$ , or every few days when  $I'_N > 1$ .

Given the series of samples  $\mathcal{B}^t$ , solving problem (7) amounts to minimizing the following empirical objective function:

$$f_M(\mathbf{L}_i) \triangleq \frac{1}{M} \sum_{t=1}^M \sum_{i=1}^N \left[ l(\mathcal{B}^t, \mathbf{L}_i) + \frac{\lambda_1}{2M} \|\mathbf{L}_i\|_F^2 \right], \quad (8)$$

where the loss function for each mini-batch is defined as

$$l(\mathcal{B}^t, \mathbf{L}_i) = \min_{\mathbf{R}_i^t, \mathcal{E}^t} \frac{1}{2} \|\mathbf{L}_i \mathbf{R}_i^{tT} + \mathbf{E}_{(i)}^t - \mathbf{B}_{(i)}^t\|_F^2 + \frac{\lambda_1}{2} \|\mathbf{R}_i^t\|_F^2 + \lambda_2 \|\mathbf{E}_{(i)}^t\|_{2,1}. \quad (9)$$

In minimizing (8), we want to find the set of basis  $\mathbf{L}_i$  that works across all samples, and that minimizes the accumulated loss of all samples. The loss for each sample under a fixed basis  $\mathbf{L}_i$  is calculated via (9), where we find the optimal coefficients  $\mathbf{R}_i^t$  and the outlier tensor  $\mathcal{E}^t$  for each sample to minimize the loss given the basis  $\mathbf{L}_i$ .

#### 4.2 Online tensor RPCA algorithm

In this section, we develop OLRTR to efficiently solve Problem (8) online, taking one mini-batch at a time. The OLRTR algorithm is summarized in Algorithm 1. In an overview, we take an alternative optimization approach to optimize  $\mathbf{R}_i^t$ ,  $\mathcal{E}^t$  and  $\mathbf{L}_i$ . Namely, at the  $t$ -th time step, after accessing the new sample  $\mathcal{B}^t$ , we first solve for the corresponding  $\mathbf{R}_i^t$  and  $\mathcal{E}^t$ , using the  $\mathbf{L}_i$  obtained in last step  $t-1$ . Then we update  $\mathbf{L}_i$ , to minimize the accumulated loss given all  $\{\mathbf{R}_i^\tau\}_{\tau=1}^t$  and  $\{\mathcal{E}^\tau\}_{\tau=1}^t$  obtained so far.

Specifically, at time step  $t$ , having accessed the minibatch  $\mathcal{B}^t$ , we first address (9) and solve for  $\mathbf{R}_i^t$  and  $\mathcal{E}^t$  with  $\mathbf{L}_i$  fixed. To this end, we alternatively update  $\mathbf{R}_i^t$  and  $\mathcal{E}^t$ . When  $\mathcal{E}^t$  is fixed, we solve  $\mathbf{R}_i^t$  via:

$$\mathbf{R}_i^t = \underset{\mathbf{R}_i}{\operatorname{argmin}} \frac{\lambda_1}{2} \|\mathbf{R}_i\|_F^2 + \frac{1}{2} \sum_{i=1}^N \|\mathbf{L}_i \mathbf{R}_i^T + \mathbf{E}_{(i)}^t - \mathbf{B}_{(i)}^t\|_F^2, \quad (10)$$

which has a closed-form solution

$$\mathbf{R}_i^t = \left( \mathbf{L}_i^T \mathbf{L}_i + \lambda_1 \mathbf{I} \right)^{-1} \mathbf{L}_i^T \left( \mathbf{B}_{(i)} - \mathbf{E}_{(i)} \right). \quad (11)$$

Then, when  $\mathbf{R}_i^t$  is fixed, we solve  $\mathcal{E}^t$  via

$$\begin{aligned} \mathcal{E}^t &= \underset{\mathcal{E}}{\operatorname{argmin}} \lambda_2 \|\mathbf{E}_{(1)}^t\|_{2,1} + \frac{1}{2} \sum_{i=1}^N \|\mathbf{E}_{(i)} + \mathbf{L}_i \mathbf{R}_i^{tT} - \mathbf{B}_{(i)}^t\|_F^2 \\ &= \underset{\mathcal{E}}{\operatorname{argmin}} \lambda_2 \|\mathbf{E}_{(1)}\|_{2,1} + \frac{1}{2} \sum_{i=1}^N \|\mathcal{E} + \operatorname{fold}_i(\mathbf{L}_i \mathbf{R}_i^{tT}) - \mathcal{B}^t\|_F^2 \end{aligned} \quad (12)$$

Following the same approach as [10, 14], problem (12) shares the same solution as

$$\mathcal{E}^t = \underset{\mathcal{E}}{\operatorname{argmin}} \lambda_2 \|\mathbf{E}_{(1)}\|_{2,1} + \frac{N}{2} \left\| \mathcal{E} - \frac{1}{N} \sum_{i=1}^N \left( \mathcal{B}^t - \operatorname{fold}_i(\mathbf{L}_i \mathbf{R}_i^{tT}) \right) \right\|_F^2. \quad (13)$$

Denoting the term  $(\mathcal{B}^t - \operatorname{fold}_i(\mathbf{L}_i \mathbf{R}_i^{tT}))$  as  $\mathbf{C}$ , then following the approach in [10], the closed form solution for (12) is:

$$\mathbf{E}_{(1)j}^t = \mathbf{C}_{(1)j} \max \left\{ 0, 1 - \frac{\lambda_2}{\mu N \|\mathbf{C}_{(1)j}\|_2} \right\}, \text{ for } j = 1, 2, \dots, p, \quad (14)$$

where  $\mathbf{E}_{(1)j}$  is the  $j^{\text{th}}$  column of  $\mathbf{E}_{(1)}$ ,  $\mathbf{C}_{(1)j}$  is the  $j^{\text{th}}$  column of  $\mathbf{C}_{(1)}$ , and  $p = I_2 \times I_3 \times \dots \times I'_N$  is the total number of columns in  $\mathbf{C}_{(1)}$ .

The sample update for  $\mathbf{R}_i$  and  $\mathcal{E}$  is summarized in Algorithm 2. The algorithm convergence criterion is met when the change between iterations is small enough, as measured by the Frobenius norm of the difference in  $\mathbf{R}_i$  and  $\mathcal{E}$  between iterations i.e.,

$$\max \left( \frac{\|\mathbf{R}_i^{(k-1)} - \mathbf{R}_i^{(k)}\|_F}{\|\mathcal{B}^t\|_F}, \frac{\|\mathcal{E}^{(k-1)} - \mathcal{E}^{(k)}\|_F}{\|\mathcal{B}^t\|_F} \right) \leq \epsilon, \quad (15)$$

where  $(k)$  denotes the iteration number, and  $\epsilon$  is the tolerance.

