Compact Decomposition of Irregular Tensors for Data Compression: From Sparse to Dense to High-Order Tensors

Anonymous Author(s)*

ABSTRACT

An irregular tensor is a collection of matrices with different numbers of rows. Real-world data from diverse domains, including medical and stock data, are effectively represented as irregular tensors due to the inherent variations in data length. For their analysis, various tensor decomposition methods (e.g., PARAFAC2) have been devised. While they are expected to be effective in compressing large-scale irregular tensors, akin to regular tensor decomposition methods, our analysis reveals that their compression performance is limited due to the larger number of first mode factor matrices.

In this work, we propose accurate and compact decomposition methods for lossy compression of irregular tensors. First, we propose Light-IT, which unifies all first mode factor matrices into a single matrix, dramatically reducing the size of compressed outputs. Second, motivated by the success of Tucker decomposition in regular tensor compression, we extend Light-IT to Light-IT^{++} to enhance its expressive power and thus reduce compression error. Finally, we generalize both methods to handle irregular tensors of any order and leverage the sparsity of tensors for acceleration.

Extensive experiments on 6 real-world datasets demonstrate that our methods are **(a) Compact:** their compressed output is up to 37× smaller than that of the most concise baseline, **(b) Accurate:** our methods are up to 5× more accurate, with smaller compressed output, than the most accurate baseline, and **(c) Versatile**: our methods are effective for sparse, dense, and higher-order tensors.

ACM Reference Format:
Anonymous Author(s). 2018. Compact Decomposition of Irregular Tensors for Data Compression: From Sparse to Dense to High-Order Tensors. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 24).* ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

A (regular) *tensor* is a multi-dimensional array [18]. Diverse real-world datasets, such as image databases [31, 32], semantic graphs [19], and email histories [2], have been represented as tensors due to their high-dimensional nature. The number of the dimensions of a tensor is called the *order* of the tensor. A d-order tensor can be viewed as a collection of $(d-1)$-order tensors of the same size.

A d-order *irregular tensor* is a collection of $(d-1)$-order tensors with potentially varying sizes, specifically in the first mode. For instance, a 3-order irregular tensor is a collection of matrices with varying numbers of rows (refer to Fig. 1 for an example from EHRs [24, 34]). In many real-world datasets (e.g., EHRs), the lengths of data (e.g. visits counts) vary depending on objects (e.g. patients), and thus they have been represented as irregular tensors [9, 14].

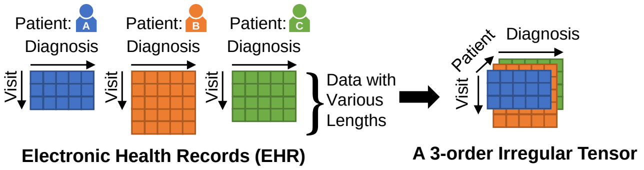


Figure 1: An example of a 3-order irregular tensor.

Given the immense sizes of real-world irregular tensors, compressing them is often necessary for efficient analysis and storage. In the case of a dense irregular tensor, losslessly storing it requires memory proportional to the number of entries in the irregular tensor. The saving cost for an irregular tensor grows exponentially to the order of the tensor since the number of entries in each slice grows exponentially to it. When saving a spare irregular tensor, only indices and values of non-zero entries are stored, but still, it can be a memory burden. For example, the EHR dataset that saves the diagnoses of patients contains 50 million non-zero entries [25].

There have been efforts to enhance the compression abilities of the decompositions for matrices and regular tensors. CUR [6] and CMD [28] decompositions for matrices decrease the memory space for the factor matrices when they are applied to sparse matrices because their factor matrices are sparse and can be saved in sparse matrix formats such as the COO format. Their compression abilities are better than T-SVD (Truncated Singular Value Decomposition) [11], as empirically shown in [20]. NeuKron [20] and TensorCodec [21] use neural networks to increase the compression abilities of Kronecker product and Tensor-Train (TT) decomposition [22], respectively.

In contrast, there is no advanced decomposition method specialized for the compression of an irregular tensor. PARAFAC2 decomposition [12], known for its utility in phenotype discovery [1, 25], fault detection [33], and tracking users' interest [29], may be appealing also for compression purposes. However, our analysis reveals that it requires a large number of factor matrices for the first mode, limiting a compression ability. In addition, their design only targets irregular tensors whose orders are three.

In this paper, we propose Light-IT and Light-IT^{++}, which are accurate and compact decompositions of irregular tensors for lossy compression. In Light-IT, multiple factor matrices for the first mode in PARAFAC2 decomposition are reduced to a single vocabulary matrix, dramatically decreasing the number of parameters. Note that how PARAFAC2 decomposition (and Light-IT) approximates an entry is similar to how CP decomposition [13] does. Light-IT^{++} enhances the expressiveness (and thus reduces the compression error) of Light-IT by generalizing its approximation scheme to mimic Tucker decomposition [30], renowned for its effectiveness

KDD 24, August 25–29, 2018, Barcelona, Spain

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

in regular-tensor compression. Moreover, the designs and implementations of our methods can handle an irregular tensor whose order is higher than three. Finally, our methods exploit the sparsity of a sparse irregular tensor to reduce the time complexity.

We show the following advantages of our methods through experiments in 6 real-world datasets:

- **Compact:** Our methods are up to 37× more concise than the best-working baseline with similar fitness (i.e., compression accuracy).
- **Accurate:** Their fitness is up to 5 times better than the most accurate baseline with similar compressed output sizes.
- **Versatile:** Our methods effectively compress various types of irregular tensors, including sparse, dense, and higher-order ones.

In the context of irregular tensor decomposition (rather than compression), we have the following contributions:

- **Extension of Tucker to Irregular Tensors:** We employ the way that Tucker decomposition approximates an entry in the decomposition of an irregular tensor.
- **Extension to Higher-order Irregular Tensors:** We provide higher-order irregular tensors built with real-world data and decompositions for them.

Reproducibility: The code and datasets are available at https://anonymous.4open.science/r/Light-IT-EF02/.

We present the related works in Section 2 and preliminaries in Section 3. Then, we analyze the memory bottleneck of PARAFAC2 decomposition in Section 4. The main methods are provided in Section 5, which are followed by the experiments in Section 6. We finalize our paper with conclusions in Section 7.

2 RELATED WORK

In this section, we review decomposition-based methods for compressing regular and irregular tensors.

General-purpose regular tensor decompositions: CP (Canonical Polyadic) [13] and Tucker [30] decomposition methods have been widely used for the factorization of regular tensors, serving general purposes including compression. Those methods decompose a regular tensor into small factor matrices (including a core tensor in the case of Tucker), approximating it through the reconstruction from the results. However, they cannot be directly applied to irregular tensor compression due to their designs being focused on regular tensors. One naive approach is to pad zero-valued rows to transform the irregular tensor into a regular one, and then apply the methods. However, this is limited because zero-padded values introduce incorrect information, and it can hinder for them to effectively capture the the different semantics of the first mode indices (e.g., visits) depending on the final mode indices (e.g., patients), which is empirically verified in Section 6.3.

Regular tensor compressions: For a sparse matrix, CUR [6] and CMD [28] compress it into a small dense matrix and tall sparse factor matrices. These sparse matrices are randomly sampled from the input, further reducing space costs by storing them in sparse formats. They are orthogonal to our work because such a sparsification can also be applicable to our methods by incorporating the constraints into the objective function. Recently, deep-learning-based methods have been introduced to enhance regular tensor compression. NeuKron [20] approximates a sparse and reorderable tensor through the Kronecker product of small tensors generated by an RNN, trained with an ordering optimization to exploit meaningful patterns in the input. TensorCodec [21] extends Tensor-Train decomposition [22] by incorporating an RNN, and optimizes it after reordering and folding a tensor during training. However, those methods are not suitable for irregular tensor compression, particularly due to their design limitations in ordering optimization exclusive for regular tensors and their heavy computations.

Table 1: Descriptions for notations

Symbol	Description
$\mathbf{X} \in \mathbb{R}^{N_1 \times N_2}$	N_1-by-N_2 matrix
$\mathbf{X}^{\dagger}$	pseudoinverse of $\mathbf{X}$
$\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_d}$	regular tensor
$nnz(\mathcal{X})$	number of non-zero entries in $\mathcal{X}$
$\mathcal{X}(i_1, \cdots, i_d)$	$(i_1, \cdots, i_d)$-th entry of $\mathcal{X}$
$vec(\mathcal{X}) \in \mathbb{R}^{N_1 \ldots N_d}$	vectorization of $\mathcal{X}$
$\mathbf{X}^{(r_1, \cdots, r_L)}$	mode-$r_1, \cdots, r_L$ matricization of $\mathcal{X}$
$\{\mathcal{X}_k\}_{k=1}^{K}$	irregular tensor
$\mathcal{X}_k \in \mathbb{R}^{N_k \times M_1 \times \cdots \times M_{d-2}}$	slice tensor
$\mathbf{P} \in \mathbb{R}^{D \times R}$	vocabulary matrix
R	rank of our methods
$\mathbf{P}_k$	first-mode factor matrix constructed from $\mathbf{P}$
$\mathbf{U}_k$	temporary matrix for $\mathbf{P}_k$
$\mathbf{V}_i \in \mathbb{R}^{M_i \times R}$	$(i+1)$-th mode factor matrix $(1 \leq i \leq d-2)$
$\mathbf{S} \in \mathbb{R}^{K \times R}$	last mode factor matrix
$[N]$	set of integers from 0 to $N-1$
$\pi_k : [N_k] \rightarrow [D]$	mappings for the k-th slice
$\mathcal{G}$	core tensor
$\otimes$	Kronecker product
$\odot$	Khatri-Rao product
$\|\cdot\|_F$	Frobenius norm

General-purpose irregular tensor decompositions: To decompose an irregular tensor, PARAFAC2 decomposition [12] has been introduced, and it factorizes each slice of the irregular tensor into distinct factor and diagonal matrices depending on the first mode and a shared factor matrix. Various methods have been proposed to calculate PARAFAC2 decomposition under different settings and algorithms. PARAFAC2-ALS [16] is a representative method exploiting the alternative least square (ALS) algorithm. SPARTan [24] leverages the sparsity in sparse irregular tensors for better efficiency. To efficiently handle a dense irregular tensor, RD-ALS [4] preprocesses it via a dimension reduction, and DPar2 [14] employs randomized SVD [10] with careful ordering of computations. For interpretability, COPA [1] introduces several constraints such as temporal smoothness, sparsity, and non-negativity in the objective function of PARAFAC2 decomposition. REPAIR [25] is designed to maintain robustness against missing and erroneous values for PARAFAC2 decomposition. Although the result of PARAFAC2 decomposition can serve as irregular tensor compression, it still lacks compactness due to the dominance of the first mode factor matrices across all slices, resulting in a substantial space requirement for compression parameters (refer to Section 4). Furthermore, the algorithmic design of these methods is restricted to 3-order irregular tensors. We addressed these limitations through our methods.

