PuzzleTensor:
A Method-Agnostic Data Transformation for Compact Tensor Factorization

August 3-7, 2025
KDD 2025

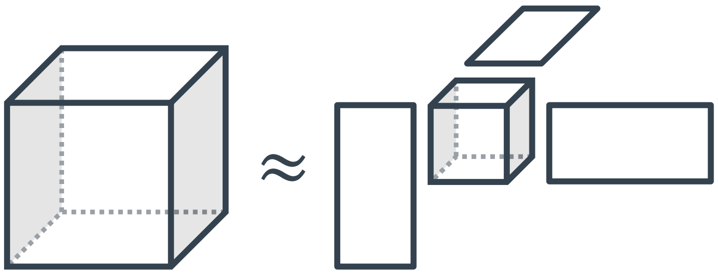
Yong-chan Park, Kisoo Kim, and U Kang
Data Mining Lab
Dept. of CSE
Seoul National University

Outline

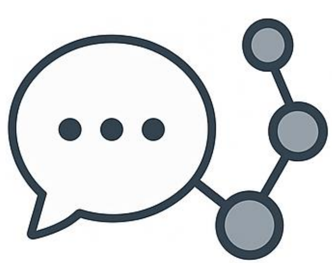
- Introduction
- Preliminaries
- Proposed Method
- Experiments
- Conclusion

Tensor Decomposition

- **Fundamental tool for numerous applications**
  - Recommender systems
  - Topic modeling
  - Hyperspectral imaging
  - Chemometrics



