SliceNStitch: Continuous CP Decomposition of Sparse Tensor Streams (Supplementary Document)

Abstract—**In this supplementary document, we provide (a) the definitions of some basic matrix and tensor operations, (b) proofs of equations and theorems in the main paper, (c) running examples, and (d) additional experimental results.**

I. Matrix and Tensor Operations

Moore-Penrose Inverse: The Moore-Penrose inverse of $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is defined as the matrix $\boldsymbol{A}^{\dagger} \in \mathbb{R}^{n\times m}$ where $\boldsymbol{A}\boldsymbol{A}^{\dagger}\boldsymbol{A} = \boldsymbol{A}$, $\boldsymbol{A}^{\dagger}\boldsymbol{A}\boldsymbol{A}^{\dagger} = \boldsymbol{A}^{\dagger}$, $\left(\boldsymbol{A}\boldsymbol{A}^{\dagger}\right)' = \boldsymbol{A}\boldsymbol{A}^{\dagger}$, and $\left(\boldsymbol{A}^{\dagger}\boldsymbol{A}\right)' = \boldsymbol{A}^{\dagger}\boldsymbol{A}$.

Kronecker Product: The Kronecker product of matrices $\boldsymbol{A} \in \mathbb{R}^{I\times J}$ and $\boldsymbol{B} \in \mathbb{R}^{K\times L}$ is defined as

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1J}B \\ a_{21}B & a_{22}B & \cdots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \cdots & a_{IJ}B \end{bmatrix} \in \mathbb{R}^{(IK)\times(JL)}.$$

Khatri-Rao Product: The Khatri-Rao product $\boldsymbol{A} \odot \boldsymbol{B} \in \mathbb{R}^{IJ\times K}$ of $\boldsymbol{A} \in \mathbb{R}^{I\times K}$ and $\boldsymbol{B} \in \mathbb{R}^{J\times K}$ is defined as $[a_1 \otimes b_1 \;\; a_2 \otimes b_2 \;\; \cdots \;\; a_K \otimes b_K]$, where a_i and b_i are the i-th columns of $\boldsymbol{A}$ and $\boldsymbol{B}$, and $\otimes$ is the Kronecker product.

Hadamard Product: The Hadamard product (a.k.a. the element-wise matrix product) of $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{I\times J}$ is defined as

$$\boldsymbol{A} * \boldsymbol{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \cdots & a_{IJ}b_{IJ} \end{bmatrix} \in \mathbb{R}^{I\times J}.$$

Matricization: The mode-m matricization of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_1\times N_2\times\cdots\times N_M}$ is defined as the matrix $\boldsymbol{X}_{(m)} \in \mathbb{R}^{N_m\times(N_1\cdots N_{m-1}N_{m+1}\cdots N_M)}$ where each column is the mode-m fiber of $\boldsymbol{\mathcal{X}}$.

Frobenius Norm: The Frobenius norm of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_1\times\cdots\times N_M}$ is defined as

$$\left\|\boldsymbol{\mathcal{X}}\right\|_2 = \sqrt{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2}\cdots\sum_{i_M=1}^{N_M}\boldsymbol{\mathcal{X}}\left(i_1, i_2, \cdots, i_M\right)^2}.$$

Outer Product: The outer product of vectors $\boldsymbol{a}^{(1)}, \cdots, \boldsymbol{a}^{(M)}$, where $\boldsymbol{a}^{(m)} \in \mathbb{R}^{N_m}$ for all $m \in \{1, \cdots, M\}$, is defined as the tensor $\boldsymbol{\mathcal{X}} = \boldsymbol{a}^{(1)} \circ \boldsymbol{a}^{(2)} \circ \cdots \circ \boldsymbol{a}^{(M)} \in \mathbb{R}^{N_1\times N_2\times\cdots\times N_M}$, where each $(i_1, i_2, \cdots, i_M)$-th entry is $x_{i_1,i_2,\cdots,i_M} = \prod_{m=1}^{M}\boldsymbol{a}_{i_m}^{(m)}$.

