Accurate Online Tensor Factorization for Temporal Tensor Streams with Missing Values

Dawon Ahn
Seoul National University
dawon@snu.ac.kr

Seyun Kim
The Cooper Union
kim79@cooper.edu

U Kang
Seoul National University
ukang@snu.ac.kr

ABSTRACT

Given a time-evolving tensor stream with missing values, how can we accurately discover latent factors in an online manner to predict missing values? Online tensor factorization is a crucial task with many important applications including the analysis of climate, network traffic, and epidemic disease. However, existing online methods have disregarded temporal locality and thus have limited accuracy.

In this paper, we propose STF (Streaming Tensor Factorization), an accurate online tensor factorization method for real-world temporal tensor streams with missing values. We exploit an attention-based temporal regularization to learn inherent temporal patterns of the streams. We also propose an efficient online learning algorithm which allows each row of the temporal factor matrix to be updated from past and future information. Extensive experiments show that the proposed method gives the state-of-the-art accuracy, and quickly processes each tensor slice.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

Tensor Streams; Online Tensor Factorization; Attention-based Temporal Regularization

ACM Reference Format:
Dawon Ahn, Seyun Kim, and U Kang. 2021. Accurate Online Tensor Factorization for Temporal Tensor Streams with Missing Values. In *Proceedings of the 30th ACM Int'l Conf. on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3482048

1 INTRODUCTION

Given a temporal stream tensor, how can we accurately decompose it and predict missing values in an online manner? Tensor streams have become a standard way for representing multi-dimensional and time-stamped event streams from various domains such as climate [23] , network traffic [15], and epidemic diseases [8]. For example, air quality data represented as triples of the form (*place*, *pollutants*, *time*) are modeled as a 3-mode temporal tensor stream containing measurements of pollutants. With rapid growth of need to handle such stream tensors, online tensor analysis tools based on CP (CANDECOMP/PARAFAC) [4, 7] decomposition have been actively developed [9, 11–13, 15–17, 24, 25]. They have effectively analyzed stream tensors in real-time by significantly reducing computational cost and time.

However, most previous approaches do not consider the inherent temporal locality patterns of tensor streams when designing online algorithms. Although a recent work [10] learns the temporal patterns, the existing methods uses only historical information. Inspecting both past and future information is useful to reveal accurate temporal patterns since most real-world tensor streams are heavily related to nearby streams; e.g., the current pollutant measurement is similar to those of previous and next 10 minutes [1]. The main challenges in designing an accurate online tensor factorization method for temporal tensor streams are (1) to model temporal locality patterns with respect to both past and future information, and (2) to design an efficient online update scheme to capture temporal patterns accurately.

In this work, we propose STF (Streaming Tensor Factorization), an accurate online tensor factorization method for missing entry prediction in temporal tensor streams. STF exploits a temporal regularization based on the attention technique [18], and provides an efficient online update method for finding temporal patterns. Through extensive experiments, we show that STF provides the state-of-the-art accuracy for missing entry prediction.

Our main contributions are as follows:

- **Method.** We propose STF, an accurate online tensor decomposition method for real-world temporal tensor streams with missing values.
- **Theory.** We theoretically analyze the time complexity of STF.
- **Experiments.** We evaluate our algorithm on six real-world tensor streams. STF achieves the state-of-the-art performance, and each tensor slice is processed in less than 0.004 second.

The source code and datasets are available at https://github.com/snudatalab/STF.

2 PRELIMINARIES & RELATED WORK

We describe the preliminaries and related works of tensor and tensor factorization. We use the symbols listed in Table 1.

2.1 Tensor

2.1.1 Tensor and Notation. Tensors are defined as multi-dimensional arrays that generalize one-dimensional arrays (or vectors) and two-dimensional arrays (or matrices) to higher dimensions [2, 3, 5, 6]. Specifically, the dimension of a tensor is referred to as mode; the

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482048

Table 1: Table of symbols.