Recommender systems



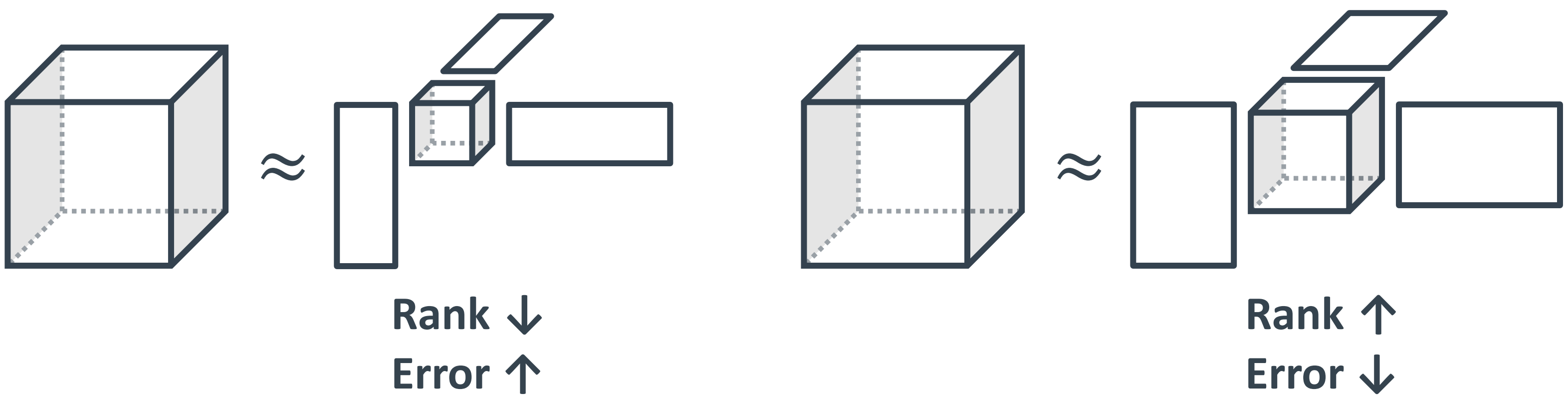
Topic modeling

Hyperspectral imaging

Chemometrics

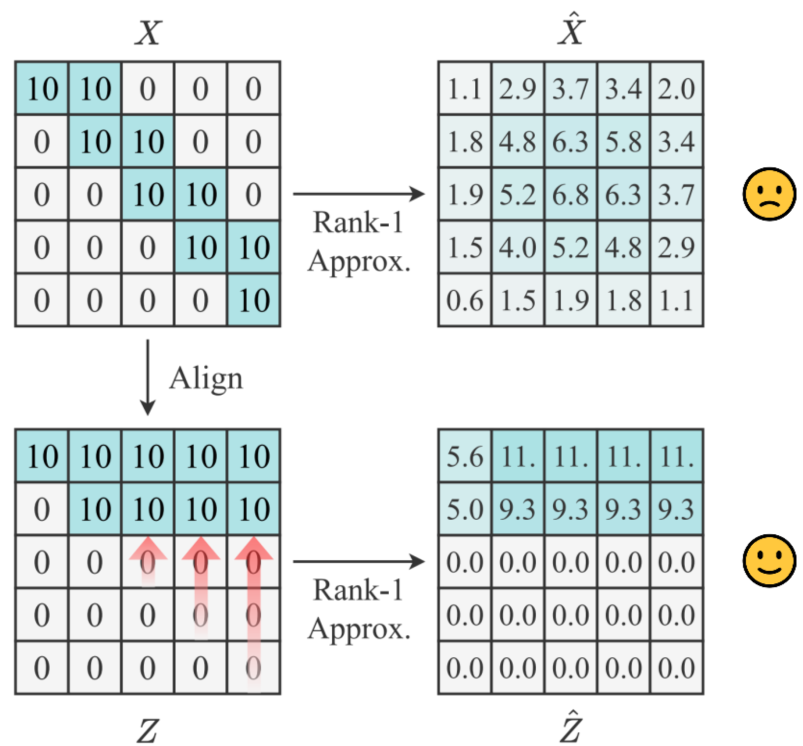
Tensor Decomposition

- **Rank-Error Trade-off**
  - A lower target rank in tensor decompositions enables a more compressed representation of the tensor
  - However, this often comes at the cost of reduced accuracy, since real-world tensors rarely conform to the strict low-rank assumptions



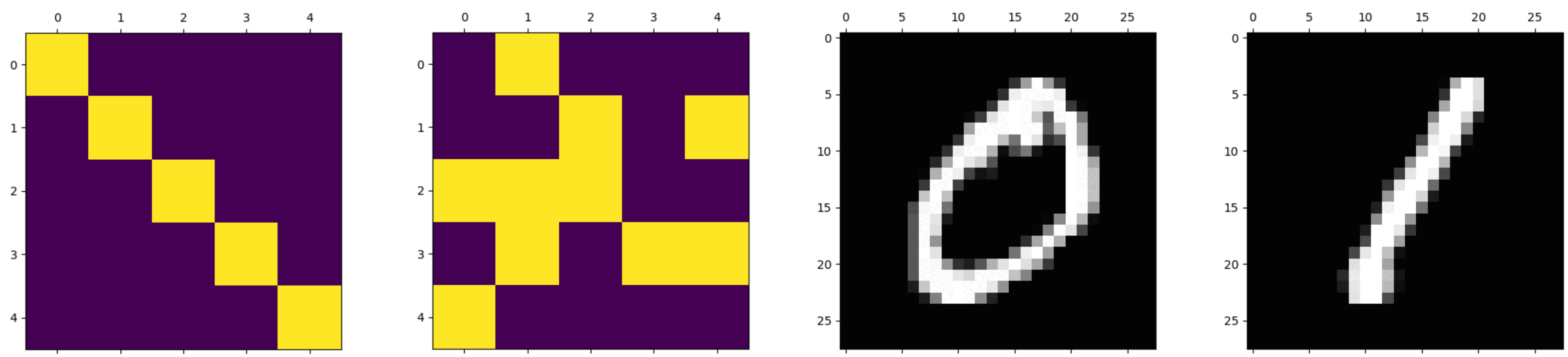
Motivation

- **Key Insight**
  - The rank of a tensor is closely tied to how its slices and modes are spatially arranged



Motivation

- **PuzzleTensor**
  - PuzzleTensor “solves the puzzle” to achieve accurate decompositions with significantly lower target ranks



Problem Definition

- **Tensor Compression**
  - **Given**
    - a D-dimensional tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_D}$
  - **Compress**
    - the tensor $\mathcal{X}$ to get $\mathcal{A}$
  - to **Minimize**
    - (1) the size of $\mathcal{A}$
    - (2) the reconstruction error $\|\mathcal{X} - \hat{\mathcal{X}}\|_F$, where $\hat{\mathcal{X}}$ is the tensor reconstructed from $\mathcal{A}$