Next, given the estimated coefficient  $\mathbf{R}_i^t$  and outlier tensor  $\mathcal{E}^t$ , we update the dictionary  $\mathbf{L}_i^t$ . We define the objective function for updating  $\mathbf{L}_i^t$  as

$$g_t(\mathbf{L}_i) \triangleq \frac{1}{t} \sum_{i=1}^t \left( \left\| \mathbf{L}_i \mathbf{R}_i^{tT} + \mathbf{E}_{(i)}^t - \mathbf{B}_{(i)}^t \right\|_F^2 + \frac{\lambda_1}{2} \|\mathbf{R}_i^t\|_F^2 + \lambda_2 \|\mathbf{E}_{(1)}^t\|_{2,1} + \frac{\lambda_1}{2t} \|\mathbf{L}_i\|_F^2 \right), \quad (16)$$

and we aim to solve

$$\mathbf{L}_i^t = \underset{\mathbf{L}_i}{\operatorname{argmin}} g_t(\mathbf{L}_i). \quad (17)$$

$g_t(\mathbf{L}_i)$  is a surrogate function for  $f_N(\mathbf{L}_i)$  in (8), and provides an upper bound for  $f_N(\mathbf{L}_i)$ . Using the relationship between the Frobenius norm and the matrix trace,  $\|\mathbf{Y}\|_F^2 = \operatorname{Tr}(\mathbf{Y}^T \mathbf{Y})$ , and the properties of matrix trace computation, Problem (17) can be transformed into

$$\mathbf{L}_i^t = \underset{\mathbf{L}_i}{\operatorname{argmin}} \frac{1}{2} \operatorname{Tr} \left( \mathbf{L}_i^T (\mathbf{A}_i^t + \lambda_1 \mathbf{I}) \mathbf{L}_i \right) - \operatorname{Tr} \left( \mathbf{L}_i^T \mathbf{D}_i^t \right), \quad (18)$$

where the two sets of accumulation matrices  $\mathbf{A}_i^t \in \mathbb{R}^{r \times r}$  and  $\mathbf{D}_i^t \in \mathbb{R}^{I_i \times r}$  are defined as

$$\begin{aligned} \mathbf{A}_i^t &= \sum_{i=1}^t \mathbf{R}_i^{tT} \mathbf{R}_i^t, \\ \mathbf{D}_i^t &= \sum_{i=1}^t \left( \mathbf{B}_{(i)}^t - \mathbf{E}_{(i)}^t \right) \mathbf{R}_i^t. \end{aligned}$$

To solve Problem (18), we adopt a block-coordinate decent approach similar to [29, 30], updating one column of  $\mathbf{L}_i$  at a time with the rest columns fixed. At each step  $t$ , we use the solution for the previous step,  $\mathbf{L}_i^{t-1}$  as warm restart. The algorithm is provided in Algorithm 3. In online computation, we store the value of  $\mathbf{A}_i^t$  and  $\mathbf{D}_i^t$  to accumulate the information in all samples, whose sizes does not change with the number of samples, thus enabling the scalability of the online algorithm.

We initialize each entry of the dictionary  $\mathbf{L}_i \in \mathbb{R}^{I_i \times r}$  from an i.i.d. uniform distribution  $\mathcal{U}(0, 1)$ , and initialize the accumulation matrices  $\mathbf{A}_i^t \in \mathbb{R}^{r \times r}$ ,  $\mathbf{D}_i^t \in \mathbb{R}^{I_i \times r}$  to zero.

### 4.3 Complexity and memory cost

The overall complexity for Algorithm 1 depends on the minibatch size. Namely, denoting the overall size for a minibatch  $\mathcal{B}^t \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_{N-1} \times I_N}$  as  $s = \prod_{i=1}^N I_i = I_1 \times I_2 \times \dots \times I_{N-1} \times I_N$ , then the computational complexity is  $O(rs)$ . To see this, we note that the complexity for line 4 in Algorithm 1 is  $O(r^2 I_i + r^3 + rs)$  using Algorithm 2, where the  $O(r^3)$  comes from the matrix inverse, and  $O(r^2 I_i + rs)$  comes from matrix multiplications. The complexity for line 6 is  $O(r^2 I_{N \setminus i} + rs)$ . The complexity for line 7 is  $O(r^2 I_i)$  with Algorithm 3, since to update each column of  $\mathbf{L}_i$  takes  $O(r I_i)$ , and there are  $r$  columns in total. Since  $r \ll I_i < s$ , the overall complexity is thus  $O(rs)$ , which is linear with the minibatch size, and is relatively small with a reasonable minibatch size.

The memory cost for Algorithm 1 in each iteration is  $O(s)$ , dominated by loading the minibatch data  $\mathcal{B}^t$  and estimating  $\mathcal{X}^t$  and  $\mathcal{E}^t$ . The historical information has been stored in  $\mathbf{A}^t$  and  $\mathbf{D}^t$ , at

**Algorithm 1** OLRTR algorithm

- 
- 1: Given  $N$ -way minibatch tensors  $[\mathcal{B}^1, \mathcal{B}^2, \dots, \mathcal{B}^M]$  with  $\mathcal{B}^t \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_{N-1} \times I'_N}$ , weighting parameters  $\lambda_1, \lambda_2$ , target rank  $r$ . Initialize dictionary  $\mathbf{L}_i \in \mathbb{R}^{I_i \times r}$  and accumulation matrices  $\mathbf{A}_i^t \in \mathbb{R}^{r \times r}, \mathbf{D}_i^t \in \mathbb{R}^{I_i \times r}$ .
  - 2: **for**  $t = 0, 1, \dots, M$  **do**
  - 3:     Access the  $t$ -th sample  $\mathcal{B}^t$
  - 4:     Solve  $\mathbf{R}_i^t$  and  $\mathcal{E}^t$  for the new sample using Algorithm 2.

$$\{\mathbf{R}_i^t, \mathcal{E}^t\} = \underset{\mathbf{R}_i^t, \mathcal{E}^t}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^N \|\mathbf{L}_i \mathbf{R}_i^{tT} + \mathbf{E}_{(i)}^t - \mathbf{B}_{(i)}^t\|_F^2 + \frac{\lambda_1}{2} \sum_{i=1}^N \|\mathbf{R}_i^t\|_F^2 + \lambda_2 \|\mathbf{E}_{(1)}^t\|_{2,1}.$$

- 5:      $\mathcal{X}^t = \frac{1}{N} \operatorname{fold}_i(\mathbf{L}_i \mathbf{R}_i^t)$
- 6:      $\mathbf{A}_i^t = \mathbf{A}_i^{t-1} + \mathbf{R}_i^{tT} \mathbf{R}_i^t; \mathbf{D}_i^t = \mathbf{D}_i^{t-1} + (\mathbf{B}_{(i)}^t - \mathbf{E}_{(i)}^t) \mathbf{R}_i^t$
- 7:     Solve  $\mathbf{L}_i^t$  with Algorithm 3, warm started with  $\mathbf{L}_i^{t-1}$

$$\mathbf{L}_i^t = \underset{\mathbf{L}_i}{\operatorname{argmin}} \frac{1}{2} \operatorname{Tr}(\mathbf{L}_i^T (\mathbf{A}_i^t + \lambda_1 \mathbf{I}) \mathbf{L}_i) - \operatorname{Tr}(\mathbf{L}_i^T \mathbf{D}_i^t)$$

- 8: **end for**
  - 9: **return** low rank tensors  $[\mathcal{X}^1, \mathcal{X}^2, \dots, \mathcal{X}^M]$  and outlier tensors  $[\mathcal{E}^1, \mathcal{E}^2, \dots, \mathcal{E}^M]$
- 