II. Proofs

A. Proof of Theorem 3

Proof. The operation in line 2, called the *matricized tensor times Khatri-Rao product*, takes $O(MR|\boldsymbol{\mathcal{X}} + \Delta\boldsymbol{\mathcal{X}}|)$ time.[1] Computing $\boldsymbol{H} \in \mathbb{R}^{R\times R}$ in line 3 takes $O(MR^2)$ time as each entry of $\boldsymbol{H}$ requires $O(M)$ products. In line 4, computing the pseudoinverse $\boldsymbol{H}^{\dagger}$ takes $O(R^3)$ time [2], and multiplying $\boldsymbol{U} \in \mathbb{R}^{N_m\times R}$ and $\boldsymbol{H}^{\dagger} \in \mathbb{R}^{R\times R}$ takes $O(N_mR^2)$ time. The time complexities of lines 5-7 are dominated by those of lines 2-4. Hence, the time complexity of Algorithm 2, which repeats lines 2-7 with $m = \{1, \cdots, M\}$ is Eq. (5). □

B. Proof of Theorem 4

Proof. Computing $\left(\boldsymbol{X} + \Delta\boldsymbol{X}\right)_{(m)}(i_m, :)\,\boldsymbol{K}^{(m)}$ in Eq. (12) takes $O(MR)$ time per non-zero in $\left(\boldsymbol{X} + \Delta\boldsymbol{X}\right)_{(m)}(i_m, :)$ and thus takes $O(MR \cdot deg(m, i_m))$ time in total. Since $\{\boldsymbol{A}^{(m)\prime}\boldsymbol{A}^{(m)}\}_{m=1}^{M}$ is maintained, computing $\boldsymbol{H}^{(m)} \in \mathbb{R}^{R\times R}$ and $\boldsymbol{H}^{(\boldsymbol{m})\dagger}$ in Eq. (12) takes $O(MR^2)$ and $O(R^3)$ time. Thus, the time complexity of computing Eq. (12) for every $m \in \{1, \cdots, M-1\}$ is Eq. (14), and it dominates the time complexity of the rest of SNS$_{\text{VEC}}$. Hence, the time complexity of SNS$_{\text{VEC}}$ is Eq. (14). □

C. Proof of Theorem 5

Proof. Computing $\left(\boldsymbol{X} + \Delta\boldsymbol{X}\right)_{(m)}(i_m, :)\,\boldsymbol{K}^{(m)}$ in Eq. (12) (line 10 of Algorithm 4) takes $O(MR)$ time per non-zero in $\left(\boldsymbol{X} + \Delta\boldsymbol{X}\right)_{(m)}(i_m, :)$, and thus it takes $O(MR\theta)$ time in total due to the condition in line 9. Computing $\left(\bar{\boldsymbol{X}} + \Delta\boldsymbol{X}\right)_{(m)}\boldsymbol{K}^{(m)}$ in Eq. (16) also takes $O(MR\theta)$ time since $\bar{\boldsymbol{\mathcal{X}}} + \Delta\boldsymbol{\mathcal{X}}$ has at most $(\theta + 2)$ non-zeros. Since $\{\boldsymbol{A}^{(m)\prime}\boldsymbol{A}^{(m)}\}_{m=1}^{M}$ is maintained, computing $\boldsymbol{H}^{(m)} \in \mathbb{R}^{R\times R}$ and $\boldsymbol{H}^{(\boldsymbol{m})\dagger}$ in Eq. (12) and Eq. (16) takes $O(MR^2)$ and $O(R^3)$ time. Similarly computing $\boldsymbol{H}_{prev}^{(m)}$ in Eq. (16) takes $O(MR^2)$ time. Thus, the time complexity of computing Eq. (12) or Eq. (16) for every $m \in \{1, \cdots, M\}$ is Eq. (18), and it dominates the time complexity of the rest of SNS$_{\text{RND}}$. Hence, the time complexity of SNS$_{\text{RND}}$ is Eq. (18). □

D. Proof of Eqs. (21)-(26)

As in the main paper, $\boldsymbol{\mathcal{X}}$ denotes a tensor window. Assume an entry $a_{i_mk}^{(m)}$ of $\boldsymbol{A}^{(m)}$ is being updated. The function f is

[1]It takes $O(MR)$ time per nonzero in $\boldsymbol{\mathcal{X}} + \Delta\boldsymbol{\mathcal{X}}$, as shown in [1].