Outline

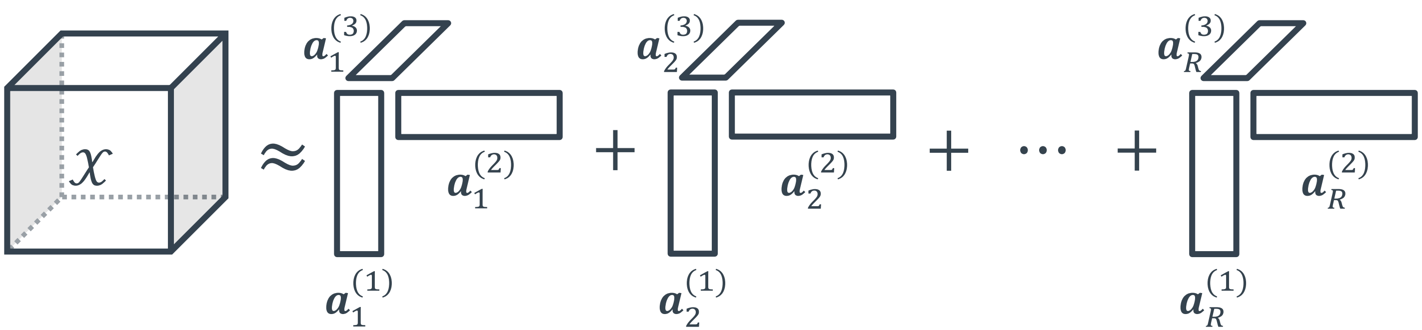
- Introduction
- **Preliminaries**
- Proposed Method
- Experiments
- Conclusion

CP Decomposition

- **CANDECOMP/PARAFAC (CP) Decomposition**
  - It factorizes an n-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ into a sum of rank-1 tensors:

$$\mathcal{X} \approx \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(n)}$$

  - R is the rank, $\boldsymbol{a}_r^{(k)} \in \mathbb{R}^{I_k}$ are the factor vectors for mode k, and $\circ$ denotes the outer product

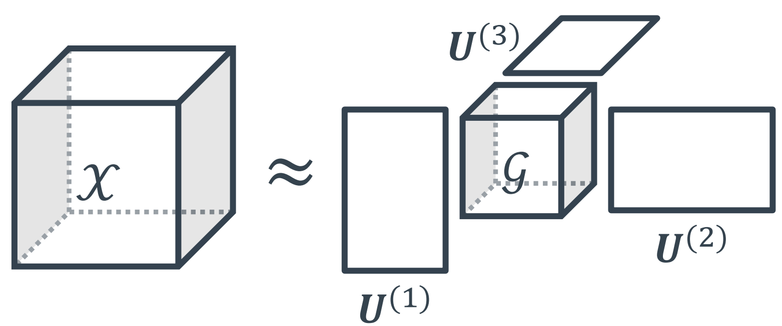


Tucker Decomposition

- **Tucker Decomposition**
  - It generalizes CP by using a core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_n}$ and factor matrices $\boldsymbol{U}^{(k)} \in \mathbb{R}^{I_k \times R_k}$:

$$\mathcal{X} \approx \mathcal{G} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \cdots \times_n \boldsymbol{U}^{(n)}$$

  - $\times_k$ denotes the mode-k product between a tensor and a matrix

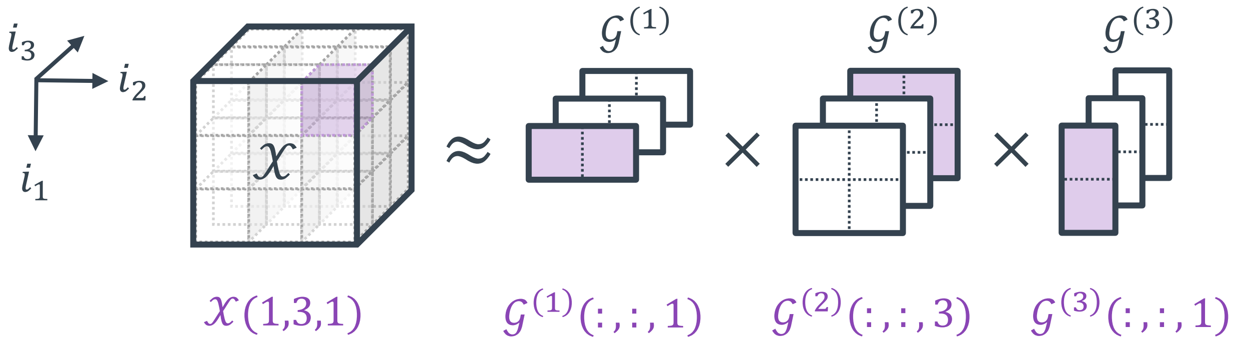


Tensor-Train Decomposition

- **Tensor-Train (TT) Decomposition**
  - It represents an n-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ as a chain of 3D tensors (cores) $\mathcal{G}^{(k)} \in \mathbb{R}^{R_{k-1} \times R_k \times I_k}$:

$$\mathcal{X}(i_1, i_2, \ldots, i_n) \approx \mathcal{G}^{(1)}(:,:,i_1) \cdot \mathcal{G}^{(2)}(:,:,i_2) \cdots \mathcal{G}^{(n)}(:,:,i_n)$$