**Algorithm 2** Sample update for  $\mathbf{R}_i$  and  $\mathcal{E}$ 

- 
- 1: Given observation  $\mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_{N-1} \times I'_N}$ , basis  $\mathbf{L}_i \in \mathbb{R}^{I_i \times r}$  and parameters  $\lambda_1, \lambda_2$ . Initialize coefficient matrix  $\mathbf{R}_i \in \mathbb{R}^{I'_N \times r}$  and outlier tensor  $\mathcal{E} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_{N-1} \times I'_N}$  to zero
  - 2: **while** not converged **do**
  - 3:      $\mathbf{C} \leftarrow (\mathcal{B} - \operatorname{fold}_i(\mathbf{L}_i \mathbf{R}_i^T))$  ▷ Update  $\mathcal{E}$ .
  - 4:     **for**  $j = 1, 2, \dots, p$  **do**
  - 5:          $\mathbf{E}_{(1)j} \leftarrow \mathbf{C}_{(1)j} \max\left\{0, 1 - \frac{\lambda}{\mu N \|\mathbf{C}_{(1)j}\|_2}\right\}$
  - 6:     **end for**
  - 7:     **for**  $i = 1, 2, \dots, N$  **do** ▷ Update  $\mathbf{R}_i$
  - 8:          $\mathbf{R}_i \leftarrow (\mathbf{L}_i^T \mathbf{L}_i + \lambda_1 \mathbf{I})^{-1} \mathbf{L}_i^T (\mathbf{B}_{(i)} - \mathbf{E}_{(i)})$ .
  - 9:     **end for**
  - 10: **end while**
  - 11: **return**  $\mathbf{R}_i$  and  $\mathcal{E}$
- 

a memory cost of  $O(rI_i)$ . Thus, the memory cost does not increase with the number of samples, meeting the need for large-scale long term monitoring systems.

## 5 EXPERIMENTS

In this section, we conduct experiments on both simulation data and real world sensor network data. We first examine the performance of tensor recovery and anomaly detection on synthetically generated low-rank tensors with known fiber-wise corruptions and random missing entries. Then we apply the algorithm on two large sensor network datasets. The first experiment is a complete

**Algorithm 3** Basis  $L_i$  update

---

```

1: Given  $L_i = [l_{i_1}, \dots, l_{i_r}] \in \mathbb{R}^{I_N \times r}$ ,  $A_i = [a_{i_1}, \dots, a_{i_r}] \in \mathbb{R}^{r \times r}$ ,  $D_i = [d_{i_1}, \dots, d_{i_r}] \in \mathbb{R}^{I_i \times r}$ .
   Let  $\tilde{A}_i = A_i + \lambda_1 I$ .
2: for  $j = 0, 1, \dots, r$  do
3:    $l_{i_j} \leftarrow \frac{1}{\tilde{A}_{i,j,j}} (d_{i_j} - L_i a_{i_j}) + l_{i_j}$ 
4: end for
5: return  $L_i$ 

```

---

NOAA [11] temperature dataset that we synthetically degrade, so that the recovery relative error can be computed. In the second sensor network experiment, we apply the method to Array of Things temperature data which contains missing data and outliers. We assess the quality of the recovery by comparing the correlation of AoT data with NOAA sensors when they are in close proximity.

We compare our method with online tensor approaches including ORLTM [34], OSTD [33], online matrix approaches OLRSC [31], STOC-RPCA [30], GRASTA [32], as well as batch tensor approaches TRPCA [39], and RTR [10].

The performance of the algorithms are measured by the relative error of the low rank tensor, as well as the F1 score of the outlier fibers. The *relative error* (RE) of low rank tensor is calculated as:

$$\text{RE} = \frac{\|\mathcal{X}_{\text{all}} - \hat{\mathcal{X}}_{\text{all}}\|_F}{\|\mathcal{X}_{\text{all}}\|_F}, \quad (19)$$

where  $\|\cdot\|_F$  is the tensor Frobenius norm, and  $\hat{\mathcal{X}}_{\text{all}} \in \mathbb{R}^{I_1 \times I_2 \times nI_3}$  is the estimated low rank tensor, which has the value 0 in the fibers that are estimated to be corrupted. For online methods,  $\hat{\mathcal{X}}_{\text{all}}$  is constructed by concatenating the estimated low rank tensors for all samples along the last dimension. Since most online algorithms are cold started, leading to large losses when first initialized, we discard the first 10 minibatch samples and only compare the performance on the remaining samples.

For our proposed model OLRTR, we set  $\epsilon = 10^{-4}$ , and we use an empirical value  $\lambda_1 = 0.01$ ,  $\lambda_2 = \frac{\alpha}{\sqrt{\log(I_m I_n)}}$  where  $I_m = \max(I_1, \dots, I_N)$ , and  $\alpha$  is a parameter to tune. For other methods, we use the default values as indicated in the original papers, or tuned for best results if the default values doesn't work. The code and data for experiments can be found in [https://github.com/yuehu9/Online\\_Robust\\_Tensor\\_Recovery](https://github.com/yuehu9/Online_Robust_Tensor_Recovery).

## 5.1 Numerical experiments

In this section, we conduct a series of numerical experiments to examine the performance of our method on synthetically generated datasets.

**5.1.1 Simulation setup.** We synthetically generate a series of minibatch observation data  $(\mathcal{B}^1, \mathcal{B}^2, \dots, \mathcal{B}^M)$ , where  $\mathcal{B}^t$  is the  $t$ -th minibatch sample, and is generated as  $\mathcal{B}^t = \mathcal{X}_0^t + \mathcal{E}_0^t \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ . The ground truth data  $\mathcal{X}_0^t$  is generated with a core tensor  $\mathcal{G}^t \in \mathbb{R}^{c_1 \times c_2 \times c_3}$  multiplied in each mode by orthogonal matrices of corresponding dimensions,  $U^{(i)} \in \mathbb{R}^{I_i \times c_i}$ , i.e.,  $\mathcal{X}_0^t = \mathcal{G}^t \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)}$ . The entries of  $\mathcal{G}^t$  are independently sampled from standard Gaussian distribution. The orthogonal matrices  $U^{(i)}$  are generated via a Gram-Schmidt orthogonalization on  $c_i$  vectors of size  $\mathbb{R}^{I_i}$  drawn from standard Gaussian distribution.  $U^{(i)}$  are kept the same across  $M$  minibatch samples, i.e., do not change with  $t$ , so that all minibatches share the same low rank basis. The fiber sparse tensor  $\mathcal{E}_0^t$  is formed by first generating a tensor  $\mathcal{E}_0^{tt} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ , whose entries are i.i.d uniform distribution  $\mathcal{U}(-2,2)$ . Then we randomly keep a fraction  $\gamma$  of the fibers of  $\mathcal{E}_0^{tt}$  to form  $\mathcal{E}_0^t$ . Finally, the corresponding fibers of  $\mathcal{X}_0^t$

with respect to non-zero fibers in  $\mathcal{E}_0^t$  are set to zero. For batch methods, we concatenate all samples  $\mathcal{B}^1, \mathcal{B}^2, \dots, \mathcal{B}^M$  along the last dimension to form the observation  $\mathcal{B}_{\text{all}} \in \mathbb{R}^{I_1 \times I_2 \times MI_3}$  with low rank and sparse components  $\mathcal{X}_{\text{all}}, \mathcal{E}_{\text{all}} \in \mathbb{R}^{I_1 \times I_2 \times MI_3}$ . For matrix-based online methods, we unfold  $\mathcal{B}_{\text{all}}$  in first mode and feed the matrix into the algorithms. In regards of hyper-parameters, we set  $\alpha = 3$  for OLRTR, and set the rank upper bound  $c$  as the true rank for all methods.