Symbol	Definition	Symbol	Definition
$\mathcal{X}$	tensor $\in \mathbf{R}^{I_1 \times \ldots \times I_N}$	$\|\mathcal{X}\|_F$	Frobenius norm of tensor $\mathcal{X}$
α	index $(i_1, ..., i_N)$	$\mathbf{A}^{(n)}$	nth factor matrix ($\in \mathbf{R}^{I_n \times R}$)
x_α	entry of $\mathcal{X}$ with index α	$\mathbf{a}^{(n)}_{i_n}$	i_nth row of $\mathbf{A}^{(n)}$
N	mode	$a^{(n)}_{i_n r}$	(i_n, r)th entry of $\mathbf{A}^{(n)}$
R	rank	W	window size
I_n	length of the nth mode	$*$	Hadamard product

length of each mode is called 'dimensionality,' denoted by $I_1, \cdots, I_n$. We use boldface Euler script letters (e.g., $\mathcal{X}$) for tensors, boldface capitals (e.g., $\mathbf{A}$) for matrices, and boldface lower cases (e.g., $\mathbf{a}$) for vectors. The $\alpha = (i_1, \cdots, i_N)$th entry of tensor $\mathcal{X}$ is denoted by x_α.

2.1.2 Tensor Factorization. We provide the definition of CP decomposition which is our base model.

Definition 1 (CP decomposition). *Given an N-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ with observed entries and a rank R, the goal of CP decomposition of $\mathcal{X}$ is to find factor matrices $\{\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R} \mid 1 \leq n \leq N\}$ by minimizing the following loss function:*

$$\mathcal{L}\left(\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(N)}\right) = \sum_{\forall \alpha \in \Omega} \left(x_\alpha - \sum_{r=1}^{R} \prod_{n=1}^{N} a^{(n)}_{i_n r} \right)^2 \quad (1)$$

where Ω indicates the set of the indices of the observed entries, x_α indicates the $\alpha = (i_1, \cdots, i_N)$th entry of $\mathcal{X}$, and $a^{(n)}_{i_n r}$ indicates (i_n, r)th entry of $\mathbf{A}^{(n)}$. □

Standard CP decomposition results in poor performance in missing entry prediction of temporal stream tensors since it is not designed to learn temporal locality.

2.2 Related Works

2.2.1 Temporal Tensor Factorization. Tensor factorization methods have been extensively studied for temporal tensor analysis in a static setting [10, 19–22]. Existing methods modeled the temporal factor in an autoregressive way, relying only on historical data to learn temporal patterns. However, it is important to leverage all the surrounding information for learning precise temporal patterns since time-evolving streams are highly correlated with their surroundings [1]. Our proposed model accurately learns the temporal patterns using both past and future information.

2.2.2 Online Tensor Factorization. Online algorithms based on the CP decomposition have been widely developed for stream tensor analysis. They focused on achieving time and memory efficiency [24, 25], eliminating outliers and noises [11], addressing sparsity [9, 15, 24], and dealing with multi-aspect streaming [11, 17]. However, most of the existing works did not take into account the temporal properties of tensors. Lee et al. [10] tried to preserve autoregressive properties of tensor streams, but this causes an inaccurate modeling of temporal patterns, as discussed in Section 2.2.1. Unlike existing online algorithms, STF exploits temporal regularization and uses past and future information to update temporal factors.

3 PROPOSED METHOD

We propose STF, an accurate online tensor factorization method to precisely predict missing entries in temporal tensor streams. Our main idea is an attention-based temporal regularization for learning a temporal property using both past and future information, and an efficient online update scheme effectively leveraging it.

Overall, STF includes two steps: (1) initialization and (2) online update. STF initializes factor matrices with the proposed regularization with an initial batch of an input tensor stream. In the online phase, STF efficiently update factor matrices (see Algorithm 1).

Algorithm 1: Online update of STF (Streaming Tensor Factorization)

Input : N-order tensor $\mathcal{X}_{cur} \in \mathbb{R}^{I_1 \times \ldots \times I_{N-1} \times I_N^{(cur)}}$, previous estimates of factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ $(n = 1 \ldots N)$, auxiliary variables $\{\mathbf{c}^{(n)}_{1,(pin)}, ..., \mathbf{c}^{(n)}_{I_n,(pin)}\}_{n=1}^{N-1}$, $\{\mathbf{B}^{(n)}_{1,(pin)}, ..., \mathbf{B}^{(n)}_{I_n,(pin)}\}_{n=1}^{N-1}$, a rank R, a window size W, and regularization parameters λ_1, λ_2.