    - where $R_0 = R_n = 1$



Discrete Fourier Transform

- **Discrete Fourier Transform (DFT)**
  - For an n-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$, the n-dimensional DFT $\widehat{\mathcal{X}}(j_1, \ldots, j_n)$ is defined as follows:

$$\sum_{i_1=0}^{I_1-1} \cdots \sum_{i_n=0}^{I_n-1} \mathcal{X}(i_1, \ldots, i_n) e^{-2\pi i \left(\frac{i_1 j_1}{I_1} + \cdots + \frac{i_n j_n}{I_n}\right)}$$

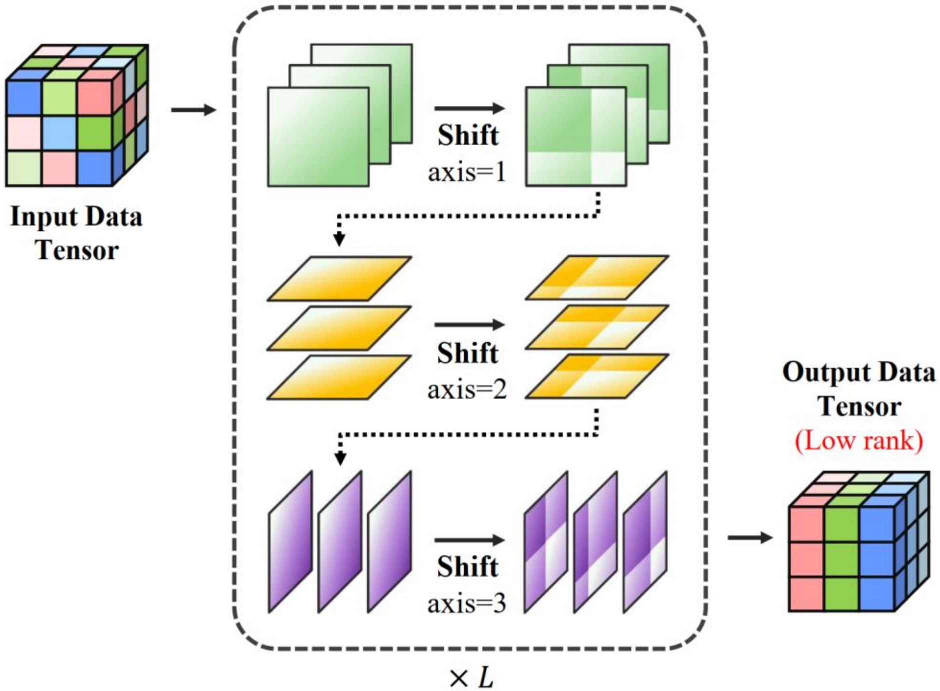
    - where $i = \sqrt{-1}$
  - It maps the time (or spatial) domain sequence to its frequency-domain representation, facilitating signal analysis through well-established spectral methods

Outline

- Introduction
- Preliminaries
- **Proposed Method**
- Experiments
- Conclusion

PuzzleTensor

- **Overview**
  - Given a D-mode tensor $\mathcal{X}$, PuzzleTensor shifts each hyperslice:
    - $\mathrm{Shift}(\mathcal{X}) \coloneqq \mathrm{Shift}_{\mathrm{axis}=D} \circ \mathrm{Shift}_{\mathrm{axis}=D-1} \circ \cdots \circ \mathrm{Shift}_{\mathrm{axis}=1}(\mathcal{X})$
    - $\mathbf{PuzzleTensor}(\mathcal{X}) \coloneqq \mathrm{Shift} \circ \cdots \circ \mathrm{Shift}(\mathcal{X})$ (iterated L times)



PuzzleTensor

- **Challenges**
  - **C-1. Learning the discrete shift operation**
    - Shifting hyperslices of a tensor along specific axes is inherently a discrete operation, which poses a significant challenge for gradient-based optimization
  - **C-2. Transforming a tensor into a low-rank structure**
    - Directly computing the rank of a tensor is an NP-hard problem, making this task computationally infeasible
  - **C-3. Scalability for large-scale tensors**
    - For extremely large tensors, directly learning shifts becomes computationally prohibitive