defined as follows:

$$f(a^{(m)}_{i_m k}, \boldsymbol{\mathcal{Y}}) = \sum_{J \in \Omega^{(m)}_{i_m}} (y_J - \sum_{r \neq k}^{R} \prod_{n=1}^{M} a^{(n)}_{j_n r} - a^{(m)}_{i_m k} \prod_{n \neq m}^{M} a^{(n)}_{j_n k})^2 \tag{29}$$

where $\boldsymbol{\mathcal{Y}}$ is a current tensor window or a tensor that approximates it, and $\Omega^{(m)}_{i_m}$ is the set of indices of $\boldsymbol{\mathcal{X}}$ of which the m-th mode index is i_m. Since minimizing f with respect to $a^{(m)}_{i_m k}$ is a least square problem, the analytical solution, where $\nabla f(a^{(m)}_{i_m k}, \boldsymbol{\mathcal{Y}}) = 0$, exists. The analytical solution is the right side of the update rule in Eq. (30).

$$a^{(m)}_{i_m k} \leftarrow \frac{\sum_{J \in \Omega^{(m)}_{i_m}} ((y_J - \sum_{r \neq k}^{R} (\prod_{n=1}^{M} a^{(n)}_{j_n r})) \prod_{n \neq m}^{M} a^{(n)}_{j_n k})}{\sum_{J \in \Omega^{(m)}_{i_m}} (\prod_{n \neq m}^{M} {a^{(n)}_{j_n k}}^2)} \tag{30}$$

1) Computing the denominator in Eq. (30): Naively computing $\sum_{J \in \Omega^{(m)}_{i_m}} (\prod_{n \neq m}^{M} {a^{(n)}_{j_n k}}^2)$ takes $O(M \prod_{n \neq m}^{M} N_n)$ time as it is a sum of $\prod_{n \neq m}^{M} N_n$ entries, each of which requires $M-1$ multiplications. In order to reduce the computational cost, we reformulate the denominator as follows:

$$\begin{aligned}
&\sum_{J \in \Omega^{(m)}_{i_m}} (\prod_{n \neq m}^{M} {a^{(n)}_{j_n k}}^2) \\
&= \sum_{j_1=1}^{N_1} \cdots \sum_{j_{m-1}=1}^{N_{m-1}} \sum_{j_{m+1}=1}^{N_{m+1}} \cdots \sum_{j_M=1}^{N_M} (\prod_{n \neq m}^{M} {a^{(n)}_{j_n k}}^2) \\
&= \sum_{j_1=1}^{N_1} {a^{(1)}_{j_1 k}}^2 \cdots \sum_{j_{m-1}=1}^{N_{m-1}} {a^{(m-1)}_{j_{m-1} k}}^2 \sum_{j_{m+1}=1}^{N_{m+1}} {a^{(m+1)}_{j_{m+1} k}}^2 \cdots \sum_{j_M=1}^{N_M} {a^{(M)}_{j_M k}}^2 \\
&= \prod_{n \neq m}^{M} (\sum_{j_n=1}^{N_n} {a^{(n)}_{j_n k}}^2)
\end{aligned} \tag{31}$$

Computing the final term, which is the same as $c^{(m)}_k$ of Eq. (20), takes only $O(\sum_{n \neq m}^{M} N_n)$ time in total. We further reduce the time complexity by incrementally updating it using Eq. (24).

2) Computing the numerator in Eq. (30): Separating the terms in the numerator leads to Eq. (32).

$$\sum_{J \in \Omega^{(m)}_{i_m}} (y_J \prod_{n \neq m}^{M} a^{(n)}_{j_n k}) - \sum_{J \in \Omega^{(m)}_{i_m}} ((\sum_{r \neq k}^{R} (\prod_{n=1}^{M} a^{(n)}_{j_n r})) \prod_{n \neq m}^{M} a^{(n)}_{j_n k}) \tag{32}$$