3 PRELIMINARIES

3.1 Tensor Notations and Operations

A matrix is denoted by $\mathbf{X}$, and a regular tensor is denoted by $\mathcal{X}$. A d-order irregular tensor with K slices is represented as $\{\mathcal{X}_k\}_{k=1}^{K}$, where $\mathcal{X}_k \in \mathbb{R}^{N_k \times M_1 \times \cdots \times M_{d-2}}$ is the k-th slice. The number of non-zero entries of $\mathcal{X}$ is denoted by $nnz(\mathcal{X})$. Matricization [17] is a process that transforms a tensor into a matrix. The details of matricization, Kronecker product, and Khatri-Rao product are available in Appendix A. The Moore-Penrose pseudoivnerse [7] of $\mathbf{X}$ is denoted by $\mathbf{X}^{\dagger}$. The Frobenius norm of a tensor is defined as the square root of the square sum of all entries in the tensor. The frequently used notations are described in Table 1.

3.2 PARAFAC2 Decomposition

Given a 3-order irregular tensor $\{\mathbf{X}_k\}_{k=1}^{K}$ where $\mathbf{X}_k \in \mathbb{R}^{N_k \times M}$, PARAFAC2 decomposition [12] compresses $\mathbf{X}_k$ to $\mathbf{U}_k \in \mathbb{R}^{N_k \times R}$, $\mathbf{S} \in \mathbb{R}^{K \times R}$, and $\mathbf{V} \in \mathbb{R}^{M \times R}$ where R is the rank of the decomposition. The slice $\mathbf{X}_k$ is approximated by $\mathbf{U}_k diag(\mathbf{S}(k,:))\mathbf{V}^{\top}$ where $\mathbf{U}_k$ is distinct for each slice index k. Harshman et al. [12] provided the additional constraints to $\mathbf{U}_k$ so that it should satisfy $\mathbf{U}_k^{\top}\mathbf{U}_k = \Phi$ for all indices k. These constraints aim to the uniqueness of $\mathbf{U}_k$ in the solution space, and these are acheived by replacing $\mathbf{U}_k$ with $\mathbf{Q}_k\mathbf{H}$ where $\mathbf{Q}_k$ are column orthogonal matrices with R columns.

The problem for PARAFAC2 decomposition is given as follows:

$$\min_{\Theta} \sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{Q}_k\mathbf{H}diag(\mathbf{S}(k,:))\mathbf{V}^{\top}\|_F^2 \text{ s.t. } \mathbf{Q}_k^{\top}\mathbf{Q}_k = \mathbf{I} \ \forall \ k, \tag{1}$$

where $\Theta = \left\{\{\mathbf{Q}_k\}_{k=1}^{K}, \mathbf{S}, \mathbf{H}, \mathbf{V}\right\}$. The representative algorithm for solving the problem is PARAFAC2-ALS [16], which alternatively optimizes each term in Θ while fixing the other parameters.

3.3 Problem Definition

The problem of lossy compression of an irregular tensor, addressed in this paper, is defined as follows:

Problem 1. (Lossy Compression of an Irregular Tensor)

- ***Given:*** *an irregular tensor* $\{\mathcal{X}_k\}_{k=1}^{K}$ *where* $\mathcal{X}_k \in \mathbb{R}^{N_k \times M_1 \times \cdots \times M_{d-2}}$,
- ***Find:*** *the compressed data* $\mathcal{D}$
- ***to Minimize:*** *(1) the size of* $\mathcal{D}$

  (2) the approximation error $\sum_{k=1}^{K} \|\mathcal{X}_k - \tilde{\mathcal{X}}_k\|_F^2$

  where $\tilde{\mathcal{X}}_k$ *is the reconstruction of the k-th slice.*

For our methods, some components of $\mathcal{D}$ are mappings between the rows of a vocabulary matrix and the first mode indices of an irregular tensor. The other components are the vocabulary matrix and the factor matrices for the modes except the first mode. In the case of Light-IT^{++}, a core tensor is included in $\mathcal{D}$.

4 BOTTLENECK ANALYSIS

In this section, we aim to analyze a limitation of PARAFAC2 decomposition as a solution of Problem 1. A commonly considered approach for compressing a 3-order irregular tensor is to use PARAFAC2 decomposition. However, its compression level is very limited due to the size $\sum_{k=1}^{K} N_i R$ of the first mode factor matrices $\{\mathbf{U}_k\}_{k=1}^{K}$ which are much larger than the other factor matrices. Figure 2 shows that the number of parameters in $\{\mathbf{U}_k\}_{k=1}^{K}$ overwhelmingly surpasses those in the remaining parameters ($\mathbf{S}$ and $\mathbf{V}$) for all datasets. In light of this result, it is crucial to reduce the size of $\{\mathbf{U}_k\}_{k=1}^{K}$ to achieve a higher compression performance. Our proposed Light-IT and Light-IT^{++}, which are explained in the following sections, focus on overcoming this limitation without an accuracy loss.

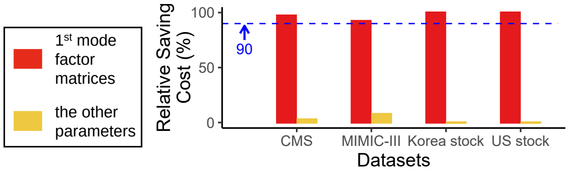


Figure 2: The first mode factor matrices of PARAFAC2 decomposition consume most of the parameter space. Relative saving cost is defined as 100 × (number of parameters)/(number of total parameters). The results are the same for all ranks.

5 PROPOSED METHOD

5.1 Overview

The technical challenges in addressing Problem 1 are as follows.

Q1. **Compactness.** As analyzed in Section 4, the compression capability of PARAFAC2 decomposition is limited due to the first mode factor matrices, requiring a substantial compression size. How can we compress it more compactly?

Q2. **Expressiveness.** How can we increase the expression power to fit irregular tensors of more complicated patterns that cannot be expressed with low-rank PARAFAC2 decomposition?

Q3. **Handling large sparse irregular tensors.** Sparse irregular tensors usually have large dimensions with few non-zero entries, requiring much time if treated as dense tensors. How can we efficiently compress them?

Q4. **Handling higher-order irregular tensors.** Real-world irregular tensors can have orders higher than three, as mentioned in [9]. How can we compress such irregular tensors?

We provide the following solutions to handle the challenges above.

A1. **Vocabulary-based compression.** We devise a vocabulary-based compression that maps the first mode factor matrices to a single compact vocabulary matrix.

A2. **Extension with core tensor.** We extend the vocabulary-based compression by incorporating a core tensor, similar to how Tucker decomposition approximates a target tensor.

A3. **Sparse design.** We design a sparse version of our methods, focusing on operations involving non-zero entries while zero entries are excluded from computation.

A4. **Higher-order design.** We design a higher-order version of our methods through careful matricizations of irregular tensors with higher orders.

In the following subsections, we describe our compact and accurate irregular tensor compression methods, Light-IT and Light-IT^{++}. Light-IT compresses an input irregular tensor with the vocabulary matrix, thereby reducing the compression size (Section 5.2). To enhance expressiveness, we extend it to Light-IT^{++} based on the Tucker decomposition, where the initial parameters of Light-IT^{++} are set to the outputs of Light-IT (Section 5.3). Efficient designs of Light-IT and Light-IT^{++} for sparse irregular tensors and extensions to higher-order irregular tensors are explained in the corresponding

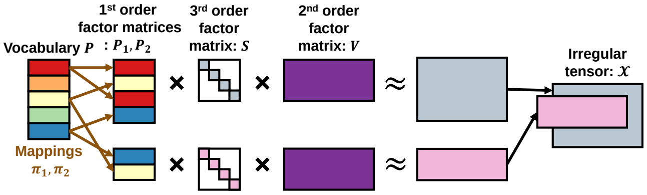

Vocabulary P
1st order factor matrices : P_1, P_2
3rd order factor matrix: S
2nd order factor matrix: V
Irregular tensor: $\mathcal{X}$
Mappings π_1, π_2
×
×
≈
×
×
≈

Figure 3: Approximation by Light-IT.

subsections. For simplicity, we initially describe our methods in the context of a 3-order irregular tensor and subsequently extend the explanation to higher-order irregular tensors.

5.2 Light-IT: Vocabulary-based Compression

To achieve compact compression, our main idea for Light-IT is to compress $\{\mathbf{U}_k\}_{k=1}^{K}$ into a single, compact vocabulary matrix $\mathbf{P}$ that is partially shared across all slices for the approximation. Specifically, Light-IT learns mapping functions $\{\pi_k\}_{k=1}^{K}$ while updating $\mathbf{U}_k \in \mathbb{R}^{N_k \times R}$ and $\mathbf{P} \in \mathbb{R}^{D \times R}$ where $\pi_k : [N_k] \rightarrow [D]$ maps a row of $\mathbf{U}_k$ to a corresponding row in $\mathbf{P}$, essentially treating it as part of the vocabulary. Note that D is the number of vocabularies, which is much smaller than $\sum_k N_k$. Suppose $\mathbf{U}_k(i,:)$ is mapped to $\mathbf{P}(\pi_k(i),:)$. Then, we can represent the approximation of $\mathbf{X}_k(i_1, i_2)$ as follows:

$$\mathbf{X}_k(i_1, i_2) \approx \tilde{\mathbf{X}}_k(i_1, i_2) = \sum_{r=1}^{R} \mathbf{P}(\pi_k(i_1), r)\mathbf{S}(k, r)\mathbf{V}(i_2, r). \tag{2}$$

Fig. 3 shows an example of the vocabulary-based approximation of Light-IT. Thanks to $\mathbf{P}$ and $\{\pi_k\}_{k=1}^{K}$ for the first mode, the size of the compressed outputs is dramatically reduced from the output size of PARAFAC2 decomposition. The compression result of Light-IT is $\left\{\mathbf{P}, \mathbf{S}, \mathbf{V}, \{\pi_k\}_{k=1}^{K}\right\}$ where $\mathbf{V} \in \mathbb{R}^{M \times R}$ and $\mathbf{S} \in \mathbb{R}^{K \times R}$ are the factor matrices for the second and final modes, respectively. We use Huffman encoding to compress the integers in $\{\pi_k\}_{k=1}^{K}$. Refer to Theorem 1 for the analysis of the size of the compressed result.