Output : updated factor matrix $\mathbf{A}^{(n)}$ and auxiliary variables $\{\mathbf{c}^{(n)}_{1,(pin)}, ..., \mathbf{c}^{(n)}_{I_n,(pin)}\}_{n=1}^{N-1}$, $\{\mathbf{B}^{(n)}_{1,(pin)}, ..., \mathbf{B}^{(n)}_{I_n,(pin)}\}_{n=1}^{N-1}$

/* Temporal factor update */

update each row of a temporal factor matrix $\mathbf{A}^{(n)}_t$ using Eq. (6)

update each row of an adjacent temporal factor matrix $\mathbf{A}^{(n)}_{re}$ using Eq. (6)

/* Non-temporal factor update */

for $n = 1 \ldots N-1$ **do**

  update each row of a non-temporal factor matrix $\mathbf{A}^{(n)}$ using Eq. (8)

  update the auxiliary variables

3.1 Initialization

We explain how to train factor matrices with the proposed temporal regularization with an initial batch of an input tensor stream.

3.1.1 Attention-based Temporal Regularization. We impose a regularization on the temporal factor matrix to reflect the temporal property that tensor slices closer in time are highly related.

Given an N-order temporal tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, with observed entries Ω, the last mode N representing time, a rank R, a window size W, and regularization parameters λ_1, λ_2, we find factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$, $1 \leq n \leq N$ that minimizes

$$\sum_{\alpha \in \Omega} \left(x_\alpha - \sum_{r=1}^{R} \prod_{n=1}^{N} a^{(n)}_{i_n r} \right)^2 + \lambda_1 \sum_{n=1}^{N-1} \|\mathbf{A}^{(n)}\|_F^2 + \lambda_2 \sum_{i_N=1}^{I_N} \|\mathbf{a}^{(N)}_{i_N} - \tilde{\mathbf{a}}^{(N)}_{i_N}\|_2^2 \quad (2)$$

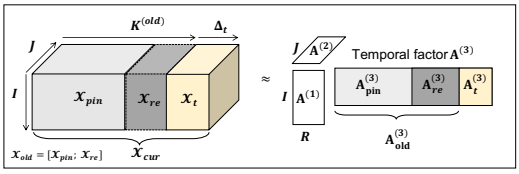
where $\tilde{\mathbf{a}}^{(N)}_{i_N} = \sum_{i_w \in \mathcal{N}(i_N, W)} v(i_N, i_w) \mathbf{a}^{(N)}_{i_w}$ and $\mathcal{N}(i_N, W)$ indicates the set of adjacent indices i_w of i_N in a window of size W. The weight $v(i_N, i_w)$ denotes the weight to give to the i_wth row of the temporal factor matrix. The $\sum_{i_N=1}^{I_N} \|\mathbf{a}^{(N)}_{i_N} - \tilde{\mathbf{a}}^{(N)}_{i_N}\|_2^2$ term in Eq. (2) indicates the proposed temporal regularization where the i_Nth row's factor is constrained to the weighted sum of the neighboring rows' factors to learn temporal dependency.

Inspired by the attention mechanism [18], we design a weight function based on the relevance between the target factor and its neighboring factors. Given a target row index i_N and an adjacent row index $i_w \in \mathcal{N}(i_N, W)$, an attention weight function is calculated as follows:

$$v(i_N, i_w) = \frac{exp\left(\mathbf{a}^{(N)\top}_{i_w} \mathbf{a}^{(N)}_{i_N}\right)}{\sum_{i_{w'}} exp\left(\mathbf{a}^{(N)\top}_{i_{w'}} \mathbf{a}^{(N)}_{i_N}\right)} \quad (3)$$

The attention gives a larger weight to the neighbor factor which produces a larger dot product with the target factor, where all weights always sum to one.