**5.1.2 Experiment results.** First, we vary the tensor size, and compare the performance in terms of residual error, F1 score and time. For each experiment, we run 100 minibatches, varying the tensor size of each minibatch  $\mathbb{R}^{I \times I \times I}$ . The rank is set at  $0.1I$  and the gross corruption ratio  $\gamma$  is set at 0.1, and the observation ratio is set at 0.9. The result is shown in Fig. 3. We can see that only batch approach RTR [10] can exactly recover the low rank tensor and find all outliers, yet its computation time increases sharply as input size increases, showing that RTR is not scalable for large online systems. Our algorithm OLRTR works the second best considering RE and F1 score, comparable with the batch method TRPCA [39]. GRASTA [32] performs well in terms of computation time and RE, but has low F1 score, meaning it is lacking in detecting outliers. STOC-RPCA [30] is the matrix counterpart to our method for online settings with element-wise outliers, and our proposed method has about a 0.1 improvement in RE over the STOC-RPCA method. This shows the advantage of our tensor approach in taking full advantage of the correlations in every dimension, to get the best estimate for the low rank tensor.

Next, we vary the fiber corruption ratio and magnitude, and investigate the residual error and F1 score. We run 100 minibatches at each corruption ratio. The low-rank tensor size for each minibatch is fixed at  $\mathbb{R}^{50 \times 50 \times 50}$  with a tucker rank of  $(3, 3, 3)$ , and the observation ratio is set at 1. The result is shown in Fig. 4. We can see that as corruption ratio increases from 0 to 0.5, RE increases for all methods. But the relative error for OLRTR is always under 0.2, second only to the batch methods RTR and TRPCA. The performance of all other online methods drop sharply, with RE above 0.5 for a corruption ratio  $\gamma$  of 0.5. We also vary the corruption magnitude with a fixed corruption ratio  $\gamma = 0.05$ . From the second row of Fig. 4, we see that the relative error is not sensitive to the corruption magnitude. However, the F1 score shows that no method detects outliers if the corruption magnitude is sufficiently small.

Finally, we vary the observation rate. We run 100 minibatches. The low-rank tensor size for each minibatch is fixed at  $\mathbb{R}^{50 \times 50 \times 50}$  with a tucker rank of  $(3, 3, 3)$ , and the corruption ratio is set at 0.05. For methods that cannot handle missing data, we linearly interpolate the missing entries, and also note that filling the missing entries with zero results in similar performance. The result is shown in Fig. 5. We can see that the RE of all methods except RTR drop as observation ratio decreases. GRASTA deals with missing data, and we can see that its performance keeps steady for observation ratios greater than 0.75, but drops sharply as the observation ratio further decreases. Only OLRTR and RTR maintain F1 score of 1.

As for the convergence of OLRTR, we conduct a series of experiments with minibatch size  $\mathbb{R}^{50 \times 50 \times I_3}$  for  $I_3 = 1, 10, 50$ . We fix the corruption ratio at 0.05 and the observation ratio at 0.9. The result is shown in Fig. 6. We can see that OLRTR converges faster with larger batch size. For  $I_3 = 10, 50$ , OLRTR converges after 10 iterations, while for  $I_3 = 1$ , OLRTR converges after about 30 iterations.

## 5.2 Synthetically degraded NOAA data.

In this section, we apply tensor factorization on a complete NOAA dataset [11]. We use temperature data from January to December, 2019 recorded from stationary, high-end climate sensors located at 37 USCRN monitoring sites [40] in the US Midwest. The accessed 37 NOAA sensors record data hourly for 24 hours a day, for 364 days, which is arranged as  $\mathcal{X} \in \mathbb{R}^{37 \times 24 \times 364}$ .

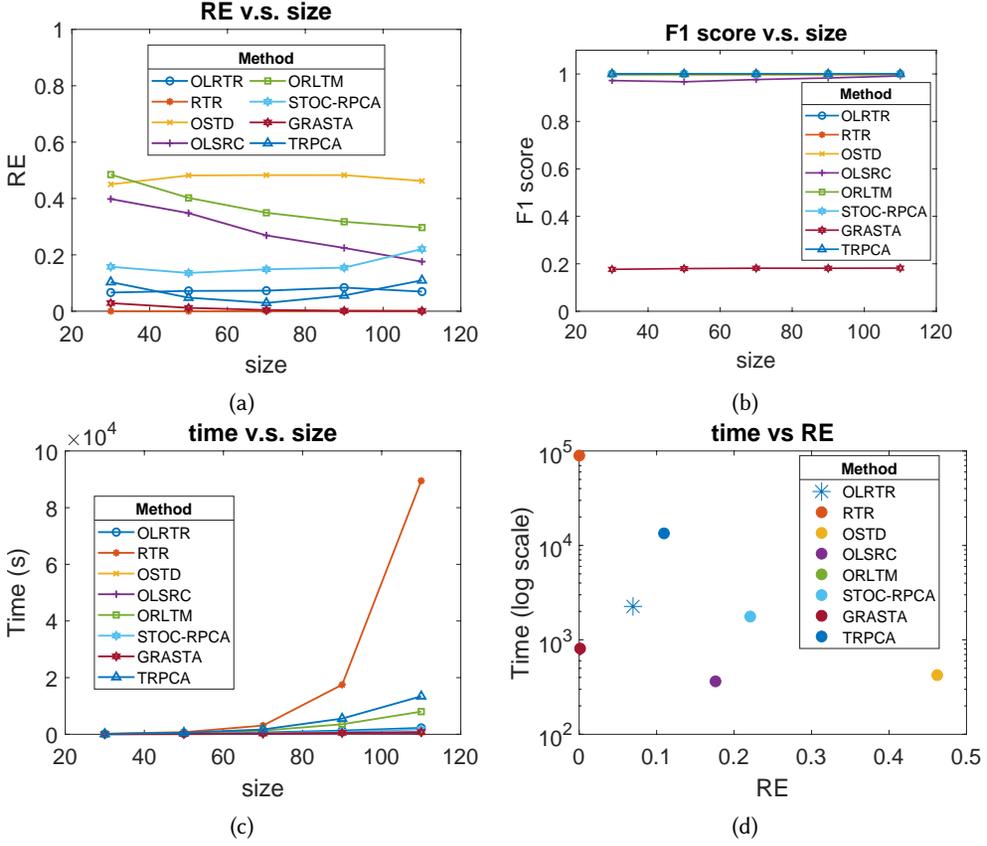


Fig. 3. Results of algorithms as a function of tensor size  $I$ . For each experiment, we run 100 minibatches, varying the tensor size of each minibatch  $\mathbb{R}^{I \times I \times I}$ . The rank is set at  $0.1I$ , the gross corruption ratio  $\gamma$  is set at 0.1, and the observation ratio is set at 0.9. The result is an average over 5 trials.

To test the online factorization method, we generate a synthetically degraded dataset  $\mathcal{B}$  from the true temperature  $\mathcal{X}_{\text{true}}$  that has missing data and erroneous values. We degrade the data by randomly masking 10% of the data, and randomly modify 5% of the tensor fibers to create outlier readings.