Next, we explain the optimization process of Light-IT with its loss function. The overall training process is described in Algorithm 1. Inspired from the general vocabulary learning technique [3], we design a learning process for the vocabulary-based compression. Specifically, we co-train the temporary factor matrices $\{\mathbf{U}_k\}_{k=1}^{K}$ that aims to guide the training of $\mathbf{P}$ and obtain $\{\pi_k\}_{k=1}^{K}$ from $\mathbf{P}$. After completing the training phase, we remove the temporary matrices $\{\mathbf{U}_k\}_{k=1}^{K}$. We define the mapping function π_k as follows:

$$\pi_k(i) := \underset{j}{\operatorname{argmin}} \|\mathbf{U}_k(i,:) - \mathbf{P}(j,:)\|_F^2. \tag{3}$$

Derived from Eq. (2), the optimization problem of Light-IT is represented as follows:

$$\min_{\Theta} \sum_{k=1}^{K}\sum_{i=1}^{N_k} \|\mathbf{X}_k(i,:) - \mathbf{P}(\pi_k(i),:)(\mathbf{V} \odot \mathbf{S}(k,:))^\top\|_F^2. \tag{4}$$

where $\Theta = \left\{\mathbf{P}, \{\mathbf{U}_k\}_{k=1}^{K}\right\}$. However, Eq. (4) is not differentiable because it involves discrete functions such as the indexing operation in $\mathbf{P}(\pi_k(i),:)$ and the argmin function in Eq. (3). To employ gradient descents, we use the $\mathbf{P}'_k(i,:)$ in place of $\mathbf{P}(\pi_k(i),:)$ in Eq. (4), where $\mathbf{P}'_k(i,:)$ is defined as follows:

$$\mathbf{P}'_k(i,:) = \mathbf{U}_k(i,:) - sg(\mathbf{U}_k(i,:) - \mathbf{P}(\pi_k(i),:)), \tag{5}$$

where $sg(\cdot)$ is the stop gradient operator which is an identity function in the forward pass but drops the gradient for the input during the backward pass. Thus, $\mathbf{P}'_k(i,:)$ becomes $\mathbf{P}(\pi_k(i),:)$ in the forward pass but only backpropagates to $\mathbf{U}_k(i,:)$, computing gradients for $\mathbf{U}_k$. Additionally, we add a regularization term denoted by $\|\mathbf{P}(sg(\pi_k(i)),:) - sg(\mathbf{U}_k(i,:))\|_F^2$ into the loss, which aims to update the vocabulary matrix $\mathbf{P}$. To sum up, the loss function $\mathcal{L}$ for Light-IT is as follows:

$$\mathcal{L} = \sum_{k=1}^{K}\left(\|\mathbf{X}_k - \mathbf{P}'_k(\mathbf{V} \odot \mathbf{S}(k,:))^\top\|_F^2 + \|\mathbf{P}_k - sg(\mathbf{U}_k)\|_F^2\right), \tag{6}$$

where the i-th row of $\mathbf{P}_k$ is $\mathbf{P}(sg(\pi_k(i)),:)$, and $\mathbf{P}'_k(\mathbf{V} \odot \mathbf{S}(k,:))^\top$ is the reconstruction result $\tilde{\mathbf{X}}$. Light-IT optimizes its parameters through gradient descent, minimizing the loss in Eq. (6). Note that under the vocabulary learning mechanism [3], the matrices $\mathbf{U}_k$ and $\mathbf{P}$ are updated at the same time by a gradient descent, and the mapping function π_k is determined by them to minimize the difference (loss function in Eq. (3)) between $\mathbf{U}_k$ and $\mathbf{P}_k$.

Handling sparse irregular tensors: To compute Eq. (6), which is the main bottleneck of Light-IT, the required time is proportional to the number of entries in $\{\mathbf{X}_k\}_{k=1}^{K}$ because $\sum_k \|\mathbf{X}_k - \tilde{\mathbf{X}}_k\|_F^2$ measures the approximation error for all entries, where $\tilde{\mathbf{X}}_k = \mathbf{P}'_k(\mathbf{V} \odot \mathbf{S}(k,:))^\top$. However, for a sparse irregular tensor, it can be time-consuming to compute an error $\left(\mathbf{X}_k(i,j) - \tilde{\mathbf{X}}_k(i,j)\right)^2$ for zero entries as most of the entries that exist within the sparse tensor are zero. Thus, we propose a sparse design that efficiently computes the loss in Eq. (6). For this, we carefully reorganize the computations in $\sum_k \|\mathbf{X}_k - \tilde{\mathbf{X}}_k\|_F^2$. The approximation error of each sparse slice $\mathbf{X}_k$ is as follows:

$$\begin{aligned}
\|\mathbf{X}_k - \tilde{\mathbf{X}}_k\|_F^2 &= \sum_{i=1}^{N_k}\sum_{j=1}^{M}\left(\mathbf{X}_k(i,j) - \tilde{\mathbf{X}}_k(i,j)\right)^2 \\
&= \sum_{(i,j)\in\Omega_k} \tilde{\mathbf{X}}_k(i,j)^2 + \sum_{(i,j)\in\Omega_k^c}\left(\mathbf{X}_k(i,j) - \tilde{\mathbf{X}}_k(i,j)\right)^2 \\
&= \|\tilde{\mathbf{X}}_k\|_F^2 + \sum_{(i,j)\in\Omega_k^c}\left(\left(\mathbf{X}_k(i,j) - \tilde{\mathbf{X}}_k(i,j)\right)^2 - \tilde{\mathbf{X}}_k(i,j)^2\right),
\end{aligned} \tag{7}$$

where Ω_k and Ω_k^c are the sets of indices of zero and non-zero entries in $\mathbf{X}_k$, respectively. Based on the decomposed results, $\|\tilde{\mathbf{X}}_k\|_F^2$ is represented as follows:

$$\|\tilde{\mathbf{X}}_k\|_F^2 = \sum_{r_1=1}^{R}\sum_{r_2=1}^{R} \mathbf{S}(k,r_1)\mathbf{S}(k,r_2)\left(\mathbf{P}'_k(:,r_1)^\top\mathbf{P}_k(:,r_2)\right)\left(\mathbf{V}(:,r_1)^\top\mathbf{V}(:,r_2)\right), \tag{8}$$

where the full derivation is provided in Appendix B.1. Note that the second term of Eq. (7) takes $O(nnz(\mathbf{X}_k))$ time. The first term, $\|\tilde{\mathbf{X}}_k\|_F^2$, takes $O\left((N_k + M)R^2\right)$ time by Eq. (8), where R is the rank, and M is the second mode length of $\mathbf{X}_k$, which is much smaller $O(N_k MR)$ time for naively reconstructing all entries in $\mathbf{X}_k$. Thus, using Eq. (7) and (8), we can efficiently calculate $\sum_k \|\mathbf{X}_k - \tilde{\mathbf{X}}_k\|_F^2$ for a large sparse irregular tensor. Considering this, we analyze the time complexity of Light-IT for sparse irregular tensors in Theorem 4.

Handling higher-order irregular tensors: To handle an irregular tensor $\{\mathcal{X}_k\}_{k=1}^{K}$ with an order d higher than three, we matricize each slice $\mathcal{X}_k$ and its approximation $\tilde{\mathcal{X}}_k$ along the first mode into $\mathbf{X}_k^{(1)}$ and $\tilde{\mathbf{X}}_k^{(1)}$, respectively. We then compute the loss in Eq. (6) using the mode-1 matricizations to update the parameters. As the order d is higher, we have more order-related factor matrices $\{\mathbf{V}_i\}_{i=1}^{d-2}$, and each entry of the slice $\mathcal{X}_k$ is approximated as follows:

$$\mathcal{X}_k(i_1, \cdots, i_{d-1}) \approx \sum_{r=1}^{R} \mathbf{P}(\pi_k(i_1), r)\mathbf{V}_1(i_2, r)\cdots\mathbf{V}_{d-2}(i_{d-1}, r)\mathbf{S}(k, r). \tag{9}$$

Algorithm 1: Compression process of Light-IT

```
Input: an irregular tensor ${\{X_k\}}_{k=1}^{K}$, a number $E$ of epochs.
Output: mappings ${\{\pi\}}_{k=1}^{K}$, factor matrices P, V, and S.
[1]Initialize $U_k$
[2]Initialize P, V, and S to random values in $[0, r)$
for $e \leftarrow 1$ to $E$ do
    // Update the mappings $\pi_k$
    for $k \leftarrow 1$ to $K$ do
        for $i \leftarrow 1$ to $N_k$ do
            $\pi_k(i) \leftarrow \mathrm{argmin}_j \|U_k(i,:) - P(j,:)\|_F^2$
            $P_k(i,:) \leftarrow P(\pi_k(i),:)$
        $P'_k \leftarrow U_k - sg(U_k - P_k)$
    // Update the matrices P, V, and S
    if ${\{X_k\}}_{k=1}^{K}$ is dense then
        $\mathcal{L}$ is computed by Eq. (6)
    else if ${\{X_k\}}_{k=1}^{K}$ is sparse then
        $\mathcal{L} \leftarrow \sum_{k=1}^{K} \|\tilde{X}_k\|_F^2$ based on Eq. (8)
        // Calculate the loss terms for non-zero entries.
        for $k \leftarrow 1$ to $K$ do
            for $(i,j) \in \{(i',j') | X_k(i',j') \neq 0\}$ do
                $\mathcal{L} \leftarrow \mathcal{L} + \big(X_k(i,j) - \tilde{X}_k(i,j)\big)^2 - \tilde{X}_k(i,j)^2$
    for $k \leftarrow 1$ to $K$ do
        $\mathcal{L} \leftarrow \mathcal{L} + \|P_k - sg(U_k)\|_F^2$
    Backpropagate $\mathcal{L}$ and update P, $U_k$, V, and S
return ${\{\pi\}}_{k=1}^{K}$, P, V, and S
```

Then, we replace the following terms in Eq. (6): $\mathbf{X}_k \leftarrow \mathbf{X}_k^{(1)}$ and $\tilde{\mathbf{X}}_k \leftarrow \tilde{\mathbf{X}}_k^{(1)} = \mathbf{P}'_k\big((\odot_{i=1}^{d-2}\mathbf{V}_i) \odot \mathbf{S}(k,:)\big)^\top$, which enables Light-IT to handle higher-order irregular tensors. For the sparse design, we replace $\mathbf{V}(:,r_1)^\top\mathbf{V}(:,r_2)$ with $\prod_{i=1}^{d-2}\mathbf{V}_i(:,r_1)^\top\mathbf{V}_i(:,r_2)$ in Eq. (8). Refer to the full equations for this in Appendix B.2.