PuzzleTensor

- **Ideas**
  - **I-1. Fourier-based shift operation**
    - We exploit the properties of the Fourier transform, which allows us to treat discrete shifts as continuous transformations in the frequency domain
  - **I-2. Optimization for low-rank structures**
    - We propose an objective function grounded in a matricized representation of the tensor, designed to capture essential low-rank characteristics
  - **I-3. Sub-block shifting**
    - The input tensor is partitioned into smaller sub-blocks, and the shift operation is independently applied to each block

PuzzleTensor

I-1. Fourier-based shift operation

- Motivation

Spatial domain		Frequency domain
$x(n)$	$\xrightarrow{①}$	$\hat{x}(m)$
		②
$x(n+h)$	$\xleftarrow{③}$	$\hat{x}(m) \times e^{2\pi i m h / N}$

  - ① Discrete Fourier transform (DFT)
  - ② Hadamard product
  - ③ Inverse DFT

$\boldsymbol{x} \in \mathbb{C}^N$: input data
$\hat{\boldsymbol{x}} \in \mathbb{C}^N$: DFT of $\boldsymbol{x}$
$h \in \mathbb{Z}$: integer shift

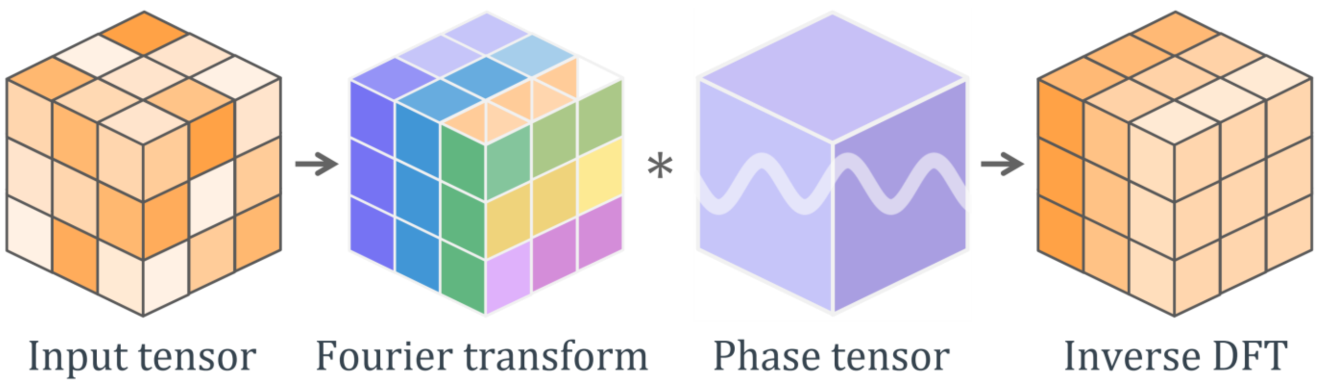
- This property generalizes naturally to **real-valued** shifts $h \in \mathbb{R}$

PuzzleTensor

I-1. Fourier-based shift operation

$$\mathrm{Shift}_{\mathrm{axis}=k}(\mathcal{X}) = \bigoplus_{1\le i_k \le I_k} \mathcal{F}_{D-1}^{-1}\left\{\bigotimes_{j\neq k} \boldsymbol{\phi}_{I_j,\, h_{k,j}(i_k)} * \mathcal{F}_{D-1}\{\mathcal{X}_{i_k}\}\right\}$$

$\mathcal{X}_{i_k} \in \mathbb{R}^{I_1\times\cdots\times I_{k-1}\times I_{k+1}\times\cdots\times I_D}$: $(D-1)$-dimensional hyperslice with fixing the index i_k along mode k
$\mathcal{F}_{D-1}$: $(D-1)$-dimensional discrete Fourier transform
$\boldsymbol{\phi}_{I_j,\, h_{k,j}(i_k)} \in \mathbb{C}^{I_j}$: conjugate-symmetric phase vector with learnable shift parameter $h_{k,j}(i_k)$
$h_{k,j}(i_k) \in \mathbb{R}$: shift amount along mode j for the slice indexed by i_k
$\bigoplus$: tensor concatenation
$\bigotimes$: outer product
$*$: Hadamard product