3.1.2 Optimization. We use an alternating least squares (ALS) method to minimize the loss function (2); we update one factor



(a) Temporal factor update

(b) Non-temporal factor update

Figure 1: Online update of STF with a 3rd order tensor. Figure (a) represents how we accurately update temporal factors at time step t. Figure (b) illustrates how we incrementally update non-temporal factors at time step t.

matrix at a time while fixing others. We update factor matrices using the row-wise update rule [14] to use only non-zeros, achieving higher accuracy. The row-wise update rule for i_nth row of the non-temporal factors is given as follows:

$$\mathbf{a}_{i_n}^{(n)} \leftarrow [\mathbf{B}_{i_n}^{(n)} + \lambda_1 \mathbf{I}_R]^{-1} \times \mathbf{c}_{i_n}^{(n)} \tag{4}$$

where

$$\mathbf{B}_{i_n}^{(n)} (\in \mathbb{R}^{R\times R}) = \sum_{\forall \alpha \in \Omega_{i_n}^{(n)}} (\delta_\alpha^{(n)})(\delta_\alpha^{(n)})^\top, \mathbf{c}_{i_n}^{(n)} (\in \mathbb{R}^R) = \sum_{\forall \alpha \in \Omega_{i_n}^{(n)}} x_\alpha \delta_\alpha^{(n)}, \tag{5}$$

$\delta_\alpha^{(n)} = \mathbf{a}_{i_1}^{(1)} * \ldots * \mathbf{a}_{i_{n-1}}^{(n-1)} * \mathbf{a}_{i_{n+1}}^{(n+1)} * \ldots * \mathbf{a}_{i_N}^{(N)}$, $*$ denotes Hadamard product, and $\Omega_{i_n}^{(n)}$ denotes the subset of Ω whose nth mode's index is i_n. Note that $\mathbf{c}_{i_n}^{(n)}$ and $\mathbf{B}_{i_n}^{(n)}$ are intermediate results considering only non-zero values. With the attention weights fixed, the row-wise update rule for i_Nth row of the temporal factor is given as follows:

$$\mathbf{a}_{i_N}^{(N)} \leftarrow [\mathbf{B}_{i_N}^{(N)} + \lambda_{i_N} \mathbf{I}_R]^{-1} \times [\mathbf{c}_{i_N}^{(N)} + \mathbf{d}_{i_N}] \tag{6}$$

where $\mathbf{d}_{i_N} = \lambda_2 \big(1 - v(i_N, i_N)\big) \sum_{i_w \neq i_N} v(i_N, i_w) \mathbf{a}_{i_w}^{(N)}$ and $\lambda_{i_N} = \lambda_2 (1 - v(i_N, i_N))^2$.

3.2 Online Update Algorithm

We propose an accurate and efficient online method for updating factor matrices. For clarity, we discuss the proposed method with a third-order tensor stream; the generalization to higher-order tensors is straightforward. We assume that the tensor only grows in time mode and other modes do not change over time. Under the assumption, a new tensor slice $\mathcal{X}_t \in \mathbb{R}^{I\times J\times \Delta_t}$ at time t is appended to the time mode of an existing tensor $\mathcal{X}_{old} \in \mathbb{R}^{I\times J\times K^{(old)}}$, to create $\mathcal{X}_{cur} \in \mathbb{R}^{I\times J\times K^{(cur)}}$ where $K^{(cur)} = K^{(old)} + \Delta_t$. We also have previous estimates of factor matrices $\{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}\}$ for $\mathcal{X}_{old}$. We optimize the loss function in Eq. (2) with the given $\mathcal{X}_{cur}$.

Optimizing the loss function with ALS is impractical in an online setting since ALS computes all the factors from scratch. Motivated by the work [24, 25], we efficiently optimize the loss function in an alternating manner: (1) update recently created temporal factors with the proposed regularization while fixing other temporal factors, and (2) then incrementally update non-temporal factor matrices according to the updated temporal factors.

Table 2: Comparison of the complexity.