5.3 Light-IT^{++}: Extension with Core Tensor

We propose Light-IT^{++}, an expressive irregular tensor decomposition that achieves a higher accuracy than Light-IT with a little additional space. In Eq. (2), the PARAFAC2-based computation has limited expressiveness since it fails to capture relationships across multiple latent dimensions; for example, there is no multiplication between $\mathbf{P}(\pi_k(i_1), r_1)$, $\mathbf{V}(i_2, r_2)$, and $\mathbf{S}(k, r_3)$ ($r_1 \neq r_2 \neq r_3$). To increase the expression power, our idea is to incorporate a core tensor $\mathcal{G} \in \mathbb{R}^{R\times R\times R}$ into the compressed outputs and employ an approximation method used in Tucker decomposition. The approximation formula is as follows:

$$\mathbf{X}_k(i,j) \approx \sum_{r_1=1}^{R}\sum_{r_2=1}^{R}\sum_{r_3=1}^{R} \mathcal{G}(r_1,r_2,r_3)\mathbf{P}(\pi_k(i_1),r_1)\mathbf{V}(j,r_2)\mathbf{S}(k,r_3). \quad (10)$$

Note that if $\mathcal{G}$ is a diagonal tensor whose diagonal entries are all 1, Eq. (10) is equivalent to Eq. (2). Thus, all irregular tensors generated by Light-IT can be generated by Light-IT^{++} of the same rank. Light-IT^{++} can fit an irregular tensor with a more complex pattern by

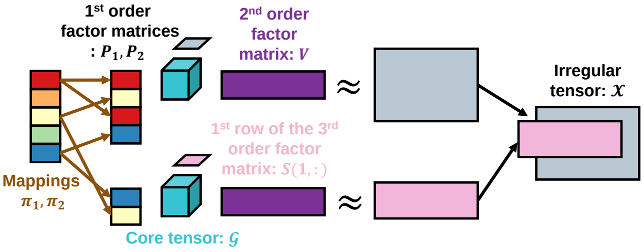

1st order factor matrices : P_1, P_2
2nd order factor matrix: V
Irregular tensor: $\mathcal{X}$
1st row of the 3rd order factor matrix: $S(1,:)$
Mappings π_1, π_2
Core tensor: $\mathcal{G}$
≈
≈

Figure 4: Approximation by Light-IT^{++}

learning the relationships between the latent dimensions (columns) of $\mathbf{P}_k$, $\mathbf{V}$, and $\mathbf{S}$ in the form of $\mathcal{G}$. To the best of our knowledge, our attempt is the first extension of Tucker decomposition to irregular tensor decomposition. The approximation process of Light-IT^{++} is illustrated in Fig. 4, and its training process is described in Algorithm 2 of Appendix B.4.

We optimize the loss function as the sum of the squared errors between the input values and their approximations in Eq. (10) (see Appendix B.3 for the detailed equation). In the training phase of Light-IT^{++}, we initialize the factor matrices $\mathbf{P}$, $\mathbf{V}$, and $\mathbf{S}$ to the corresponding outputs of Light-IT. The core tensor $\mathcal{G}$ is initialized to a diagonal tensor whose diagonal entries are all one, and the mappings ${\{\pi_k\}}_{k=1}^{K}$ from Light-IT are used without changes. To update the parameters $\mathcal{G}$, $\mathbf{P}$, $\mathbf{V}$, and $\mathbf{S}$, we employ the alternating least square (ALS) method, alternatively optimizing each parameter while fixing the others in turn[3].

Updating $\mathcal{G}$: In the ALS scheme, the optimization problem for the core tensor $\mathcal{G}$ is represented as follows:

$$\min_{\mathcal{G}} \sum_{k=1}^{K} \|\mathbf{X}_k^\top - \mathbf{V}\mathbf{G}^{(2)}(\mathbf{P}_k \otimes \mathbf{S}(k,:))^\top\|_F^2, \quad (11)$$

where ⊗ denotes Kronecker Product. Note that $\mathbf{P}_k$ is a matrix built from $\mathbf{P}$ and π_k where $\mathbf{P}_k(i,:)$ is $\mathbf{P}(\pi_k(i),:)$. This is a least squares problem, and the following solution is obtained analytically by setting its gradient to zero:

$$\mathbf{G}^{(2)} \leftarrow (\mathbf{V}^\top\mathbf{V})^\dagger\Big(\mathbf{V}^\top\sum_{k=1}^{K}\mathbf{X}_k^\top(\mathbf{P}_k \otimes \mathbf{S}(k,:))\Big)\Big(\sum_{k=1}^{K}\mathbf{P}_k^\top\mathbf{P}_k \otimes \mathbf{S}(k,:)^\top\mathbf{S}(k,:)\Big)^\dagger, \quad (12)$$

where † denotes the pseudoinverse of a matrix.

Updating P: We approach the problem for each row of $\mathbf{P}$. Assuming that each element (j,k) in a set T_i satisfies $\pi_k(j) = i$ (i.e. the j-th row of $\mathbf{X}_k$ is mapped to the i-th row of $\mathbf{P}$ by Light-IT). The problem for the i-th row of $\mathbf{P}$ is formulated as follows:

$$\min_{\mathbf{P}(i,:)} \sum_{(j,k)\in T_i} \|\mathbf{X}_k(j,:) - \mathbf{P}(i,:)\mathbf{G}^{(1)}(\mathbf{V} \otimes \mathbf{S}(k,:))^\top\|_F^2. \quad (13)$$

The above is also the least square problem, and the update formula for the i-th row of $\mathbf{P}$ is given as follows:

$$\begin{aligned}\mathbf{P}(i,:) \leftarrow &\Big(\sum_{(j,k)\in T_i}\mathbf{X}_k(j,:)(\mathbf{V} \otimes \mathbf{S}(k,:))\mathbf{G}^{(1)\top}\Big) \\ &\times \Big(\mathbf{G}^{(1)}\Big((\mathbf{V}^\top\mathbf{V}) \otimes \Big(\sum_{(j,k)\in T_i}\mathbf{S}(k,:)^\top\mathbf{S}(k,:)\Big)\Big)\mathbf{G}^{(1)\top}\Big)^\dagger.\end{aligned} \quad (14)$$

Updating V: We use the objective function in Eq. (11) but minimize the function with respect to $\mathbf{V}$ instead of $\mathcal{G}$ when updating $\mathbf{V}$. The solution on $\mathbf{V}$ is represented as follows:

[1] We use the process of earning $\mathbf{Q}_k$ in PARAFAC2-ALS illustrated in [24] for the faster training of the algorithm. After computing $\mathbf{Q}_k$, we set $\mathbf{A}_k$ to $\mathbf{Q}_k\mathbf{H}$ where $\mathbf{H}$ is sampled randomly form $[0,1)$. It allows $\mathbf{A}_k^\top\mathbf{A}_k$ to be $\mathbf{H}^\top\mathbf{H}$ for all k in $[K]$, helping Light-IT share the first mode factor matrices along the slices.

[2] If the given tensor is dense, we set $r = 0.1$; otherwise, $r = 0.01$.

[3] The update order is motivated by the HOOI algorithm [5].

$$\mathbf{V} \leftarrow \Big(\sum_{k=1}^{K} \mathbf{X}_k^{\top}(\mathbf{P}_k \otimes \mathbf{S}(k,:))\mathbf{G}^{(2)\top}\Big)\Big(\mathbf{G}^{(2)}\Big(\sum_{k=1}^{K} \mathbf{P}_k^{\top}\mathbf{P}_k \otimes \mathbf{S}(k,:)^{\top}\mathbf{S}(k,:)\Big)\mathbf{G}^{(2)\top}\Big)^{\dagger}. \tag{15}$$

Updating S: The problem for $\mathbf{S}$ is defined in a row-wise manner after vectorizing the slices and their approximations:

$$\min_{\mathbf{S}(k,:)} \|vec(\mathbf{X}_k) - \mathbf{S}(k,:)\mathbf{G}^{(3)}(\mathbf{P}_k \otimes \mathbf{V})^{\top}\|_F^2. \tag{16}$$

The optimal solution of the least square problem in Eq. (16) is represented as follows:

$$\mathbf{S}(k,:) \leftarrow \Big(vec(\mathbf{X}_k)(\mathbf{P}_k \otimes \mathbf{V})\mathbf{G}^{(3)\top}\Big)\Big(\mathbf{G}^{(3)}(\mathbf{P}_k^{\top}\mathbf{P}_k \otimes \mathbf{V}^{\top}\mathbf{V})\mathbf{G}^{(3)\top}\Big)^{\dagger}. \tag{17}$$

Handling sparse irregular tensors: In the aforementioned update formulas, one of the most dominant parts is the multiplication of a slice and the result of Kronecker product, e.g., $\mathbf{X}_k^{\top}(\mathbf{P}_k \otimes \mathbf{S}(k,:))$ in Eq. (12) and (15), $\mathbf{X}_k(j,:)(\mathbf{V} \otimes \mathbf{S}(k,:))$ in Eq. (14), and $vec(\mathbf{X}_k)(\mathbf{P}_k \otimes \mathbf{V})$ in Eq. (17). We explain the way of efficiently computing the term $\mathbf{X}_k^{\top}(\mathbf{P}_k \otimes \mathbf{S}(k,:))$; the other terms can be computed in the same manner.