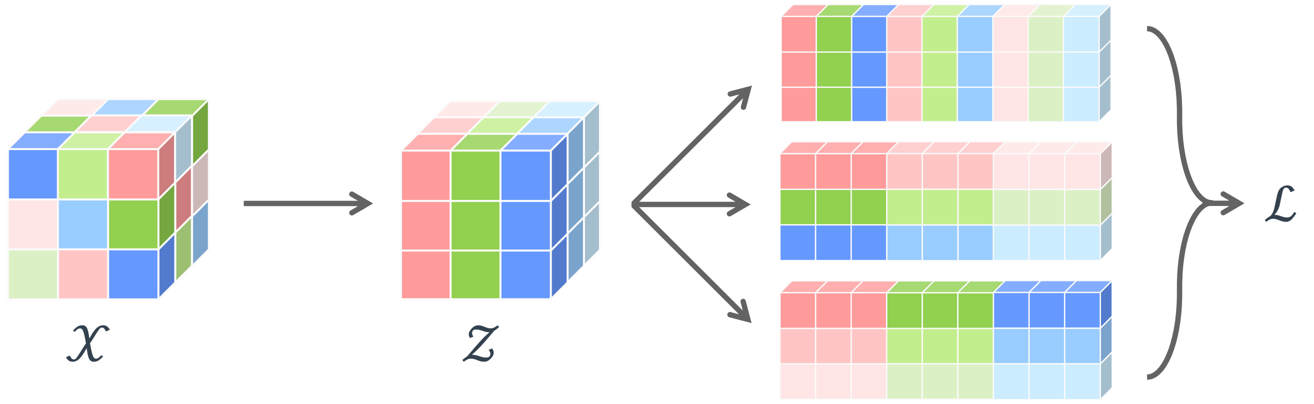
PuzzleTensor

I-2. Optimization for low-rank structures

Loss function

$$\mathcal{L} = \sum_{1 \le k \le D} \frac{1}{\sqrt{I_k}} \|Z_{(k)}\|_*$$

D: dimension of input tensor
I_k: size of the k-mode
$\|\cdot\|_*$: nuclear norm
$Z_{(k)} \in \mathbb{R}^{I_k \times \prod_{j \ne k} I_j}$: k-mode matricization of the transformed tensor



PuzzleTensor

- I-2. Optimization for low-rank structures
  - Theoretical analysis

Theorem. *Let* $\mathcal{S} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$ *be the core tensor of the HOSVD of the transformed tensor* $\mathcal{Z}$ *and* $1 \leq k \leq D$*. Then, for sufficiently large* $I_1 \cdots I_{k-1} I_{k+1} \cdots I_D$*, we have the asymptotic equality*

$$\mathrm{E}[\|vec(\mathcal{S})\|_1] \sim \|Z_{(k)}\|_* \sqrt{\frac{2}{\pi} I_1 \cdots I_{k-1} I_{k+1} \cdots I_D}$$

- Minimizing the nuclear norm of each matricized view of the tensor induces **sparsity in the core tensor** of the corresponding higher-order singular value decomposition (HOSVD)

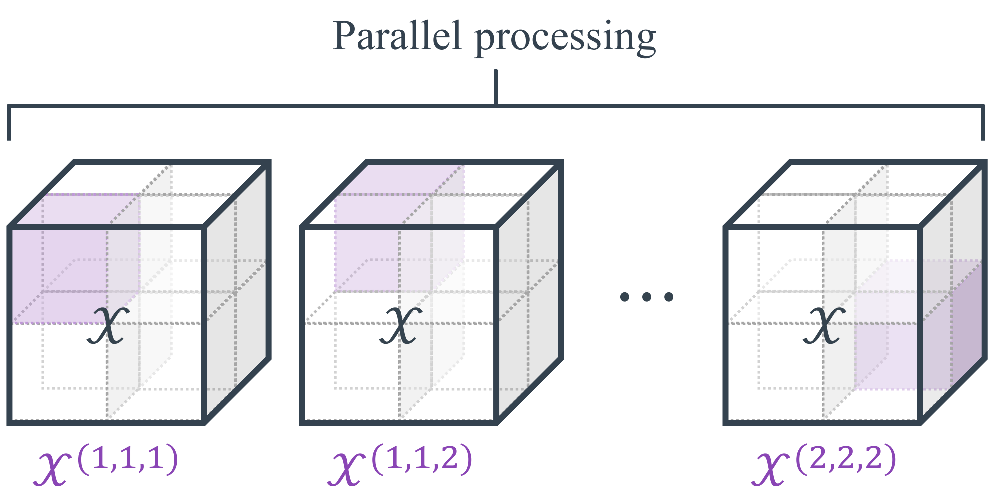
PuzzleTensor

I-3. Sub-block shifting

$$0 = p_{k,0} < p_{k,1} < \cdots < p_{k,B_k-1} < p_{k,B_k} = I_k \quad (k = 1, \dots, D)$$

$$\mathcal{X}^{(\boldsymbol{b})} = \{\mathcal{X}(n_1, \dots, n_D) : p_{k,b_k-1} + 1 \le n_k \le p_{k,b_k} \quad (k = 1, \dots, D)\}$$