Name	Time
OnlineSCP [24]	$O((\Sigma_{n=1}^{N-1} I_n + \Delta_t)R^2 + \lvert\Omega_t\rvert NR)$
SOFIA [10]	$O(\lvert\Omega_t\rvert NR)$
STF	$O((\Sigma_{n=1}^{N-1} I_n + \Delta_t + W)R^3 + (\lvert\Omega_t\rvert + \lvert\Omega_{t,W}\rvert)NR(N+R))$

3.2.1 Temporal Factor Update. We explain the update procedure of temporal factors as illustrated in the Figure 1(a). Before the time step t, we have the temporal factor $\mathbf{A}_{old}^{(3)} \in \mathbb{R}^{K^{(old)}\times R}$. Due to the addition of $\mathcal{X}_t$, the temporal factor should be augmented to $[\mathbf{A}_{old}^{(3)}; \mathbf{A}_t^{(3)}] \in \mathbb{R}^{K^{(cur)}\times R}$, where $K^{(cur)} = K^{(old)} + \Delta_t$. Our key observation is that in addition to $\mathbf{A}_t^{(3)}$, the last part of $\mathbf{A}_{old}^{(3)}$ should be updated as well, since our temporal regularization in Eq. (2) enforces that each row of the temporal factor matrix is affected by past and future information within the window size W. Thus, we divide $\mathbf{A}_{old}^{(3)}$ into $\mathbf{A}_{pin}^{(3)} \in \mathbb{R}^{(K^{(old)}-W)\times R}$ and $\mathbf{A}_{re}^{(3)} \in \mathbb{R}^{W\times R}$, and update $\mathbf{A}_{re}^{(3)}$ as well as $\mathbf{A}_t^{(3)}$ while fixing $\mathbf{A}_{pin}^{(3)}$. This enables efficient update of temporal factors while imposing temporal regularization.

3.2.2 Non-Temporal Factor Update. We describe how to incrementally update non-temporal factors using the row-wise update rule. We use the first row of the mode-1 factor matrix as a running example, as shown in Figure 1(b). The first row $\mathbf{a}_1^{(1)}$ is updated with $\mathbf{c}_1^{(1)}$ and $\mathbf{B}_1^{(1)}$ where they are divided into two parts, respectively, based on $\mathbf{A}_{pin}^{(3)}$ and $\mathbf{A}_{up}^{(3)} = [\mathbf{A}_{re}^{(3)}; \mathbf{A}_t^{(3)}] \in \mathbb{R}^{(\Delta_t+W)\times R}$, as follows.

$$\mathbf{a}_1^{(1)} \leftarrow [\mathbf{B}_{1,(pin)}^{(1)} + \mathbf{B}_{1,(up)}^{(1)} + \lambda_1 \mathbf{I}_R]^{-1} \times [\mathbf{c}_{1,(pin)}^{(1)} + \mathbf{c}_{1,(up)}^{(1)}] \tag{7}$$

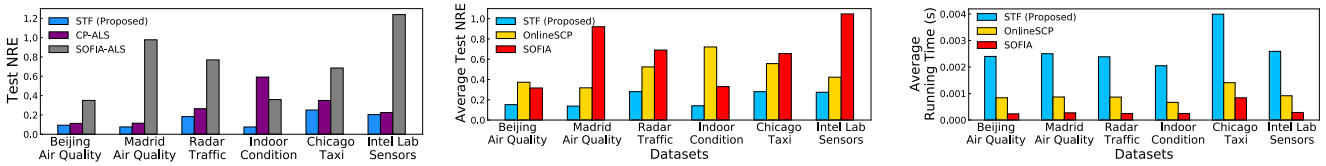
where $\mathbf{c}_{1,(pin)}^{(1)}$ and $\mathbf{B}_{1,(pin)}^{(1)}$ are auxiliary variables storing reusable components with regard to $\mathbf{A}_{pin}^{(3)}$, and $\mathbf{c}_{1,(up)}^{(1)}$ and $\mathbf{B}_{1,(up)}^{(1)}$ are newly computed results with regard to $\mathbf{A}_{up}^{(3)}$. We reduce the computational cost by reusing the variables while computing the rest in online. To prepare auxiliary variables for the next update, we divide $\mathbf{c}_{1,(up)}^{(1)}$ and $\mathbf{B}_{1,(up)}^{(1)}$ as follows:

$$\mathbf{c}_{1,(up)}^{(1)} = [\mathbf{c}_{1,(save)}^{(1)} + \mathbf{c}_{1,(skip)}^{(1)}], \mathbf{B}_{1,(up)}^{(1)} = [\mathbf{B}_{1,(save)}^{(1)} + \mathbf{B}_{1,(skip)}^{(1)}] \tag{8}$$

where each component is related to the two resultant matrices, $\mathbf{A}_{save}^{(3)} \in \mathbb{R}^{\Delta_t\times R}$ and $\mathbf{A}_{skip}^{(3)} \in \mathbb{R}^{W\times R}$, from dividing $\mathbf{A}_{up}^{(3)}$ by the size W from its end. We then add only $\mathbf{c}_{1,(save)}^{(1)}$ and $\mathbf{B}_{1,(save)}^{(1)}$ to $\mathbf{c}_{1,(pin)}^{(1)}$ and $\mathbf{B}_{1,(pin)}^{(1)}$, respectively, since $\mathbf{c}_{1,(skip)}^{(1)}$ and $\mathbf{B}_{1,(skip)}^{(1)}$ will be recalculated at the next update due to our temporal regularization.

3.3 Complexity Analysis

We analyze the computational complexity of STF and existing methods in Table 2. Let the length of the temporal mode in $\mathcal{X}_{cur}$, $\mathcal{X}_t$, and $\mathcal{X}_{re}$ be $I_N^{(cur)}$, Δ_t, and W, respectively. We use $\lvert\Omega_{cur}\rvert$, $\lvert\Omega_t\rvert$, and $\lvert\Omega_{t,W}\rvert$ for the number of nonzeros of the $\mathcal{X}_{cur}$, and $\mathcal{X}_t$, and $\mathcal{X}_{re}$, respectively. Our online iteration takes $O((\lvert\Omega_t\rvert + \lvert\Omega_{t,W}\rvert)NR(N+R) + R^3(\sum_{n=1}^{N-1} I_n + \Delta_t + W))$ while the full ALS iteration takes $O(\lvert\Omega_{cur}\rvert NR(N+R) + R^3(\sum_{n=1}^{N-1} I_n + I_N^{(cur)}))$ at time t (see [14]).



(a) Initialization (b) Missing Entry Prediction (c) Online Update Speed

Figure 2: STF is the most accurate. (a) STF accurately trains an initial model. (b) STF provides the state-of-the-art accuracy for online missing entry prediction. (c) STF processes each slice quickly, (< 0.004 second), while relatively slower than competitors.

4 EXPERIMENT

We run experiments to answer the following questions.

- **Q1 Initialization (Section 4.2.1).** How accurately does STF factorize real-world stream tensors in the initialization step?
- **Q2 Online Accuracy (Section 4.2.2).** How precisely does STF predict missing entries of the streams in an online manner?
- **Q3 Online Update Speed (Section 4.2.3).** How fast is STF to process streams?

4.1 Settings

Machine and Implementation. We implement our algorithm and competitors in Python and conduct all experiments on a PC equipped with a 4.20GHz Intel i7-7700k CPU and 32 GB memory.

Datasets. We evaluate STF on six real-world datasets summarized in Table 3. Beijing Air Quality and Madrid Air Quality are 3-mode tensors (hour, locations, pollutants) containing measurements of pollutants. Radar Traffic is a 3-mode tensor (hour, locations, directions) containing traffic volumes. Chicago Taxi is a 3-mode tensor (hour, sources, destination) containing pickup counts in Chicago. Indoor Condition and Intel Lab Sensor are 3-mode tensors (hour, locations, sensor type) containing measurements. Each dataset is randomly split into training, validation, and test sets with the ratio of 8:1:1; the validation set is used for determining an early stopping point. We set a period of initial streams to 432 for Intel Lab Sensor and 504 for other datasets, and time length of a tensor slice to 50 for Radar Traffic and 10 for the other data sets.

Baselines. We compare STF with the following competitors. SOFIA [10] is the state-of-the-art online CP decomposition method for missing entry prediction. OnlineSCP [24] is the most efficient online CP algorithm for sparse tensors. For static methods used in the initialization step of OnlineSCP and SOFIA, we use CP-ALS [4] and SOFIA-ALS [10], respectively.