Computing $\mathbf{X}_k^{\top}(\mathbf{P}_k \otimes \mathbf{S}(k,:))$ requires operations whose number is linear in the number of entries in $\mathbf{X}_k$. We reduce the cost to be linear in the number of non-zeros in $\mathbf{X}_k$ by conducting computations in units of a single entry in $\mathbf{X}_k$. Let $\mathbf{B}_k$ denotes a matrix of size $M \times R^2$, which is initialized to a zero matrix but will be $\mathbf{X}_k^{\top}(\mathbf{P}_k \otimes \mathbf{S}(k,:))$. An entry $\mathbf{X}_k(i,j)$ contributes to $\mathbf{B}_k$ as follows.

$$\mathbf{B}_k(j,:) \leftarrow \mathbf{B}_k(j,:) + \mathbf{X}_k(i,j)\big(\mathbf{P}_k(i,:) \otimes \mathbf{S}(k,:)\big). \tag{18}$$

Note that $\mathbf{B}_k$ becomes $\mathbf{X}_k^{\top}(\mathbf{P}_k \otimes \mathbf{S}(k,:))$ if we perform Eq. (18) for all entries in $\mathbf{X}_k$. Therefore, only non-zero entries of $\mathbf{X}_k$ are engaged in the computation, and the time complexity becomes linear in $O(nnz(\mathbf{X}_k))$. Refer to Theorem 6 for the detailed time complexity.

Handling higher-order irregular tensors: For a higher-order irregular tensor, an entry $\mathcal{X}_k(i_1, \ldots, i_{d-1})$ is approximated as follows:

$$\sum_{r_1=1}^{R} \cdots \sum_{r_d=1}^{R} \Big(\mathcal{G}(r_1, \ldots, r_d)\mathbf{P}(\pi_k(i_1), r_1)\prod_{j=2}^{d-1}\mathbf{V}_{j-1}(i_j, r_j)\mathbf{S}(k, r_d)\Big). \tag{19}$$

Regarding the problem formulations of Light-IT^{++}, we matricize the slice $\mathcal{X}_k$ and its approximation $\tilde{\mathcal{X}}_k$ with the decomposed results of Light-IT^{++}. All problems are least square problems when viewed from the row of a factor matrix or a factor matrix perspective, and the analytic solutions for them exist.

For example, when updating $\mathbf{V}_i$, we matricize the slices and their approximations along the i-th mode, and formulate the problem on $\mathbf{V}_i$ as follows:

$$\min_{\mathbf{V}_i} \sum_{k=1}^{K} \|\mathbf{X}_k^{(i)} - \mathbf{V}_i\mathbf{G}^{(i)}(\mathbf{P}_k \otimes \mathbf{S}(k,:) \otimes (\otimes_{j\neq i}\mathbf{V}_j))^{\top}\|_F^2. \tag{20}$$

By solving the problem in Eq. (20), the update formula on $\mathbf{V}_i$ is given as follows:

$$\begin{aligned}\mathbf{V}_i \leftarrow &\Big(\sum_{k=1}^{K} \mathbf{X}_k^{(i)}(\mathbf{P}_k \otimes \mathbf{S}(k,:) \otimes (\otimes_{j\neq i}\mathbf{V}_j))\mathbf{G}^{(i)\top}\Big)\\ &\times \Big(\mathbf{G}^{(i)}\Big(\big(\sum_{k=1}^{K} \mathbf{P}_k^{\top}\mathbf{P}_k \otimes \mathbf{S}(k,:)^{\top}\mathbf{S}(k,:)\big) \otimes \big(\otimes_{j\neq i}\, \mathbf{V}_j^{\top}\mathbf{V}_j\big)\Big)\mathbf{G}^{(i)\top}\Big)^{\dagger}.\end{aligned} \tag{21}$$

Due to the lack of spaces, all details of the problems and update formulas for $\mathcal{G}$, $\mathbf{P}$ and $\mathbf{S}$ are in Appendix B.5.

Table 2: Statistics of real-world datasets.

Name	N_{max}	N_{avg}	Size (except the 1st mode)	Order	Density
CMS	175	35.4	$284 \times 91,586$	3	5.01×10^{-3}
MIMIC-III	280	12.3	$1,000 \times 37,163$	3	7.33×10^{-3}
Korea-stock	5,270	3696.5	$88 \times 1,000$	3	9.98×10^{-1}
US-stock	7,883	3192.6	$88 \times 1,000$	3	1
Enron	554	80.6	$1,000 \times 1,000 \times 939$	4	6.93×10^{-5}
Delicious	312	16.4	$1,000 \times 1,000 \times 31,311$	4	3.97×10^{-6}
Enron_small	408	59.4	$50 \times 50 \times 759$	4	5.07×10^{-3}
Delicious_small	39	13.9	$50 \times 50 \times 1,001$	4	1.16×10^{-3}

5.4 Theoretical Analysis

We analyze the size of compressed outputs and compression time for our methods. The size of $\mathcal{X}_k$ in $\{\mathcal{X}_k\}_{k=1}^{K}$ is $N_k \times M_1 \times \cdots \times M_{d-2}$ and the maximum values among $\{N_k\}_{k=1}^{K}$ is N_{max}. We denote the rank of our method to R and the size of the vocabulary to D. The number of non-zero entries in $\mathcal{X}_k$ is denoted by $nnz(\mathcal{X}_k)$. Proofs of all theorems are available in Appendix B.6.

Size of compressed outputs: The compressed output sizes of our methods are described in Theorems 1 and 2. It is worth noting that the output size $O(\log D \sum_{k=1}^{K} N_k)$ of Huffman encoding is empirically much smaller than the size $O(\sum_{k=1}^{K} N_k R)$ of the first mode factor matrices in PARAFAC2 decomposition.

Theorem 1 (Compressed size of Light-IT). *The compressed size of Light-IT is $O(R(D + \sum_{i=1}^{d-2} M_i + K) + \log D \sum_{k=1}^{K} N_k)$.*

Theorem 2 (Compressed size of Light-IT^{++}). *The compressed size of Light-IT^{++} is $O(R(D + \sum_{i=1}^{d-2} M_i + K) + \log D \sum_{k=1}^{K} N_k + R^d)$.*

Compression time: We analyze the time for a single epoch of Light-IT and Light-IT^{++}. Note that there are terms that include the number of entries (i.e. $\sum_{k=1}^{K} N_K \prod_{i=1}^{d-2} M_i$) in the complexities of the methods for dense irregular tensors and the terms that include the number of non-zero entries (i.e. $\sum_{k=1}^{K} nnz(\mathcal{X}_k)$) in the complexities of the methods for spares irregular tensors.

Theorem 3 (Compression time of Light-IT). *The compression time of Light-IT is $O\big((\prod_{i=1}^{d-2} M_i)(\sum_{k=1}^{K} N_k)R + (\sum_{k=1}^{K} N_k)DR\big)$.*

Theorem 4 (Compression time of Light-IT for sparse irregular tensors). *The compression time of the spares version of Light-IT is $O\big((\sum_{k=1}^{K} N_k + \sum_{i=1}^{d-2} M_i)R^2 + \sum_{k=1}^{K} nnz(\mathcal{X}_k)dR + (\sum_{k=1}^{K} N_k)DR\big)$.*

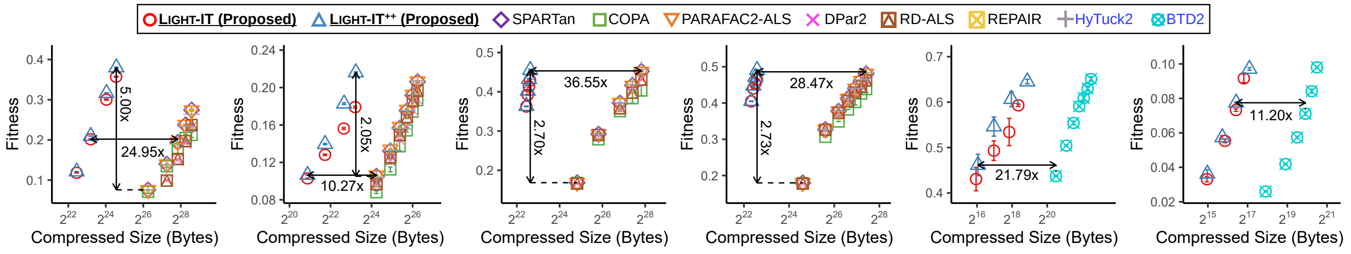
Theorem 5 (Compression time of Light-IT^{++}). *The compression time of Light-IT^{++} is $O\big(R^{3(d-2)} + KR^4 + R^6 + R^{2d-1}D + d(\sum_{k=1}^{K} N_k)\prod_{i=1}^{d-2} M_i)R^{d-1} + KR^{2d-1}\big)$.*

Theorem 6 (Compression time of Light-IT^{++} for spare irregular tensors). *The compression time of the sparse version of Light-IT^{++} is $O\big(\prod_{i=1}^{d-2} M_i\, R^4 + \sum_{k=1}^{K} nnz(\mathcal{X}_k)\, R^d d + R^{3(d-2)} + (\prod_{i=1}^{d-2} M_i)R^d + KR^4 + R^6 + R^{2d-1}D + \sum_{i=1}^{d-2} M_i K R^d + d(\sum_{k=1}^{K} N_k)R^2 + KR^{2d-1}\big)$.*

6 EXPERIMENTS

We conduct experiments to answer the following questions.

Q1. **Compression Performance:** How do Light-IT and Light-IT^{++} compactly and accurately compress irregular tensors compared to their competitors?

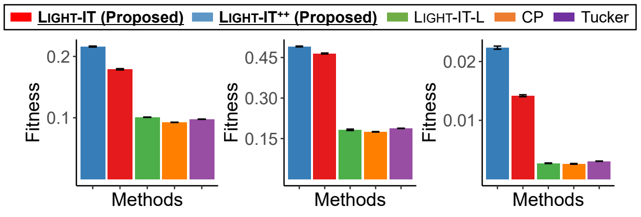


(a) CMS (b) MIMIC-III (c) Korea-stock (d) US-stock (e) Enron_small (f) Delicious_small

Figure 5: Light-IT and Light-IT^{++} concisely and accurately compress irregular tensors. Notably, the output size of Light-IT^{++} is up to 37× smaller than that of the most compact baseline, showing a similar fitenss. Light-IT^{++} shows up to 5× higher fitness than the most accurate baseline, even compressing to smaller outputs. We add the results of HyTuck2 and BTD2 on 3-order tensors and 4-order tensors, respectively. Note that HyTuck2 is designed for 3-order tensors, while BTD2 is tailored for 4-order tensors. We reduced the size of the 4-order tensors by sampling the second-order and third-order indices (i.e from Enron to Enron_small and Delicious to Delicious_small) because O.O.M occurred when running BTD2 on the original tensors. The sizes of Enron_small and Delicious_small are in Table 2.