$\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$: D-mode input tensor
B_k: number of blocks with respect to mode k
$\boldsymbol{b} = (b_1, \dots, b_D)$: sub-block index with $b_k \in \{1, \dots, B_k\}$



Outline

- Introduction
- Preliminaries
- Proposed Method
- **Experiments**
- Conclusion

Experiments

- **Q1. Performance**
  - How accurately does PuzzleTensor reconstruct tensor data compared to baselines?
- **Q2. Scalability**
  - How does PuzzleTensor scale with increasing input size?
- **Q3. Ablation study**
  - How do different design choices (number of shift layers and block size) affect performance?

Experiments

- **Dataset**
  - We use both synthetic and real-world datasets

Dataset	Type	Size	Density
$\{D_n\}_{n=4,\cdots,8}$	Synthetic	$2^n \times 2^n \times 2^n$	1.000
$\{S_n\}_{n=4,\cdots,8}$	Synthetic	$2^n \times 2^n \times 2^n$	0.010
Uber	Real-world	$183 \times 24 \times 1140$	0.138
Action	Real-world	$100 \times 570 \times 567$	0.393
PEMS-SF	Real-world	$963 \times 144 \times 440$	0.999
Activity	Real-world	$337 \times 570 \times 320$	0.569
Stock	Real-world	$1317 \times 88 \times 916$	0.816
NYC	Real-world	$265 \times 265 \times 28 \times 35$	0.118

- **Measure**
  - Reconstruction error: $||\mathcal{X} - \mathcal{Y}||_F / ||\mathcal{X}||_F$ (lower is better)
    - $\mathcal{X}$: input tensor
    - $\mathcal{Y}$: reconstruction from factors
    - $|| \cdot ||_F$: Frobenius norm

Experiments

- **Q1. Performance**
  - Each decomposition method benefits from PuzzleTensor

Reconstruction errors (lower is better; best within each pair is highlighted)

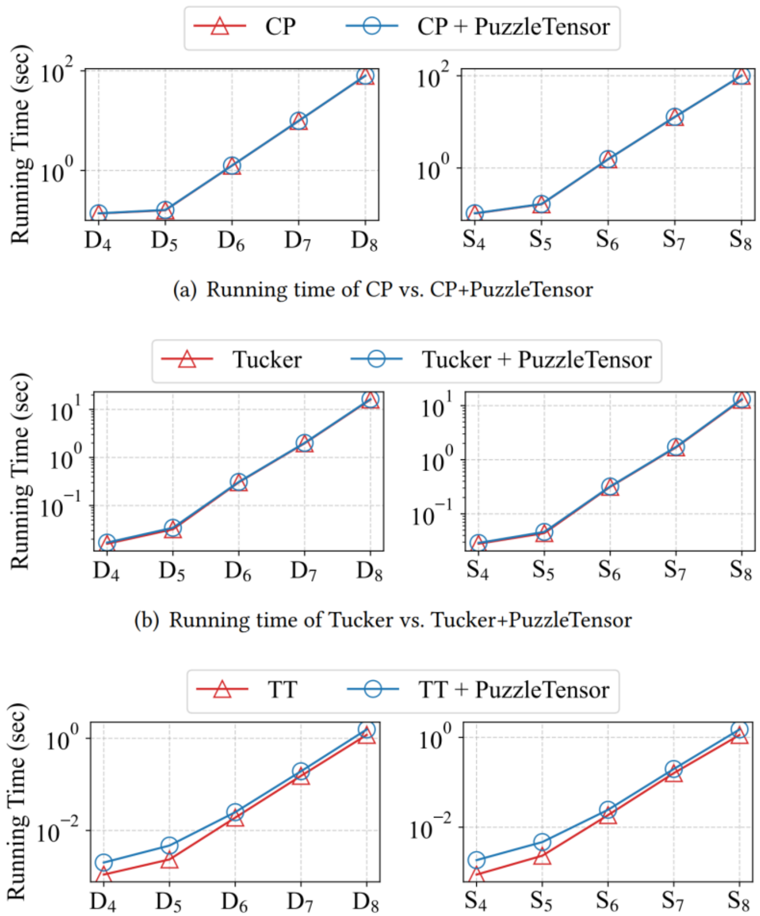
Dataset	D_8			S_8			Uber			Action		
Compressed Size (MB)	116	128	140	116	128	140	34	38	42	220	245	270
CP	0.2488	0.1912	0.1204	0.2051	0.1687	0.1019	0.2189	0.1598	0.1149	0.1935	0.1520	0.1028
CP+PuzzleTensor	0.2185	0.1774	0.1089	0.1764	0.1493	0.0822	0.2048	0.1515	0.1084	0.1825	0.1463	0.0944
Tucker	0.2929	0.2060	0.1493	0.2205	0.1793	0.1176	0.2343	0.1564	0.1040	0.2127	0.1565	0.1178
Tucker+PuzzleTensor	0.2597	0.1788	0.1267	0.1886	0.1407	0.0955	0.2171	0.1376	0.0912	0.1901	0.1426	0.0969
TT	0.2857	0.2088	0.1481	0.2093	0.1556	0.1070	0.2165	0.1497	0.0959	0.2083	0.1438	0.0956
TT+PuzzleTensor	0.2514	0.1739	0.1276	0.1646	0.1364	0.0864	0.1922	0.1329	0.0881	0.1859	0.1372	0.0843