Given  $\mathcal{B}$ , we set the OLRTR hyper parameter  $\alpha = 70$ . To capture the daily temperature pattern, we feed in each 24-hour data  $\mathcal{X}^t \in \mathbb{R}^{37 \times 24 \times 1}$  as a minibatch. For all methods, the target rank is set as  $c = 20$  determined by grid search. The results are summarized in Table 1. We can see that among all methods, RTR has the best overall performance in RE and F1 score, whereas among all online methods, OLRTR performs the best overall. In particular, with full observation, OLRTR has comparable RE and F1 score with the batch methods, and has an F1 score at least 0.2 higher than all other online methods. Under partial observations, the RE and F1 score drop for all methods. GRASTA has slightly lower RE than OLRTR, but has a low F1 score of 0.01. OLRTR is the only method among the online methods to have F1 score above 0.9. This experiment on NOAA data shows the validity of our methods on real world sensor networks for data recovery and anomalous sensor detection.

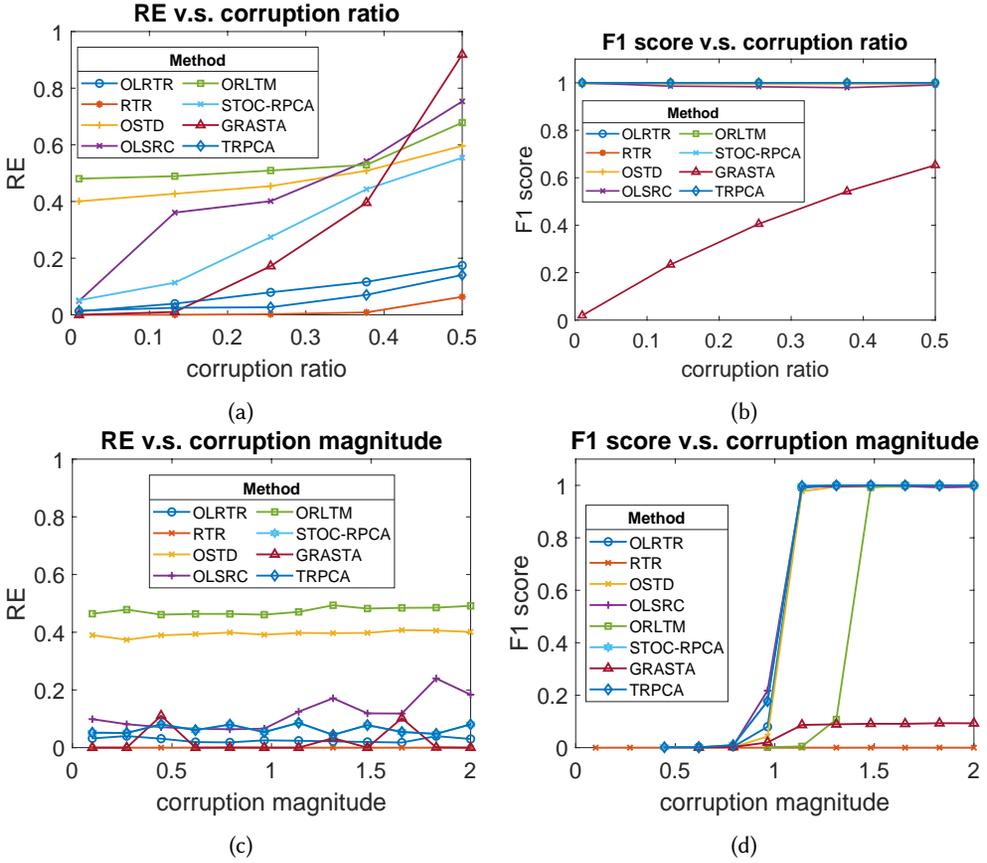


Fig. 4. Results of algorithms with varying corruption ratio (upper row) and magnitude(lower row). We run 100 minibatches. The low-rank tensor size for each minibatch is fixed at  $\mathbb{R}^{50 \times 50 \times 50}$  with a Tucker rank of (3, 3, 3), and the observation ratio is set at 1. The result is an average over 5 trials.

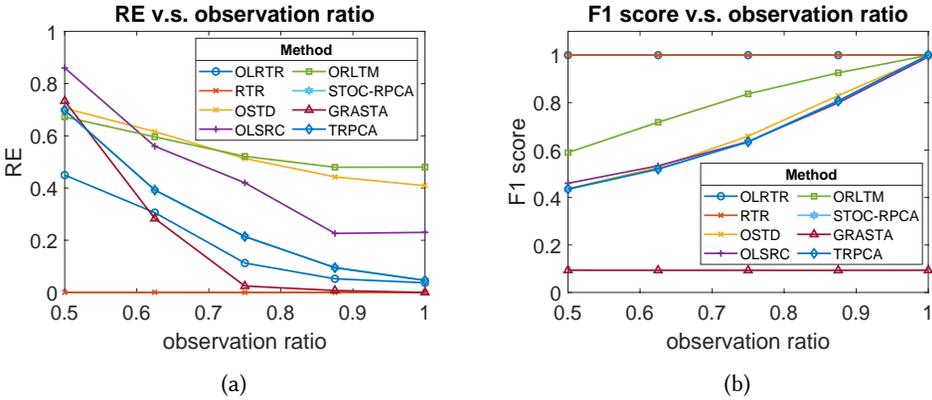


Fig. 5. Results of algorithms with varying observation ratio. The low-rank tensor size for each minibatch is fixed at  $\mathbb{R}^{50 \times 50 \times 50}$  with a Tucker rank of (3, 3, 3), and the corruption ratio is set at 0.05. The result is an average over 5 trials.

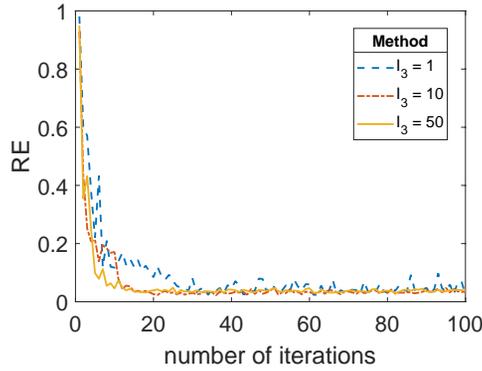


Fig. 6. Convergence speed of OLTR under different minibatch size  $\mathbb{R}^{50 \times 50 \times I_3}$  for  $I_3 = 1, 10, 50$ .

Table 1. The performance of the algorithms on NOAA data. A horizontal line separates the online methods and the batch methods (i.e., RTR & TRPCA).

observation rate	1		0.9	
metrics	RE	F1 score	RE	F1 score
ORLTM [34]	0.157	0.834	0.192	0.309
STOC-RPCA [30]	0.644	0.867	0.644	0.820
OSTD [33]	0.327	0.294	0.394	0.179
OLSRC [31]	0.158	0.800	0.352	0.199
GRASTA [32]	0.085	0.100	<b>0.096</b>	0.099
OLTR	<b>0.053</b>	<b>0.974</b>	0.120	<b>0.953</b>
RTR [10]	0.029	<b>0.984</b>	<b>0.030</b>	<b>0.984</b>
TRPCA [39]	<b>0.012</b>	0.877	0.078	0.188

### 5.3 Array of Things data

The method is finally applied to *Array of Things* (AoT), a dense urban sensor network in Chicago [2] that collects real-time open data on the urban environment, infrastructure, and activity. We construct an AoT temperature tensor as  $\mathcal{X} \in \mathbb{R}^{52 \times 24 \times 183}$ , representing 52 temperature sensor stations aggregated hourly, for 24 hours a day and for 365 days from March 1 2018 to March 1 2019. Approximately 16% of the AoT data is missing in this period. We set  $\alpha = 370$  in OLTR, and the target rank at  $c = 3$ . For all online algorithms, we pass the data three epochs to refine the estimation. Due to the lack of a ground truth dataset, the recovered AOT data is quantitatively compared to the closest NOAA sensor. We use the Pearson correlation coefficient to quantify the agreement since the temperature field is spatially varying.