The computational cost of the first term in Eq. (32) depends on algorithms used (e.g., SNS$^+_{\text{VEC}}$ and SNS$^+_{\text{RND}}$). Naively computing the second term takes $O(MR \prod_{n \neq m}^{M} N_n)$ time since $\Omega^{(m)}_{i_m}$ has $\prod_{n \neq m}^{M} N_n$ indices. To efficiently compute the term, we reformulate it as follows:

$$\begin{aligned}
&\sum_{J \in \Omega^{(m)}_{i_m}} ((\sum_{r \neq k}^{R} (\prod_{n=1}^{M} a^{(n)}_{j_n r})) \prod_{n \neq m}^{M} a^{(n)}_{j_n k}) \\
&= \sum_{J \in \Omega^{(m)}_{i_m}} (\sum_{r \neq k}^{R} ((\prod_{n=1}^{M} a^{(n)}_{j_n r})(\prod_{n \neq m}^{M} a^{(n)}_{j_n k}))) \\
&= \sum_{r \neq k}^{R} (\sum_{J \in \Omega^{(m)}_{i_m}} ((\prod_{n=1}^{M} a^{(n)}_{j_n r})(\prod_{n \neq m}^{M} a^{(n)}_{j_n k}))) \\
&= \sum_{r \neq k}^{R} (a^{(m)}_{i_m r} \sum_{J \in \Omega^{(m)}_{i_m}} (\prod_{n \neq m}^{M} (a^{(n)}_{j_n r} a^{(n)}_{j_n k})))
\end{aligned} \tag{33}$$

The last term of Eq. (33) can be expressed as Eq. (34) by applying the technique used in Eq. (31) to $\sum_{J \in \Omega^{(m)}_{i_m}} (\prod_{n \neq m}^{M} (a^{(n)}_{j_n r} a^{(n)}_{j_n k}))$.

$$\sum_{r \neq k}^{R} (a^{(m)}_{i_m r} \prod_{n \neq m}^{M} (\sum_{j_n=1}^{N_n} a^{(n)}_{j_n r} a^{(n)}_{j_n k})) \tag{34}$$

Computing Eq. (34), which is the same as $d^{(m)}_{i_m k}$ of Eq. (20), takes $O(R \sum_{n \neq m}^{M} N_n)$ time. The time complexity can be further reduced thorough incremental updates using Eq. (25).

To sum up, using the notations defined in Eq.(20), we express Eq. (30) as follows:

$$a^{(m)}_{i_m k} \leftarrow (\sum_{J \in \Omega^{(m)}_{i_m}} (y_J \prod_{n \neq m}^{M} a^{(n)}_{j_n k}) - d^{(m)}_{i_m k}) / c^{(m)}_k \tag{35}$$

3) Proof of Eq. (21):

Proof. When using Eq. (21), we use $\boldsymbol{\mathcal{X}} + \Delta \boldsymbol{\mathcal{X}}$ as $\boldsymbol{\mathcal{Y}}$ in Eq. (29) and Eq. (35). Recall that $\Delta \boldsymbol{\mathcal{X}}$ is the change of $\boldsymbol{\mathcal{X}}$ due to the given event. Replacing y_J in Eq. (35) with $x_J + \Delta x_J$ leads to Eq. (21), which is

$$a^{(m)}_{i_m k} \leftarrow (\sum_{J \in \Omega^{(m)}_{i_m}} ((x_J + \Delta x_J) \prod_{n \neq m}^{M} a^{(n)}_{j_n k}) - d^{(m)}_{i_m k}) / c^{(m)}_k. \quad \square$$

4) Proof of Eq. (22):

Proof. When using Eq. (22), we use $\widetilde{\boldsymbol{\mathcal{X}}} + \Delta \boldsymbol{\mathcal{X}}$ as $\boldsymbol{\mathcal{Y}}$. Replacing y_J in Eq. (35) with $\widetilde{x}_J + \Delta x_J$ leads to Eq. (36).

$$a^{(m)}_{i_m k} \leftarrow (\sum_{J \in \Omega^{(m)}_{i_m}} ((\widetilde{x}_J + \Delta x_J) \prod_{n \neq m}^{M} a^{(n)}_{j_n k}) - d^{(m)}_{i_m k}) / c^{(m)}_k. \tag{36}$$

Let $B^{(m)} \equiv A^{(m)}_{prev}$ be $A^{(m)}$ before any update. Then, $\widetilde{x}_J = \sum_{r=1}^{R}(\prod_{n=1}^{M} b^{(n)}_{j_n r})$. Then, Eq.(36) is expressed as follows:

$$a^{(m)}_{i_m k} \leftarrow (\sum_{J\in\Omega^{(m)}_{i_m}} (((\sum_{r=1}^{R}(\prod_{n=1}^{M} b^{(n)}_{j_n r})) \prod_{n\neq m}^{M} a^{(n)}_{j_n k}) + \sum_{J\in\Omega^{(m)}_{i_m}} (\Delta x_J (\prod_{n\neq m}^{M} a^{(n)}_{j_n k})) - d^{(m)}_{i_m k})/c^{(m)}_k. \quad (37)$$