Hyper-parameters. We set the rank to 5 and the window size to 1 for all experiments. The regularization penalty is set to 0.1 for Beijing Air Quality, 100 for Indoor Condition, and 10 for others.

Metric. We evaluate the performance using NRE (Normalized Reconstruction Error) and ART (Average Running Time):

$$\text{NRE} = \frac{\sqrt{\sum_{\forall \alpha \in \Omega} (x_\alpha - \hat{x}_\alpha)^2}}{\sqrt{\sum_{\forall \alpha \in \Omega} x_\alpha^2}}, \quad \text{ART} = \frac{1}{T} \sum_{t=1}^{T} Time(t)$$

where Ω indicates the set of the indices of nonzeros. x_α stands for the entry with index α and $\hat{x}_\alpha$ is the corresponding reconstructed value. Note that T indicates the total time steps, and $Time(t)$ represents the running time at time step t. In an online setting, we measure the NRE with $\mathcal{X}_{cur}$ at each time step.

Table 3: Summary of real-world tensor streams. Bold text denotes temporal mode. The datasets are available at https://github.com/snudatalab/STF.

Name	Dimensionality	Nonzero	Granularity
Beijing Air Quality	$12 \times 6 \times$ **5,994**	618,835	hourly
Madrid Air Quality	$26 \times 17 \times$ **3,043**	383,279	hourly
Indoor Condition	$9 \times 2 \times$ **2,622**	59,220	hourly
Radar Traffic	$17 \times 5 \times$ **6,419**	181,719	hourly
Chicago Taxi	$77 \times 77 \times$ **2,904**	424,440	hourly
Intel Lab Sensor	$52 \times 4 \times$ **2,994**	513,508	10 minutes

4.2 Results

4.2.1 Initialization (Q1). We compare the initialization accuracy of STF and the two competitors in terms of test NRE of initial tensor streams. As shown in Figure 2(a), STF shows the best accuracy among all the competitors. STF accurately trains initial factor matrices with up to 4.77x lower test NRE than the second-best method, thanks to the attention-based temporal regularization.

4.2.2 Accuracy (Q2). We measure how accurately STF predicts missing entries of streams in an online manner. Figure 2(b) shows that STF produces outstanding results compared to competitors; it achieves up to 2.34x lower average test NRE compared to the second-best method. Thanks to the initialization strategy and the effective online update algorithm, STF accomplishes superior performance in online missing entry prediction.

4.2.3 Online Update Speed (Q3). We evaluate STF in terms of average running time. Although STF is slower than SOFIA and OnlineSCP, STF runs quickly (< 0.004 sec.) as shown in Figure 2(c), and thus can handle online real time tensors easily. Furthermore, STF provides superior accuracy as shown in Figures 2(a) and 2(b).

5 CONCLUSION

We propose STF, an accurate online tensor decomposition method for temporal stream tensors. STF learns temporal dependencies with a temporal regularization based on attention, and utilizes a fast and accurate online update scheme. On six real-world datasets, STF shows the best accuracy, achieving up to 2.34x lower test NRE for online missing entry prediction among competitors. Future works include extending the idea to tensor forecasting.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea(NRF) funded by MSIT(2019R1A2C2004990). The Institute of Engineering Research and ICT at Seoul National University provided research facilities for this work. U Kang is the corresponding author.

REFERENCES

[1] Dawon Ahn, Jun-Gi Jang, and U Kang. 2021. Time-Aware Tensor Decomposition for Sparse Tensors. *Machine Learning* (2021).

[2] Dawon Ahn, Sangjun Son, and U Kang. 2020. Gtensor: Fast and Accurate Tensor Analysis System using GPUs. In *CIKM*. ACM.

[3] Dongjin Choi, Jun-Gi Jang, and U Kang. 2019. S3CMTF: Fast, accurate, and scalable method for incomplete coupled matrix-tensor factorization. *PLOS ONE* 14, 6 (06 2019), 1–20.

[4] Richard A Harshman et al. 1970. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multimodal factor analysis. (1970).