Dataset	PEMS-SF			Activity			Stock			NYC		
Compressed Size (MB)	418	465	512	425	470	515	720	800	880	470	520	570
CP	0.1761	0.1373	0.0953	0.1621	0.1353	0.0893	0.1644	0.1401	0.0958	0.1739	0.1375	0.0955
CP+PuzzleTensor	0.1697	0.1316	0.0902	0.1573	0.1267	0.0836	0.1559	0.1276	0.0871	0.1694	0.1311	0.0909
Tucker	0.1928	0.1452	0.1136	0.1777	0.1419	0.1069	0.1753	0.1456	0.1072	0.1974	0.1431	0.1179
Tucker+PuzzleTensor	0.1804	0.1328	0.0887	0.1708	0.1240	0.0851	0.1674	0.1288	0.0850	0.1866	0.1395	0.1055
TT	0.1966	0.1358	0.0924	0.1839	0.1305	0.0904	0.1883	0.1275	0.0939	0.1980	0.1346	0.0934
TT+PuzzleTensor	0.1744	0.1289	0.0802	0.1722	0.1188	0.0793	0.1668	0.1163	0.0797	0.1843	0.1276	0.0861

Experiments

Q2. Scalability

- Running time of CP, Tucker, and TT decompositions **with and without** PuzzleTensor
- The additional shift operations by PuzzleTensor do not significantly impact computational cost



(a) Running time of CP vs. CP+PuzzleTensor

(b) Running time of Tucker vs. Tucker+PuzzleTensor

(c) Running time of TT vs. TT+PuzzleTensor

Experiments

- Q3. Ablation study
  - Reconstruction errors at varying numbers of layers
    - Even a single layer of PuzzleTensor yields a notable improvement

Number of Layers (L)	0	1	2	3	4
CP+PuzzleTensor	0.621	0.539	0.514	0.506	0.504
Tucker+PuzzleTensor	0.659	0.543	0.501	0.483	0.480
TT+PuzzleTensor	0.675	0.558	0.536	0.514	0.517

  - Effect of the block size
    - Increasing the block size enhances computational efficiency while causing only a minor decrease in accuracy

Block Size (B)	1	2	4	8	16
Running time (sec)	1.355	1.107	0.785	0.562	0.410
Reconstruction error	0.661	0.659	0.663	0.676	0.688

Outline

- Introduction
- Preliminaries
- Proposed Method
- Experiments
- **Conclusion**

Conclusion

- **PuzzleTensor**
  - Leverages hyperslice shifts to achieve compact tensor factorization
- **Main ideas**
  - Learning the discrete shift using Fourier-based operation
  - Minimizing rank via a novel objective function based on a matricized representation of a tensor
  - Reducing runtime with sub-block decomposition
- **Experiments**
  - PuzzleTensor consistently outperforms baselines

THANK YOU!

https://github.com/snudatalab/PuzzleTensor