Q2. **Ablation Study:** How does each component of our methods affect the accuracy and time?
Q3. **Scalability:** Is the compression time of our methods linearly scalable to the number of (non-zero) entries in the input tensor?
Q4. **Total Compression Time:** How efficiently do our methods compress an irreuglar tensor in terms of compression speed?
Q5. **Vocabulary Size:** What is the optimal size of the vocabulary?

The answer of the question 5 is in Appendix C.8.



(a) MIMIC-III (b) US-stock (c) Delicious

Figure 6: Our ideas on vocabulary-based compression and extension with core tensor are effective for compression.

6.1 Experimental Specifications

Machines: The experiments except the ablation study and scalability test were conducted on a workstation having an RTX 3090Ti GPU and 128GB RAM. Our methods for the ablation study and scalability test in Section 6.3 and 6.4 were conducted on a machine with an RTX 2080Ti GPU and 128GB RAM. The competitors that do not require a GPU were executed on a machine with an i5-9600K (6 cores) and 64GB RAM. The accuracies and compression ratios of the methods are independent of the machine's specifications.
Datasets: We used 6 public real-world datasets whose statistics are summarized in Table 2. Note that the datasets include sparse, dense, and higher-order irregular tensors. Refer to Appendix C.3 for the semantics, sources, and preprocessing steps of the datasets.
Competitors: We use 6 state-of-the-art methods for irregular tensor decomposition as our competitors. PARAFAC2-ALS [16], RD-ALS [4], and DPar2[4] [14] are targeting at dense irregular tensors, and COPA[5] [1], SPARTan[6] [24], and REPAIR[7] [25] are tailored for sparse irregular tensors. Note that no method is designed for irregular tensors with orders higher than three. Refer to Section 2 for the explanations of the competitors. Since the official implementations of PARAFAC2-ALS and RD-ALS have not been released, we implemented them according to the corresponding papers. For the other competitors, we used the open-sourced implementations provided by the authors. All of them were implemented in MATLAB R2020a.
Evaluation metrics: We evaluated the accuracy of a compression result by measuring fitness, defined as follows:

$$fitness = 1 - \sqrt{\left(\sum_{k=1}^{K}\|\mathcal{X}_k - \bar{\mathcal{X}}_k\|_F^2\right) / \sum_{k=1}^{K}\|\mathcal{X}_k\|_F^2},$$

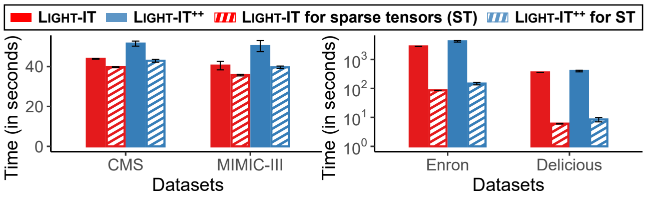
where $\bar{\mathcal{X}}_k$ is the approximation of a given $\mathcal{X}_k$ by a compression method, and it is tailored for measuring the accuracy of irregular tensor compression, extending from the fitness applied to regular tensors. The concept of fitness has been widely used in [17, 23] for measuring the accuracy of tensor approximation (or compression). It is smaller than 1, with higher fitness indicating more accurate compression. The compressed size was measured by counting the number of parameters in the compression result in bytes. For our methods, we further applied Huffman encoding to compress the integer numbers in the mappings $\{\pi_k\}_{k=1}^{K}$. More details on the compressed size are in Appendix C.1.
Training details: Training details are provided in Appendix C.2.

6.2 Compression Performance

We evaluated the compression performances of our methods and competitors by comparing the trade-offs between accuracy and compressed size. In this experiment, we measured the fitness of each method by varying the compressed size in bytes. We compared the compressed sizes (or accuracies) of our methods with those of their competitors at a similar level of accuracy (or size). We set the size of the vocabulary of our methods to the largest mode length of the first mode in a given irregular tensor. The values of the other hyperparameters of all methods are provided in Appendx C.4.

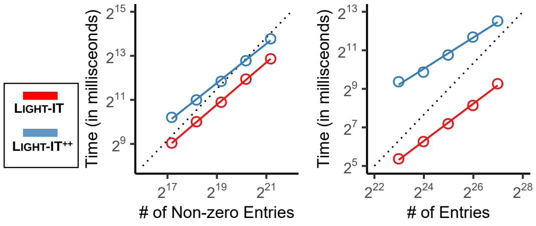
As shown in Figure 5, Light-IT and Light-IT^{++} significantly outperform their competitors by providing the best trade-off between accuracy and size. Specifically, the compressed size of Light-IT^{++} is

[4] https://datalab.snu.ac.kr/dpar2
[5] https://github.com/aafshar/COPA
[6] https://github.com/kperros/SPARTan
[7] https://github.com/Emory-AIMS/Repair/tree/master



(a) 3-order irregular tensors (b) 4-order irregular tensors

Figure 7: For sparse irregular tensors, our methods leveraging the sparsity are faster than the dense counterparts.



(a) 4-order sparse tensors (b) 4-order dense tensors

Figure 8: Compression time of our methods increases (sub-)linearly to the number of non-zero entries or entries. The results for 3-order irregular tensors are in Appendix C.6.

36.55× smaller than that of the most compact baseline, showing a similar approximation accuracy in the `Korea-stock` dataset. In the `CMS` dataset, Light-IT^{++} shows 5.02× higher fitness than the most accurate baseline. Note that the maximum settings of our methods yield much smaller compressed sizes than the minimum settings of the other methods. As expected, Light-IT^{++} achieves higher accuracy than Light-IT, albeit at the cost of slightly more compressed size in most datasets. Note that no competitor can be run on the `Enron` and `Delicious` datasets since they form 4-dimensional irregular tensors, respectively. The experiments on REPAIR ran out of memory in all 3-order tensors, except for `MIMIC-III`.

6.3 Ablation Studies

We compared Light-IT and Light-IT^{++} with the following methods to check the effectiveness of our ideas.

- Light-IT-L: a variant of Light-IT that directly maps the i-th row in a slice to the i-th row of the vocabulary matrix without **l**earning the mapping functions.
- CP and Tucker: CP and Tucker decompositions of a regular tensor obtained by zero-padding an irregular tensor.
- Light-IT and Light-IT^{++} for ST: sparse versions of our methods, leveraging sparsity for irregular sparse tensors (ST).

We used the same rank for all methods, and refer to Appendix C.5 for other details and results.

Vocabulary-based compression: Despite being under the same vocabulary-based scheme, Light-IT-L performs worse than Light-IT, as shown in Fig. 6, indicating it is crucial to accurately learn the mappings, achieved by our approach. In addition, Light-IT (or Light-IT^{++}) outperforms CP (or Tucker), emphasizing the effectiveness of the vocabulary-based compression over the decompositions on zero-padded regular tensors.

Extension with core tensor: As Light-IT^{++} outperforms Light-IT at the same rank, our extension with the core tensor provides enhanced expressiveness for compression.

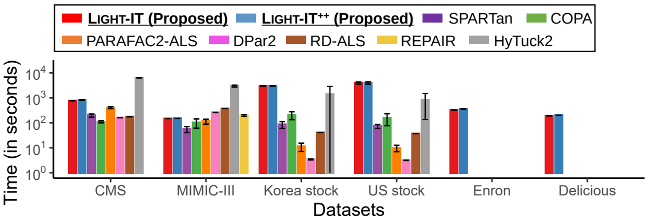


Figure 9: Total time for compressing irregular tensors.

Sparse design: As shown in Fig. 7, the compression time required by both Light-IT and Light-IT^{++} decreases when using the sparse design compared to their dense counterparts. The difference is distinct in the `Enron` and `Delicious` datasets, which are much sparser than the others, indicating that leveraging the sparsity plays an important role in the efficient compression of irregular sparse tensors.

6.4 Scalability

We investigated the scalability of Light-IT and Light-IT^{++} concerning the compression (training) time per epoch w.r.t. the number of non-zero entries in sparse irregular tensors and the number of entries in dense irregular tensors, respectively. For sparse irregular tensors, we fixed their sizes while varying the densities by a factor of 2. For dense irregular tensors, we varied their sizes by a factor of 2. All non-zero entries were randomly sampled from $[0, 1)$. More details on sizes and densities are provided in Appendix C.6.

Our methods exhibit near-linear scalability with the number of non-zero entries in sparse irregular tensors, as shown in Fig. 8(a), aligning with the analysis in Theorems 4 and 6 where it asymptotically dominates the other terms. For dense irregular tensors, as shown in Fig. 8(b), the compression time of our methods sublinearly increases to the number of entries, in line with the analysis presented in Theorems 3 and 5. The results for 3-order irregular tensors are in Appendix C.6.

6.5 Total Compression Time

We evaluated the total compression time of all methods on each dataset under parameter settings that yield similar accuracy (refer to Appendix C.7 for details). As shown in Fig. 9, our methods require more time than the competitors for compressing 3-order irregular tensors, except for `MIMIC-III`. The main cause is learning mappings, and despite slowing down our methods, it is worthwhile for effectively compressing irregular tensors. Notably, only our methods can handle 4-order irregular tensors such as `Enron` and `Delicious`, whereas the others are limited to 3-order irregular tensors. Another observation is that Light-IT^{++} spends a little more time than Light-IT in expense of a higher accuracy.

7 CONCLUSION

This paper proposes novel decompositions of an irregular tensor, Light-IT and Light-IT^{++}, specialized for compression. We invented a vocabulary matrix for reducing the size of the first mode factor matrices of PARAFAC2 decomposition (Light-IT). We devised Light-IT^{++} that includes a Tucker operation for enhancing the accuracy. For the same rank, Light-IT is faster, but Light-IT^{++} is more accurate with a small increase in the number of parameters. With these contributions, our methods have the following advantages:

- **Compact:** The compressed output of our method is up to 37× smaller than that of the most compact baseline, providing a similar approximation error.
- **Accurate:** Our method with a smaller compressed output offers up to 5× better fitness than the most precise baseline.
- **Versatile:** Our methods exploit the sparsity of sparse irregular tensors for reducing the compression time. Furthermore, they successfully compress irregular tensors of any order.