[5] Jun-Gi Jang and U Kang. 2020. D-Tucker: Fast and Memory-Efficient Tucker Decomposition for Dense Tensors. In *ICDE*. IEEE.

[6] Jun-Gi Jang and U Kang. 2021. Fast and Memory-Efficient Tucker Decomposition for Answering Diverse Time Range Queries. In *KDD*. ACM.

[7] U. Kang, Evangelos E. Papalexakis, Abhay Harpale, and Christos Faloutsos. 2012. GigaTensor: scaling tensor analysis up by 100 times - algorithms and discoveries. In *KDD*. ACM.

[8] Nikos Kargas, Cheng Qian, Nicholas D. Sidiropoulos, Cao Xiao, Lucas M. Glass, and Jimeng Sun. 2021. STELAR: Spatio-temporal Tensor Factorization with Latent Epidemiological Regularization. In *AAAI*. AAAI Press, 4830–4837.

[9] Hiroyuki Kasai. 2016. Online low-rank tensor subspace tracking from incomplete data by CP decomposition using recursive least squares. In *ICASSP*. IEEE, 2519–2523.

[10] Dongjin Lee and Kijung Shin. 2021. Robust Factorization of Real-world Tensor Streams with Patterns, Missing Values, and Outliers. In *ICDE*. IEEE, 840–851.

[11] Mehrnaz Najafi, Lifang He, and Philip S. Yu. 2019. Outlier-Robust Multi-Aspect Streaming Tensor Completion and Factorization. In *IJCAI*. ijcai.org, 3187–3194.

[12] Madhav Nimishakavi, Bamdev Mishra, Manish Gupta, and Partha Pratim Talukdar. 2018. Inductive Framework for Multi-Aspect Streaming Tensor Completion with Side Information. In *CIKM*. ACM, 307–316.

[13] Dimitri Nion and Nicholas D. Sidiropoulos. 2009. Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor. *IEEE Trans. Signal Process.* 57, 6 (2009), 2299–2310.

[14] Kijung Shin, Lee Sael, and U Kang. 2017. Fully Scalable Methods for Distributed Tensor Factorization. *IEEE Trans. Knowl. Data Eng.* 29, 1 (2017), 100–113.

[15] Shaden Smith, Kejun Huang, Nicholas D. Sidiropoulos, and George Karypis. 2018. Streaming Tensor Factorization for Infinite Data Sources. In *SDM*. SIAM, 81–89.

[16] Qingquan Song, Hancheng Ge, James Caverlee, and Xia Hu. 2019. Tensor Completion Algorithms in Big Data Analytics. *ACM Trans. Knowl. Discov. Data* 13, 1 (2019), 6:1–6:48.

[17] Qingquan Song, Xiao Huang, Hancheng Ge, James Caverlee, and Xia Hu. 2017. Multi-Aspect Streaming Tensor Completion. In *KDD*. ACM, 435–443.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[19] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, and Nitesh V. Chawla. 2019. Neural Tensor Factorization for Temporal Interaction Learning. In *WSDM*. ACM, 537–545.

[20] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. 2010. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. In *SDM*. SIAM, 211–222.

[21] Kejing Yin, Ardavan Afshar, Joyce C. Ho, William K. Cheung, Chao Zhang, and Jimeng Sun. 2020. LogPar: Logistic PARAFAC2 Factorization for Temporal Binary Data with Missing Values. In *KDD*. ACM, 1625–1635.

[22] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S. Dhillon. 2016. Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. In *NIPS*. 847–855.

[23] Rose Yu, Dehua Cheng, and Yan Liu. 2015. Accelerated Online Low Rank Tensor Learning for Multivariate Spatiotemporal Streams. In *ICML (JMLR Workshop and Conference Proceedings)*, Vol. 37. JMLR.org, 238–247.

[24] Shuo Zhou, Sarah M. Erfani, and James Bailey. 2018. Online CP Decomposition for Sparse Tensors. In *ICDM*. IEEE Computer Society, 1458–1463.

[25] Shuo Zhou, Xuan Vinh Nguyen, James Bailey, Yunzhe Jia, and Ian Davidson. 2016. Accelerating Online CP Decompositions for Higher Order Tensors. In *KDD*. ACM, 1375–1384.