In turn, using the technique applied to Eq. (37) and the technique applied to Eq. (31), the first term of Eq. (37) is expressed as Eq. (38), which is the same as $e^{(m)}_{i_m k}$ defined in Eq. (20).

$$\sum_{r=1}^{R}(b^{(m)}_{i_m r} \prod_{n\neq m}^{M} (\sum_{j_n=1}^{N_n} a^{(n)}_{j_n r} a^{(n)}_{j_n k})) \quad (38)$$

Replacing the first term in Eq. (36) with Eq. (38) (i.e., $e^{(m)}_{i_m k}$) leads to Eq. (22), which is

$$a^{(m)}_{i_m k} \leftarrow (e^{(m)}_{i_m k} + \sum_{J\in\Omega^{(m)}_{i_m}} (\Delta x_J (\prod_{n\neq m}^{M} a^{(n)}_{j_n k})) - d^{(m)}_{i_m k})/c^{(m)}_k. \quad \square$$

5) *Proof of Eq. (23):*

Proof. When using Eq. (23), we use $\widetilde{\boldsymbol{\mathcal{X}}} + \bar{\boldsymbol{\mathcal{X}}} + \Delta\boldsymbol{\mathcal{X}}$ as $\boldsymbol{\mathcal{Y}}$ in Eq. (29) and Eq. (35). Replacing y_J in Eq. (35) with $\widetilde{x}_J + \bar{x}_J + \Delta x_J$ leads to Eq. (39).

$$a^{(m)}_{i_m k} \leftarrow (\sum_{J\in\Omega^{(m)}_{i_m}} ((\widetilde{x}_J + \bar{x}_J + \Delta x_J) \prod_{n\neq m}^{M} a^{(n)}_{j_n k}) - d^{(m)}_{i_m k})/c^{(m)}_k \quad (39)$$

Note that replacing Δx_J in Eq. (36) $\bar{x}_J + \Delta x_J$ leads to Eq. (39). Thus, as in the proof of Eq.(22), we can prove Eq.(23), which is

$$a^{(m)}_{i_m k} \leftarrow (e^{(m)}_{i_m k} + \sum_{J\in\Omega^{(m)}_{i_m}} ((\bar{x}_J + \Delta x_J) \prod_{n\neq m}^{M} a^{(n)}_{j_n k})) - d^{(m)}_{i_m k})/c^{(m)}_k.$$

Note that replacing Δx_J in Eq. (22) with $\bar{x}_J + \Delta x_J$ gives Eq. (23). $\square$

6) *Proof of Eq. (24), Eq. (25), and Eq. (26):*

Proof. Let $\boldsymbol{B}^{(m)} \equiv \boldsymbol{A}^{(m)}_{prev}$ be $\boldsymbol{A}^{(m)}$ before any update; and let $\boldsymbol{Q}^{(m)} \equiv \boldsymbol{A}^{(m)'}\boldsymbol{A}^{(m)}$ and $\boldsymbol{U}^{(m)} \equiv \boldsymbol{A}^{(m)'}_{\boldsymbol{prev}}\boldsymbol{A}^{(m)}$, as in the main paper. By the definition of the matrix product, the following equations are established for all $r \in \{1, \cdots, R\}$.

$$\begin{aligned} q^{(m)}_{kk} &= \sum_{j_m=1}^{N_m} {a^{(m)}_{j_m k}}^2, \\ q^{(m)}_{rk} &= \sum_{j_m=1}^{N_m} a^{(m)}_{j_m r} a^{(m)}_{j_m k}, \\ u^{(m)}_{rk} &= \sum_{j_m=1}^{N_m} b^{(m)}_{j_m r} a^{(m)}_{j_m k}. \end{aligned} \quad (40)$$