Reproducibility: The code and datasets are available at https://anonymous.4open.science/r/Light-IT-EF02/.

REFERENCES

[1] Ardavan Afshar, Ioakeim Perros, Evangelos E Papalexakis, Elizabeth Searles, Joyce Ho, and Jimeng Sun. 2018. COPA: Constrained PARAFAC2 for sparse & large datasets. In *CIKM*.

[2] Brett W Bader, Michael W Berry, and Murray Browne. 2008. Discussion tracking in Enron email using PARAFAC. In *Survey of Text Mining II: Clustering, Classification, and Retrieval*. Springer, 147–163.

[3] Ting Chen, Lala Li, and Yizhou Sun. 2020. Differentiable product quantization for end-to-end embedding compression. In *ICML*.

[4] Yao Cheng and Martin Haardt. 2019. Efficient computation of the PARAFAC2 decomposition. In *ACSCC*. IEEE.

[5] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. 2000. On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM journal on Matrix Analysis and Applications* 21, 4 (2000), 1324–1342.

[6] Petros Drineas, Ravi Kannan, and Michael W Mahoney. 2006. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. *SIAM J. Comput.* 36, 1 (2006), 184–206.

[7] Gene H Gloub and Charles F Van Loan. 1996. Matrix computations. *Johns Hopkins Universtiy Press, 3rd edtion* (1996).

[8] Olaf Görlitz, Sergej Sizov, and Steffen Staab. 2008. PINTS: peer-to-peer infrastructure for tagging systems.. In *IPTPS*. 19.

[9] Ekta Gujral, Georgios Theocharous, and Evangelos E Papalexakis. 2020. Spade: S treaming pa rafac2 de composition for large datasets. In *SDM*. SIAM.

[10] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review* 53, 2 (2011), 217–288.

[11] Per Christian Hansen. 1987. The truncated SVD as a method for regularization. *BIT Numerical Mathematics* 27 (1987), 534–553.

[12] Richard A Harshman et al. 1972. PARAFAC2: Mathematical and technical notes. *UCLA working papers in phonetics* 22, 3044 (1972), 122215.

[13] Frank L Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *J. Math. Phys* 6, 1-4 (1927), 164–189.

[14] Jun-Gi Jang and U Kang. 2022. Dpar2: Fast and scalable parafac2 decomposition for irregular dense tensors. In *ICDE*. IEEE.

[15] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3, 1 (2016), 1–9.

[16] Henk AL Kiers, Jos MF Ten Berge, and Rasmus Bro. 1999. PARAFAC2—Part I. A direct fitting algorithm for the PARAFAC2 model. *Journal of Chemometrics: A Journal of the Chemometrics Society* 13, 3-4 (1999), 275–294.

[17] Tamara Gibson Kolda. 2006. *Multilinear operators for higher-order decompositions.* Technical Report. Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA (United States).

[18] Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.

[19] Tamara G Kolda, Brett W Bader, and Joseph P Kenny. 2005. Higher-order web link analysis using multilinear algebra. In *ICDM*.

[20] Taehyung Kwon, Jihoon Ko, Jinhong Jung, and Kijung Shin. 2023. NeuKron: Constant-Size Lossy Compression of Sparse Reorderable Matrices and Tensors. In *WWW*.

[21] Taehyung Kwon, Jihoon Ko, Jinhong Jung, and Kijung Shin. 2023. TensorCodec: Compact Lossy Compression of Tensors without Strong Data Assumptions. In *ICDM*.

[22] Ivan V Oseledets. 2011. Tensor-train decomposition. *SIAM Journal on Scientific Computing* 33, 5 (2011), 2295–2317.

[23] Ioakeim Perros, Evangelos E Papalexakis, Haesun Park, Richard Vuduc, Xiaowei Yan, Christopher Defilippi, Walter F Stewart, and Jimeng Sun. 2018. Sustain: Scalable unsupervised scoring for tensors and its application to phenotyping. In *KDD*.

[24] Ioakeim Perros, Evangelos E Papalexakis, Fei Wang, Richard Vuduc, Elizabeth Searles, Michael Thompson, and Jimeng Sun. 2017. SPARTan: Scalable PARAFAC2 for large & sparse data. In *KDD*.

[25] Yifei Ren, Jian Lou, Li Xiong, and Joyce C Ho. 2020. Robust irregular tensor factorization and completion for temporal health data analysis. In *CIKM*.

[26] Jitesh Shetty and Jafar Adibi. 2004. The Enron email dataset database schema and brief statistical report. *Information sciences institute technical report, University of Southern California* 4 (2004).

[27] Shaden Smith, Jee W. Choi, Jiajia Li, Richard Vuduc, Jongsoo Park, Xing Liu, and George Karypis. 2017. *FROSTT: The Formidable Repository of Open Sparse Tensors and Tools.* http://frostt.io/

[28] Jimeng Sun, Yinglian Xie, Hui Zhang, and Christos Faloutsos. 2007. Less is more: Compact matrix decomposition for large sparse graphs. In *SDM*.

[29] Yu Sun, Nicholas Jing Yuan, Yingzi Wang, Xing Xie, Kieran McDonald, and Rui Zhang. 2016. Contextual intent tracking for personal assistants. In *KDD*.

[30] Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31, 3 (1966), 279–311.

[31] M Alex O Vasilescu and Demetri Terzopoulos. 2002. Multilinear analysis of image ensembles: Tensorfaces. In *ECCV*.

[32] M Alex O Vasilescu and Demetri Terzopoulos. 2003. Multilinear subspace analysis of image ensembles. In *ICPR*.

[33] Barry M Wise, Neal B Gallagher, and Elaine B Martin. 2001. Application of PARAFAC2 to fault detection and diagnosis in semiconductor etch. *Journal of Chemometrics: A Journal of the Chemometrics Society* 15, 4 (2001), 285–298.

[34] 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*.

A TENSOR NOTATIONS AND OPERATIONS

A.1 Matricization

A matricization of a tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_d}$ is defined as follows. Let the ordered sets $\mathcal{R} = \{r_1, \cdots, r_L\}$ and $\mathcal{C} = \{c_1, \cdots, c_M\}$ be a partitioning of the modes $\mathcal{D} = \{1, \cdots, d\}$. The mode-$r_1, \cdots, r_L$ maticization of $\mathcal{X}$, $\mathbf{X}^{(r_1,\cdots,r_L)}$, is a matrix of size $\prod_{r \in \mathcal{R}} N_r \times \prod_{c \in C} N_c$. Specifically, an entry of $\mathcal{X}$, $\mathcal{X}(i_1, i_2, \cdots, i_d)$, is mapped to $\mathbf{X}^{(r_1,\cdots,r_L)}$ (J, K) where

$$J = \sum_{l=1}^{L} \left(N_{r_l} \prod_{l'=1}^{l-1} N_{r_{l'}} \right) \text{ and } K = \sum_{m=1}^{M} \left(N_{c_m} \prod_{m'=1}^{m-1} N_{c_{m'}} \right).$$

A.2 Kronecker Product and Khatri-rao Product

Kronecker product. Given two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$, the matrix $\mathbf{C} \in \mathbb{R}^{IK \times JL}$ is the result of Kronecker product between $\mathbf{A}$ and $\mathbf{B}$:

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix} \tag{22}$$

where a_{ij} is the (i, j)th element of the matrix $\mathbf{A}$.

Khatri-rao product. Khatri-rao product is a column-wise Kronecker product. Given two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times J}$ which have the same column size, the Khatri-rao product produces the matrix $\mathbf{C} \in \mathbb{R}^{IK \times J}$:

$$\mathbf{C} = \mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{A}(:,1) \otimes \mathbf{B}(:,1) & \cdots & \mathbf{A}(:,J) \otimes \mathbf{B}(:,J) \end{bmatrix} \tag{23}$$

where $\mathbf{A}(:, j)$ and $\mathbf{B}(:, j)$ are the jth column vectors of $\mathbf{A}$ and $\mathbf{B}$, respectively.

B SUPPLEMENTS FOR PROPOSED METHODS

B.1 Computing Loss of Light-IT for Sparse Irregular Tensors

$\|\mathbf{X}'_k\|_F^2$ can be efficiently computed by the following lemma:

Lemma 1. *$\|\mathbf{X}'_k\|_F^2$ is equal to Eq. (24)*

$$\sum_{r_1=1}^{R} \sum_{r_2=1}^{R} \mathbf{S}(k,r_1)\mathbf{S}(k,r_2)\Big(\mathbf{P}'_k(:,r_1)^\top \mathbf{P}_k(:,r_2)\Big)\Big(\mathbf{V}(:,r_1)^\top \mathbf{V}(:,r_2)\Big) \tag{24}$$

Proof. Assuming that the vectorization of $\mathbf{X}'_k$ is denoted by $vec(\mathbf{X}'_k)$, the following equations hold.

$$\begin{aligned} \|\mathbf{X}'_k\|_F^2 &= vec(\mathbf{X}'_k)^\top vec(\mathbf{X}'_k) \\ &= \left(\sum_{r_1=1}^{R} (\mathbf{P}'_k(:,r_1) \odot \mathbf{V}(:,r_1))\mathbf{S}(k,r_1) \right)^\top \left(\sum_{r_2=1}^{R} (\mathbf{P}'_k(:,r_2) \odot \mathbf{V}(:,r_2))\mathbf{S}(k,r_2) \right) \\ &= \sum_{r_1=1}^{R} \sum_{r_2=1}^{R} \left(\mathbf{S}(k,r_1)\mathbf{S}(k,r_2)\Big(\mathbf{P}'_k(:,r_1)^\top \mathbf{P}_k(:,r_2)\Big)\Big(\mathbf{V}(:,r_1)^\top \mathbf{V}(:,r_2)\Big) \right) \end{aligned} \tag{25}$$

The final formula in Eq. (25) is the same with Eq. (24). □

B.2 Higher-order Version of Light-IT

The loss function in Eq. (6) becomes Eq. (26) when the order of an irregular tensor is higher than three.

$$\sum_{k=1}^{K} \Big(\|\mathbf{X}_k - \mathbf{P}'_k((\odot_{i=1}^{d-2}\mathbf{V}) \odot \mathbf{S}(k,:))^\top\|_F^2 + \|\mathbf{P}_k - sg(\mathbf{U}_k)\|_F^2 \Big). \tag{26}$$