Table 2 shows the results<sup>1</sup>. We see that the batch method RTR has the highest correlation of 0.98, with the same range as original input,  $[-30, 43]$  Celsius. OLTR has an comparable correlation of around 0.97, with an reasonable temperature range of  $[-28, 43]$  Celsius. STOC-RPCA also has high correlation of 0.978, rightly capturing the trends, but the recovery falls in an unrealistic range of  $[-1, 2]$ . We note that the actual record temperature of Chicago in the studied period is  $[-23, 36]$ , yet AOT has a larger range. This is likely due to the local conditions at the site of the sensor (e.g.,

<sup>1</sup>The GRASTA code generates warnings that the resulting matrix is singular, close to singular or badly scaled, produces NAN results, and is thus not listed.

Table 2. The performance of the algorithms on AOT data, measured by the correlation coefficient between AoT and a nearby NOAA sensor, and the range of recovered temperature ( $^{\circ}\text{C}$ ). We include the raw data measurements, and separate the results of batch methods (RTR, TRPCA) with the online methods into two groups.

	Raw	STOC-RPCA	OSTD	OLSRC	ORLTM	OLRTR	RTR	TRPCA
$r$	0.836	0.978	0.909	0.854	0.862	0.968	<b>0.980</b>	0.841
Range	$[-30, 43]$	$[-1, 2]$	$[-22, 72]$	$[-10, 20]$	$[-37, 90]$	$[-28, 43]$	$[-30, 43]$	$[-29, 42]$

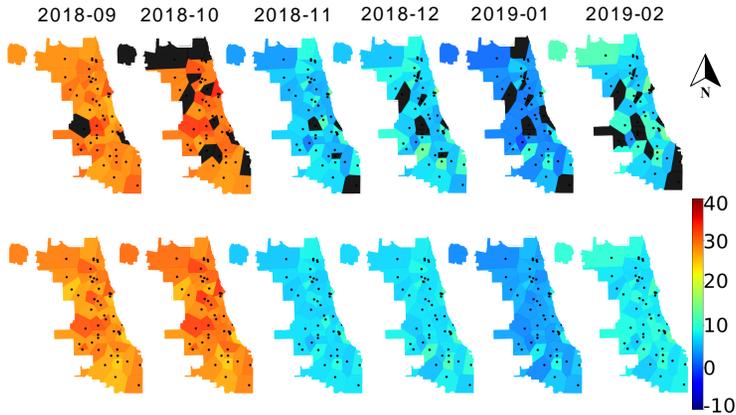


Fig. 7. Voronoi heat maps ( $^{\circ}\text{C}$ ) at 3PM for half a year from Sept. 2018 to Feb. 2018 produced by raw (top; missing data in black) and recovered (bottom) air temperature data. Each dot marks an active AoT unit.

lighting conditions). Fig. 7 shows temperature variation in Chicago in half a year from Sept. 2018 to Feb. 2019 in the raw and recovered dataset. The method recovers a fine-grind temperature map from the raw input with outliers and missing values.

## 6 CONCLUSION

This work introduced an online tensor robust recovery method, and showed its successful application to preprocess data from large urban sensor networks. OLRTR can detect anomalous sensors and impute missing data simultaneously, taking advantage of the multi-dimensional correlations in the dataset. Moreover, by storing and updating a small-sized dictionary that captures the underlying patterns, OLRTR can handle the data sequentially in minibatches, ensuring computational and memory efficiency in streaming systems. Extensive experiments on synthesised and real-world sensor network datasets show significant advantages of OLRTR over other established online methods, and has comparable performance with batch-based methods without the computational overhead.

While we have demonstrated the applications on temperature data, for next step we are also interested to extend the approach to accommodate other environmental sensors co-located on the AoT platform. Ultimately the cleaned data will assist its use by city planners and urban scientists interested in neighborhood-specific heat mitigation strategies to reduce adverse impacts.

## ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grants OAC-1532133 & CMMI-1727785, and the USDOT Eisenhower Fellowship program (No. 693JJ32045011).

## REFERENCES

- [1] United Nations. “The sustainable development goals report”. New York, NY., 2016.
- [2] C. E. Catlett, P. H. Beckman, R. Sankaran, and K. Galvin. Array of things: A scientific research instrument in the public way: Platform design and early lessons learned. In *Proc. of the International Workshop on Science of Smart City Operations and Platforms Engineering*, pages 26–33, 2017.
- [3] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, and M. Welsh. Citysense: An urban-scale wireless sensor network and testbed. In *2008 IEEE Conf. on Technologies for Homeland Security*, pages 583–588, May 2008.
- [4] A. Lewis, W. R. Peltier, and E. von Schneidemesser. Low-cost sensors for the measurement of atmospheric composition: overview of topic and future applications. 2018.
- [5] F. Karagulian, M. Barbieri, A. Kotsev, L. Spinelle, M. Gerboles, F. Lagler, N. Redon, S. Crunaire, and A. Borowiak. Review of the performance of low-cost sensors for air quality monitoring. *Atmosphere*, 10(9):506, 2019.
- [6] M. Daszykowski, K. Kaczmarek, Y. Vander Heyden, and B. Walczak. Robust statistics in data analysis – a review: Basic concepts. *Chemometrics and Intelligent Laboratory Systems*, 85(2):203 – 219, 2007.
- [7] D. J. Hill and B. S. Minsker. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environmental Modelling & Software*, 25(9):1014 – 1022, 2010.
- [8] M. Yu. Smirnov and G. D. Egbert. Robust principal component analysis of electromagnetic arrays with missing data. *Geophysical Journal International*, 190(3):1423–1438, 09 2012.
- [9] X. Y. Chen, Z. C. He, Y. X. Chen, Y. H. Lu, and J. W. Wang. Missing traffic data imputation and pattern discovery with a bayesian augmented tensor factorization model. *Transportation Research Part C: Emerging Technologies*, 104:66 – 77, 2019.
- [10] Y. Hu and D. B. Work. Robust tensor recovery with fiber outliers for traffic events. *ACM Trans. on Knowledge Discovery from Data (TKDD)*, 15(1):1–27, 2020.
- [11] National Centers for Environmental Information. Global summary of the year (GSOY), version 1. <https://www.ncei.noaa.gov/access/search/data-search/global-summary-of-the-year>.
- [12] University of Chicago. Array of Things file browser. <https://afb.plenar.io/data-sets/chicago-complete>, 2019.
- [13] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.
- [14] D. Goldfarb and Z. Qin. Robust low-rank tensor recovery: Models and algorithms. *SIAM Journal on Matrix Analysis and Applications*, 35(1):225–253, 2014.
- [15] L. R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.
- [16] J. F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.
- [17] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.
- [18] A. Famili, W. M. Shen, R. Weber, and E. Simoudis. Data preprocessing and intelligent data analysis. *Intelligent data analysis*, 1(1):3–23, 1997.
- [19] D. M. Dunlavy, T. G. Kolda, and E. Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Trans. on Knowledge Discovery from Data (TKDD)*, 5(2):10, 2011.
- [20] S. Li, M. Shao, and Y. Fu. Multi-view low-rank analysis with applications to outlier detection. *ACM Trans. on Knowledge Discovery from Data (TKDD)*, 12(3):32, 2018.
- [21] Y. K. Wu, H. C. Tan, Y. Li, F. Li, and H. W. He. Robust tensor decomposition based on cauchy distribution and its applications. *Neurocomputing*, 223:107–117, 2017.
- [22] Y. N. Yang, Y. L. Feng, and J. AK. Suykens. Robust low-rank tensor recovery with regularized re-descending m-estimator. *IEEE Trans. on neural networks and learning systems*, 27(9):1933–1946, 2015.
- [23] H. Xu, C. Caramanis, and S. Sanghavi. Robust pca via outlier pursuit. In *Advances in Neural Information Processing Systems*, pages 2496–2504, 2010.
- [24] P. Zhou and J. Feng. Outlier-robust tensor PCA. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2263–2271, 2017.
- [25] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.
- [26] Q. B. Zhao, L. Q. Zhang, and A. Cichocki. Bayesian cp factorization of incomplete tensors with automatic rank determination. *IEEE Trans. on pattern analysis and machine intelligence*, 37(9):1751–1763, 2015.
- [27] Y. K. Wu, H. C. Tan, Y. Li, J. Zhang, and X. X. Chen. A fused cp factorization method for incomplete tensors. *IEEE Trans. on neural networks and learning systems*, 30(3):751–764, 2018.
- [28] Y. L. Chen, C. T. Hsu, and H. Y. M. Liao. Simultaneous tensor decomposition and completion using factor priors. *IEEE Trans. on pattern analysis and machine intelligence*, 36(3):577–591, 2013.