Since the terms in Eq. (40), which depend on $a^{(m)}_{i_m k}$, are included in the terms in Eq.(20), it is necessary to update them when $a^{(m)}_{i_m k}$ is updated. This can be done by subtracting the term with old $a^{(m)}_{i_m k}$ and adding the same term with new $a^{(m)}_{i_m k}$ as follows:

$$\begin{aligned} q^{(m)}_{kk} &\leftarrow q^{(m)}_{kk} - (b^{(m)}_{i_m r})^2 + (a^{(m)}_{i_m r})^2, \\ q^{(m)}_{rk} &\leftarrow q^{(m)}_{rk} - a^{(m)}_{i_m r} b^{(m)}_{i_m k} + a^{(m)}_{i_m r} a^{(m)}_{i_m k}, \\ u^{(m)}_{rk} &\leftarrow u^{(m)}_{rk} - b^{(m)}_{i_m r} b^{(m)}_{i_m k} + b^{(m)}_{i_m r} a^{(m)}_{i_m k}. \end{aligned} \quad \square$$

E. *Proof of Theorem 6*

Computing $\sum_{J\in\Omega^{(m)}_{i_m}} \big((x_J + \Delta x_J) \prod_{n\neq m}^{M} a^{(n)}_{j_n k}\big)$ takes $O(M \cdot |\Omega^{(m)}_{i_m}|) = O(M \cdot deg(m, i_m))$ time, and computing $c^{(m)}_k$ and $d^{(m)}_{i_m k}$ takes $O(M)$ and $O(MR)$ time, respectively, as $\sum_{j_n=1}^{N_n} (a^{(n)}_{j_n k})^2$ and $\sum_{j_n=1}^{N_n} a^{(n)}_{j_n r} a^{(n)}_{j_n k}$ are maintained. Thus, computing Eq. (21) takes $O(M \cdot deg(m, i_m) + MR)$ time, and the time complexity of computing Eq. (21) for every $m = \{1, \cdots, M-1\}$ and $k = \{1, \cdots, R\}$, which dominates the time complexity of the rest of SNS^{+}$_{\text{VEC}}$, is Eq. (27). Hence, the time complexity of SNS^{+}$_{\text{VEC}}$ is Eq. (27).

F. *Proof of Theorem 7*

As explained in the proof of Theorem 6, computing Eq.(21) takes $O(M \cdot deg(m, i_m) + MR) = O(M\theta + MR)$, which the equality is due to the condition in line 13. It can be shown similarly that computing Eq. (23) takes $O(M\theta + MR)$ time. Note that in Eq. (23), $\bar{\boldsymbol{\mathcal{X}}} + \Delta\boldsymbol{\mathcal{X}}$ has at most $(\theta + 2)$ non-zeros, and computing $e^{(m)}_{i_m k}$ takes $O(MR)$ time since $\sum_{j_n=1}^{N_n} b^{(n)}_{j_n r} a^{(n)}_{j_n k}$ is maintained. Thus, the time complexity of computing Eq. (21) or Eq. (23) for every $m = \{1, \cdots, M\}$ and $k = \{1, \cdots, R\}$ is Eq. (28). Since it dominates the time complexity of the rest of SNS^{+}$_{\text{RND}}$, the time complexity of SNS^{+}$_{\text{RND}}$ is Eq. (28).

III. Running Examples

We provide running examples for SNS$_{\text{VEC}}$ and SNS$_{\text{RND}}$ in Fig. 10 and Fig. 11, respectively.

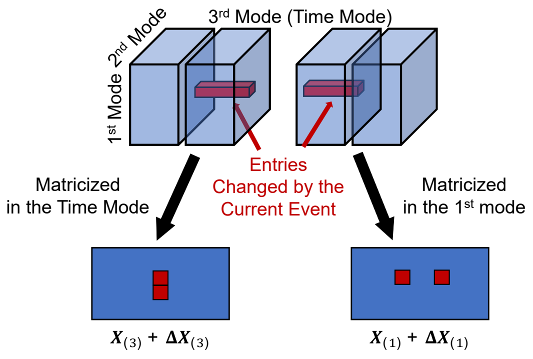
IV. Additional Experiments

A. *Experiments on Synthetic Datasets*

In this section, we study the speed and fitness of different versions of SliceNStitch on synthetic datasets. We generated the M-order synthetic rank R sparse tensor stream $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_M}$ as follows.

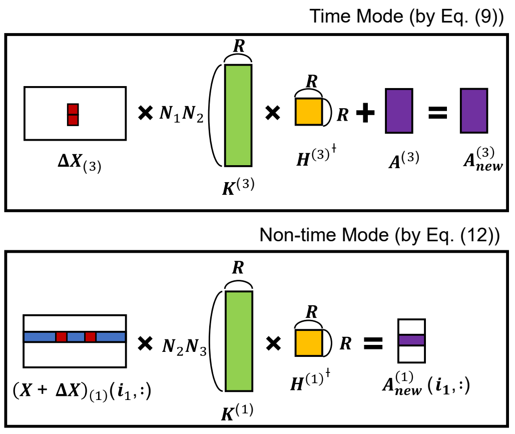
$$\begin{aligned} \boldsymbol{\mathcal{X}} &= [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \cdots, \boldsymbol{A}^{(M)}]\!] \\ \boldsymbol{A}^{(m)}(i,j) &= X_{m,i,j} Y_{m,i,j} \\ X_{m,i,j} &\sim Bernoulli\,(p) \\ Y_{m,i,j} &\sim unif\,(0,1) \end{aligned}$$

where $\boldsymbol{A}^{(m)} \in \mathbb{R}^{N_m \times R}$ for all $m \in \{1, \cdots, M\}$, and $p \in [0,1]$ is a parameter that controls the density of the output tensor stream. Specifically, decreasing p lowers the density of the output tensor stream.



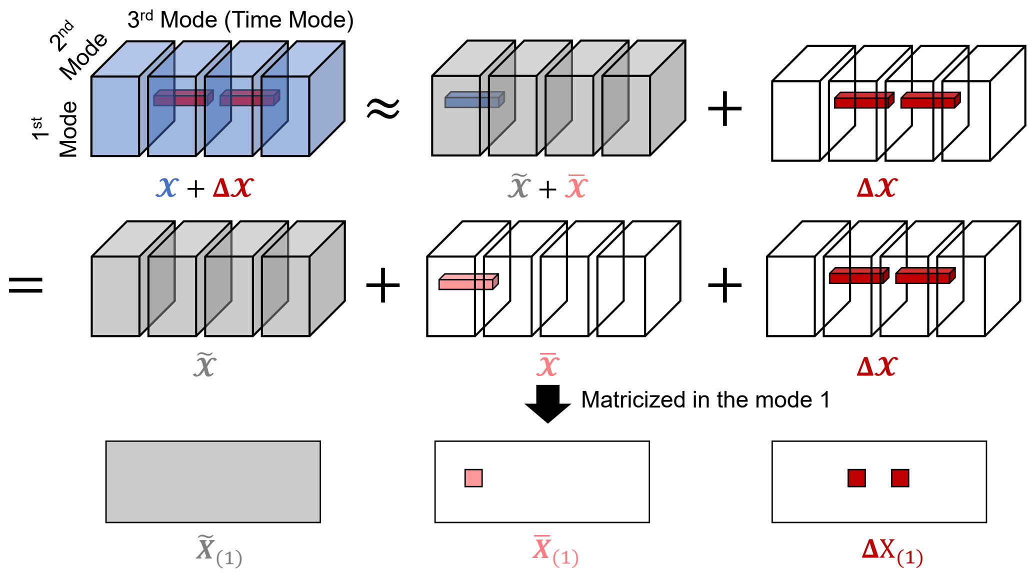
3rd Mode (Time Mode)
2nd Mode
1st Mode
Entries Changed by the Current Event
Matricized in the Time Mode
Matricized in the 1st mode
$X_{(3)} + \Delta X_{(3)}$
$X_{(1)} + \Delta X_{(1)}$