- [29] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(1), 2010.
- [30] Jiashi Feng, Huan Xu, and Shuicheng Yan. Online robust pca via stochastic optimization. In *Advances in neural information processing systems*, pages 404–412, 2013.
- [31] J. Shen, P. Li, and H. Xu. Online low-rank subspace clustering by basis dictionary pursuit. In *International Conf. on Machine Learning*, pages 622–631, 2016.
- [32] J. He, L. Balzano, and J. Lui. Online robust subspace tracking from partial information. *arXiv preprint arXiv:1109.3827*, 2011.
- [33] A. Sobral, S. Javed, S. Ki Jung, T. Bouwmans, and E. H. Zahzah. Online stochastic tensor decomposition for background subtraction in multispectral video sequences. In *Proc. of the IEEE International Conf. on Computer Vision Workshops*, pages 106–113, 2015.
- [34] P. Li, J. S. Feng, X. J. Jin, L. M. Zhang, X. H. Xu, and S. C. Yan. Online robust low-rank tensor modeling for streaming data analysis. *IEEE Trans. on neural networks and learning systems*, 30(4):1061–1075, 2019.
- [35] G. Mateos and G. B. Giannakis. Robust pca as bilinear decomposition with outlier-sparsity regularization. *IEEE Trans. on Signal Processing*, 60(10):5176–5190, 2012.
- [36] H. Kasai, W. Kellerer, and M. Kleinsteuber. Network volume anomaly detection and identification in large-scale networks based on online time-structured traffic tensor tracking. *IEEE Trans. on Network and Service Management*, 13(3):636–650, 2016.
- [37] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.
- [38] J. DM. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proc. of the 22nd international Conf. on Machine learning*, pages 713–719, 2005.
- [39] C. Y. Lu, J. S. Feng, Y. D. Chen, W. Liu, Z. C. Lin, and S. C. Yan. Tensor robust principal component analysis with a new tensor nuclear norm. *IEEE Trans. on pattern analysis and machine intelligence*, 42(4):925–938, 2019.
- [40] What’s a USCRN station? <https://www.ncei.noaa.gov/news/what-is-a-uscrn-station>. Accessed: 2019-08-29.

## APPENDIX

### 6.1 Online algorithm for partial observations

The online algorithm (Algorithm 4) for partial observations is derived similarly to the complete observation setting. Starting with the batch problem (3) and relaxing the constraint as an objective function penalty, we get

$$\begin{aligned} \min_{\mathbf{X}_i, \mathcal{E}} \quad & \frac{1}{2} \sum_{i=1}^N \|\mathbf{X}_i + \mathcal{E} + \mathbf{O} - \mathcal{B}\|_F^2 + \lambda_1 \sum_{i=1}^N \|\mathbf{X}_{i(i)}\|_* + \lambda_2 \|\mathbf{E}_{(1)}\|_{2,1} \\ \text{s.t.} \quad & \mathbf{O}_\Omega = 0, \end{aligned} \quad (20)$$

Next, using the substitution for  $\|\mathbf{X}_{i(i)}\|_*$  as in (5), we obtain:

$$\begin{aligned} \min_{\mathbf{L}_i, \mathbf{R}_i, \mathcal{E}} \quad & \frac{1}{2} \sum_{i=1}^N \|\mathbf{L}_i \mathbf{R}_i^T + \mathbf{E}_{(i)} + \mathbf{O}(i) - \mathbf{B}_{(i)}\|_F^2 + \frac{\lambda_1}{2} \sum_{i=1}^N (\|\mathbf{L}_i\|_F^2 + \|\mathbf{R}_i\|_F^2) + \lambda_2 \|\mathbf{E}_{(1)}\|_{2,1}, \\ \text{s.t.} \quad & \mathbf{O}_\Omega = 0. \end{aligned} \quad (21)$$

Then, we divide the batch data into series of minibatches along the last dimension,  $\mathcal{B} = [\mathcal{B}^1, \mathcal{B}^2, \dots, \mathcal{B}^M]$ , where  $\mathcal{B}^t \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_{N-1} \times I'_N}$ , and  $I'_N = \lfloor I_N/M \rfloor$ . Solving (21) amounts to minimizing the empirical objective function:

$$f_M(\mathbf{L}_i) \triangleq \frac{1}{M} \sum_{t=1}^T \sum_{i=1}^N \left[ l(\mathcal{B}^t, \mathbf{L}_i) + \frac{\lambda_1}{2M} \|\mathbf{L}_i\|_F^2 \right], \quad (22)$$

where the loss function for each mini-batch  $l(\mathcal{B}^t, \mathbf{L}_i)$  is:

$$\begin{aligned} l(\mathcal{B}^t, \mathbf{L}_i) = \quad & \min_{\mathbf{R}_i^t, \mathcal{E}^t, \mathbf{O}^t} \frac{1}{2} \left\| \mathbf{L}_i \mathbf{R}_i^{tT} + \mathbf{E}_{(i)}^t + \mathbf{O}^t(i) - \mathbf{B}_{(i)}^t \right\|_F^2 + \frac{\lambda_1}{2} \|\mathbf{R}_i^t\|_F^2 + \lambda_2 \|\mathbf{E}_{(1)}^t\|_{2,1} \\ \text{s.t.} \quad & \mathbf{O}_\Omega^t = 0. \end{aligned} \quad (23)$$

We take an alternative optimization approach to optimize  $\mathbf{R}_i^t, \mathcal{E}^t, \mathbf{O}^t$  and  $\mathbf{L}_i$ . Namely, at the  $t$ -th time step, after accessing the new sample  $\mathcal{B}^t$ , we first solve for the corresponding  $\mathbf{R}_i^t$  and  $\mathcal{E}^t, \mathbf{O}^t$  using the  $\mathbf{L}_i$  obtained in last step  $t-1$ . Then we update  $\mathbf{L}_i$ , to minimize the accumulated loss given all  $\{\mathbf{R}_i^\tau\}_{\tau=1}^t$  and  $\{\mathcal{E}^\tau\}_{\tau=1}^t$  obtained so far.