(a) Matricization in SNS_{VEC}

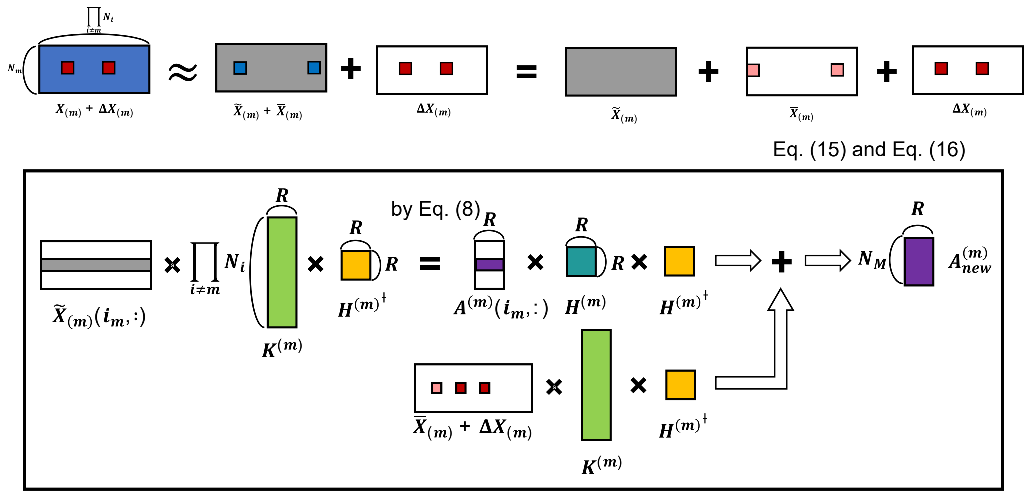


(b) Update processes of SNS_{VEC}

Fig. 10. A running example of SNS_{VEC}. Two rows of the time-mode factor matrix are updated using nonzero entries (which are only two) of $\Delta\boldsymbol{\mathcal{X}}$. A single row of each non-time mode factor matrix is updated with nonzero entries in the corresponding slice of $\boldsymbol{\mathcal{X}} + \Delta\boldsymbol{\mathcal{X}}$, or equivalently, the corresponding row of $(\boldsymbol{X} + \Delta\boldsymbol{X})_{(i)}$.



(a) Approximation and matricization in SNS_{RND}

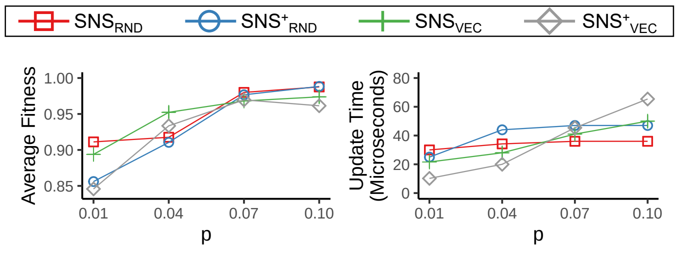


(b) Update process of SNS_{RND}

Fig. 11. A running example of SNS_{RND}. Rows of every factor matrix are updated by the same procedure. Still, two rows are updated in the time-mode factor matrix. and a single row is updated in the other factor matrices.

TABLE IV
HYPERPARAMETERS FOR THE SYNTHETIC DATASETS.

p	R	W	T (Period)	θ	η
$0.01, 0.04, 0.07, 0.1$	20	10	100	10	1000



(a) Average fitness (b) Average speed

Fig. 12. Average fitness and update time of different versions of SLICENSTITCH on the synthetic datasets.

We used four randomly generated three-order tensor streams of size $100 \times 100 \times 6000$ with the hyperparameters listed in Table IV. Similar to Section VI of the main paper, we initialized factor matrices using ALS on the initial tensor window, and we processed the events during $5WT$ time units. We measured fitness 5 times and reported the mean in Fig. 12.

As can be seen in Fig. 12b, the average update time of SNS_{RND} and $\text{SNS}^{+}_{\text{RND}}$ increases as p increased while the average update time of SNS_{VEC} and $\text{SNS}^{+}_{\text{VEC}}$ did not change much. It is consistent with the fact that the time complexities of SNS_{RND} and $\text{SNS}^{+}_{\text{RND}}$ are proportional to the average degree of indices, which is proportional to the density of the tensor. The experimental results indicate that when p is small (i.e., if the density is low), it is recommended to use SNS_{VEC} and $\text{SNS}^{+}_{\text{VEC}}$ since they lead to shorter update times with similar fitness than SNS_{RND} and $\text{SNS}^{+}_{\text{RND}}$. Conversely, when p is large (i.e., if the density is high), it is recommended to use SNS_{RND} and $\text{SNS}^{+}_{\text{RND}}$ because their update times are shorter with similar or even higher fitness than SNS_{VEC} and $\text{SNS}^{+}_{\text{VEC}}$.

REFERENCES

[1] B. W. Bader and T. G. Kolda, "Efficient matlab computations with sparse and factored tensors," *SIAM Journal on Scientific Computing*, vol. 30, no. 1, pp. 205–231, 2008.
[2] P. Courrieu, "Fast computation of moore-penrose inverse matrices," *arXiv preprint arXiv:0804.4809*, 2008.