The update for  $\mathbf{R}_i^t$  and  $\mathcal{E}^t$  follows a similar approach as in (11), (13). To update  $\mathbf{O}^t$ , we fix  $\mathbf{R}_i^t$  and  $\mathcal{E}^t$  and solve:

$$\begin{aligned} \mathbf{O}^t = \quad & \underset{\mathbf{O}}{\operatorname{argmin}} \sum_{i=1}^N \left\| \mathbf{L}_i \mathbf{R}_i^{tT} + \mathbf{E}_{(i)}^t + \mathbf{O}(i) - \mathbf{B}_{(i)}^t \right\|_F^2 \\ = \quad & \underset{\mathbf{O}}{\operatorname{argmin}} \sum_{i=1}^N \left\| \left( \mathcal{B}^t - \operatorname{fold}_i(\mathbf{L}_i \mathbf{R}_i^{tT}) - \mathcal{E}^t \right) - \mathbf{O} \right\|_F^2 \\ = \quad & \underset{\mathbf{O}}{\operatorname{argmin}} N \left\| \frac{1}{N} \sum_{i=1}^N \left( \mathcal{B}^t - \operatorname{fold}_i(\mathbf{L}_i \mathbf{R}_i^{tT}) - \mathcal{E}^t \right) - \mathbf{O} \right\|_F^2 \\ \text{s.t.} \quad & \mathbf{O}_\Omega = 0, \end{aligned} \quad (24)$$

where in the second row we replace the matrix norm with its tensor norm, which are the same. For (24), we simply set  $\mathbf{O} = \frac{1}{N} \sum_{i=1}^N (\mathcal{B}^t - \operatorname{fold}_i(\mathbf{L}_i \mathbf{R}_i^{tT}) - \mathcal{E}^t)$  for entries  $(I_1, I_2, \dots, I_N) \in \Omega^C$ , and zero otherwise. The sample update for  $\mathbf{R}_i, \mathbf{O}$  and  $\mathcal{E}$  is summarized in Algorithm 5. The stopping criterion is the same as (15). The update for  $\mathbf{L}_i$  follows a similar approach as in the full observation case (18).

---

**Algorithm 4** OLRTR algorithm for partial observation
 

---

- 1: Given  $N$ -way minibatch tensors  $[\mathcal{B}^1, \mathcal{B}^2, \dots, \mathcal{B}^M]$  with  $\mathcal{B}^t \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_{N-1} \times I'_N}$ , weighting parameters  $\lambda_1, \lambda_2$ , target rank  $r$ . Initialize dictionary  $\mathbf{L}_i \in \mathbb{R}^{I_i \times r}$  and accumulation matrices  $\mathbf{A}_i^t \in \mathbb{R}^{r \times r}, \mathbf{D}_i^t \in \mathbb{R}^{I_i \times r}$ .
  - 2: **for**  $t = 0, 1, \dots, M$  **do**
  - 3:   Access the  $t$ -th sample  $\mathcal{B}^t$
  - 4:   Solve Problem (21) for  $\mathbf{R}_i^t, \mathcal{E}^t$  and  $\mathcal{O}^t$  for the new sample using Algorithm 5.
  - 5:    $\mathcal{X}^t = \frac{1}{N} \text{fold}_i(i) (\mathbf{L}_i \mathbf{R}_i^t)$
  - 6:    $\mathbf{A}_i^t = \mathbf{A}_i^{t-1} + \mathbf{R}_i^{tT} \mathbf{R}_i^t; \mathbf{D}_i^t = \mathbf{D}_i^{t-1} + \left( \mathbf{B}_{(i)}^t - \mathbf{E}_{(i)}^t \right) \mathbf{R}_i^t$
  - 7:   Solve  $\mathbf{L}_i^t$  with Algorithm 3, using  $\mathbf{L}_i^{t-1}$  as warm restart.
 
$$\mathbf{L}_i^t = \underset{\mathbf{L}_i}{\text{argmin}} \frac{1}{2} \text{Tr} \left( \mathbf{L}_i^T (\mathbf{A}_i^t + \lambda_1 \mathbf{I}) \mathbf{L}_i \right) - \text{Tr} \left( \mathbf{L}_i^T \mathbf{D}_i^t \right)$$
  - 8: **end for**
  - 9: **return** low rank tensors  $[\mathcal{X}^1, \mathcal{X}^2, \dots, \mathcal{X}^M]$  and outlier tensors  $[\mathcal{E}^1, \mathcal{E}^2, \dots, \mathcal{E}^M]$
- 

**Algorithm 5** Sample update for  $\mathbf{R}_i, \mathcal{E}$  and  $\mathcal{O}$ 


---

- 1: Given observation  $\mathcal{B} \in \mathbb{R}^{I_1 \times \dots \times I_{N-1} \times I'_N}$ , dictionary  $\mathbf{L}_i \in \mathbb{R}^{I_i \times r}$  and parameters  $\lambda_1, \lambda_2$ . Initialize coefficient matrix  $\mathbf{R}_i \in \mathbb{R}^{I'_N \times r}$ , outlier tensor  $\mathcal{E} \in \mathbb{R}^{I_1 \times \dots \times I_{N-1} \times I'_N}$  and compensation tensor  $\mathcal{O} \in \mathbb{R}^{I_1 \times \dots \times I_{N-1} \times I'_N}$  to zero.
  - 2: **while** not converged **do**
  - 3:    $\mathbf{C} \leftarrow (\mathcal{B} - \mathcal{O} - \text{fold}_i(\mathbf{L}_i \mathbf{R}_i^T))$  ▷ Update  $\mathcal{E}$ .
  - 4:   **for**  $j = 1, 2, \dots, p$  **do**
  - 5:      $\mathbf{E}_{(1)j} \leftarrow \mathbf{C}_{(1)j} \max \left\{ 0, 1 - \frac{\lambda}{\mu N \|\mathbf{C}_{(1)j}\|_2} \right\}$
  - 6:   **end for**
  - 7:   **for**  $i = 1, 2, \dots, N$  **do** ▷ Update  $\mathbf{R}_i$
  - 8:      $\mathbf{R}_i \leftarrow (\mathbf{L}_i^T \mathbf{L}_i + \lambda_1 \mathbf{I})^{-1} \mathbf{L}_i^T (\mathbf{B}_{(i)} - \mathbf{E}_{(i)} - \mathcal{O})$ .
  - 9:   **end for**
  - 10:    $\mathcal{O} \leftarrow \sum_{i=1}^N (\mathcal{B} - \mathcal{E} - \text{fold}_i(\mathbf{L}_i \mathbf{R}_i^T))$  ▷ Update  $\mathcal{O}$ .
  - 11:   set  $\mathcal{O}_\Omega = 0$
  - 12: **end while**
  - 13: **return**  $\mathbf{R}_i, \mathcal{E}$  and  $\mathcal{O}$
-