The right hand side of Eq. (8) becomes Eq. (27)

$$\sum_{r_1=1}^{R} \sum_{r_2=1}^{R} \mathbf{S}(k,r_1)\mathbf{S}(k,r_2)\Big(\mathbf{P}'_k(:,r_1)^\top \mathbf{P}_k(:,r_2)\Big)\Big(\prod_{i=1}^{d-2} \mathbf{V}_i(:,r_1)^\top \mathbf{V}_i(:,r_2)\Big). \tag{27}$$

B.3 Computing Loss of Light-IT^{++}

We use the following formula when computing the square errors of Light-IT^{++} for fitness.

$$\sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{P}_k\mathbf{G}^{(1)}(\mathbf{V} \otimes \mathbf{S}(k,:))^\top\|_F^2. \tag{28}$$

When the input tensor is sparse, we exploit its sparsity to accelerate computation speed. We reformulate the loss function in Eq. (28) as follows using a similar approach in Eq. (7),

$$\sum_{k=1}^{K} \|\tilde{\mathbf{X}}_k\|_F^2 + \sum_{k=1}^{K} \sum_{(i,j)\in\Omega_k^c} ((\mathbf{X}_k(i,j) - \tilde{\mathbf{X}}_k(i,j))^2 - \tilde{\mathbf{X}}_k(i,j)^2), \tag{29}$$

where the $\tilde{\mathbf{X}}_k$ is $\mathbf{P}_k\mathbf{G}^{(1)}(\mathbf{V} \otimes \mathbf{S}(k,:))^\top$ and Ω_k^c is the set of non-zero entries in $\mathbf{X}_k$. By Lemma 2, $\sum_{k=1}^{K} \|\tilde{\mathbf{X}}_k\|_F^2$ can be quickly computed.

Lemma 2. *$\sum_{k=1}^{K} \|\tilde{\mathbf{X}}_k\|_F^2$ in Eq. (29) is equal to Eq. (30).*

$$sum\Big(\Big(\mathbf{V}\mathbf{G}^{(2)} \sum_{k=1}^{K} \mathbf{P}_k^\top \mathbf{P}_k \otimes \mathbf{S}(k,:)^\top \mathbf{S}(k,:)\Big) * (\mathbf{V}\mathbf{G}^{(2)})\Big). \tag{30}$$

Note that $sum(\cdot)$ denotes the sum of all entries in a matrix, and $*$ denotes an elementwise product.

Proof. The following equations hold.

$$\begin{aligned} \sum_{k=1}^{K} \|\tilde{\mathbf{X}}_k\|_F^2 &= \sum_{k=1}^{K} tr(\tilde{\mathbf{X}}_k^\top \tilde{\mathbf{X}}_k) \\ &= \sum_{k=1}^{K} tr(\mathbf{V}\mathbf{G}^{(2)}(\mathbf{P}_k \otimes \mathbf{S}(k,:))^\top (\mathbf{P}_k \otimes \mathbf{S}(k,:))\mathbf{G}^{(2)\top}\mathbf{V}^\top) \\ &= \sum_{k=1}^{K} tr(\mathbf{V}\mathbf{G}^{(2)}(\mathbf{P}_k^\top \mathbf{P}_k \otimes \mathbf{S}(k,:)^\top \mathbf{S}(k,:))\mathbf{G}^{(2)\top}\mathbf{V}^\top) \\ &= tr\Big(\mathbf{V}\mathbf{G}^{(2)}\Big(\sum_{k=1}^{K} \mathbf{P}_k^\top \mathbf{P}_k \otimes \mathbf{S}(k,:)^\top \mathbf{S}(k,:)\Big)\mathbf{G}^{(2)\top}\mathbf{V}^\top\Big) \end{aligned} \tag{31}$$

The final formula in Eq. (31) is the same as Eq. (30). Note that $\tilde{\mathbf{X}}_k^\top$ can be also written as $\mathbf{V}\mathbf{G}^{(2)}(\mathbf{P}_k \otimes \mathbf{S}(k,:))^\top$. □

Computation of Eq. (30) is much faster than naively summing squares of all entries in $\{\tilde{\mathcal{X}}_k\}_{k=1}^{K}$. Computation of the second term in Eq. (29) takes time linear in the number of non-zero entries in

Algorithm 2: Compression process of Light-IT^{++}

```
Input: an irregular tensor $\{\mathbf{X}_k\}_{k=1}^{K}$
Output: mappings $\{\pi\}_{k=1}^{K}$, a vocabulary matrix $\mathbf{P}$,
        factor matrices $\mathbf{V}$ and $\mathbf{S}$, a core-tensor $\mathcal{G}$
Initialize $\{\pi\}_{k=1}^{K}$, $\mathbf{P}$, $\mathbf{V}$, and $\mathbf{S}$ with Algorithm 1
[8]Initialize $\mathcal{G}$
while fitness does not converge do
    Update $\mathcal{G}$ with Eq. (12)
    for $i \leftarrow 1$ to $D$ do
        Update $\mathbf{P}(i,:)$ with Eq. (14)
    Update $\mathcal{G}$ with Eq. (12)
    Update $\mathbf{V}$ with Eq. (15)
    Update $\mathcal{G}$ with Eq. (12)
    for $i \leftarrow 1$ to $K$ do
        Update $\mathbf{S}(i,:)$ with Eq. (17)
    fitness $\leftarrow 1 - \frac{\sqrt{\sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{P}_k \mathbf{G}^{(1)}(\mathbf{V} \otimes \mathbf{S}(k,:))\|_F^2}}{\sqrt{\sum_{k=1}^{K} \|\mathbf{X}_k\|_F^2}}$
return $\{\pi\}_{k=1}^{K}$, $\mathbf{P}$, $\mathbf{V}$, $\mathbf{S}$, and $\mathcal{G}$
```

$\{\mathbf{X}_k\}_{k=1}^{K}$. When compressing an irregular tensor whose order is higher than three, Eq. (28) becomes Eq (32).

$$\sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{P}_k \mathbf{G}^{(1)}((\otimes_{i=1}^{d-2}\mathbf{V}_i) \otimes \mathbf{S}(k,:))^{\top}\|_F^2. \tag{32}$$

Eq. (30) is altered as follows:

$$sum\Big(\big((\otimes_{i=1}^{d-2}\mathbf{V}_i)\mathbf{G}^{(2)} \sum_{k=1}^{K} \mathbf{P}_k^{\top}\mathbf{P}_k \otimes \mathbf{S}(k,:)^{\top}\mathbf{S}(k,:)\big) * ((\otimes_{i=1}^{d-2}\mathbf{V}_i)\mathbf{G}^{(2)})\Big). \tag{33}$$

The time complexity for a total calculation during an epoch is in Theorem 5 and 6.

B.4 Pseudocode of Light-IT^{++}

The pseudocode of Light-IT^{++} is in Algorithm 2

B.5 Higher Order Version of Light-IT^{++}

Update $\mathcal{G}$: Eq. (11) is changed as follows.

$$\min_{\mathcal{G}} \sum_{k=1}^{K} \|\mathbf{X}_k^{(2,\ldots,d-1)} - (\otimes_{i=1}^{d-2}\mathbf{V})\mathbf{G}^{(2,\ldots,d-1)}(\mathbf{P}_k \otimes \mathbf{S}(k,:))^{\top}\|_F^2, \tag{34}$$

Remark that $\mathbf{X}_k^{(2,\ldots,d-1)}$ is the mode-$(2,\ldots,d-1)$ matricization of $\mathcal{X}_k$ (see Appendix A.1). Eq. (12) becomes Eq. (35).

$$\mathbf{G}^{(2)} \leftarrow (\otimes_{i=1}^{d-2}\mathbf{V}_i^{\top}\mathbf{V}_i)^{\dagger}\Big((\otimes_{i=1}^{d-2}\mathbf{V}_i)^{\top} \sum_{k=1}^{K} \mathbf{X}_k^{(2,\cdots,d-1)}(\mathbf{P}_k \otimes \mathbf{S}(k,:)\Big)$$
$$\times \Big(\sum_{k=1}^{K} \mathbf{P}_k^{\top}\mathbf{P}_k \otimes \mathbf{S}(k,:)^{\top}\mathbf{S}(k,:)\Big)^{\dagger}, \tag{35}$$

Update P: Eq. (13) is changed as follows.

$$\min_{\mathbf{P}(i,:)} \sum_{(j,k)\in T_i} \|\mathbf{X}_k^{(1)}(j,:) - \mathbf{P}(i,:)\mathbf{G}^{(1)}\big((\otimes_{i=1}^{d-2}\mathbf{V}_i) \otimes \mathbf{S}(k,:)\big)^{\top}\|_F^2. \tag{36}$$

Eq. (14) becomes Eq. (37).

$$\mathbf{P}(i,:) \leftarrow \Big(\sum_{(j,k)\in T_i} \mathbf{X}_k^{(1)}(j,:)((\otimes_{i=1}^{d-2}\mathbf{V}_i) \otimes \mathbf{S}(k,:))\mathbf{G}^{(1)\top}\Big)$$
$$\times \Big(\mathbf{G}^{(1)}\Big((\otimes_{i=1}^{d-2}\mathbf{V}_i^{\top}\mathbf{V}_i) \otimes \big(\sum_{(j,k)\in T_i} \mathbf{S}(k,:)^{\top}\mathbf{S}(k,:)\big)\Big)\mathbf{G}^{(1)\top}\Big)^{\dagger}. \tag{37}$$

Update V: The higher-order version of Eq. (15) is Eq. (21).
Update S: Eq. (16) is changed as follows.

$$\min_{\mathbf{S}(k,:)} \|vec(\mathcal{X}_k) - \mathbf{S}(k,:)\mathbf{G}^{(d)}(\mathbf{P}_k \otimes (\otimes_{i=1}^{d-2}\mathbf{V}_i))^{\top}\|_F^2. \tag{38}$$

Eq. (17) becomes Eq. (39)

$$\mathbf{S}(k,:) \leftarrow \Big(vec(\mathcal{X}_k)(\mathbf{P}_k \otimes (\otimes_{i=1}^{d-2}\mathbf{V}_i))\mathbf{G}^{(d)\top}\Big)$$
$$\times \Big(\mathbf{G}^{(d)}(\mathbf{P}_k^{\top}\mathbf{P}_k \otimes (\otimes_{i=1}^{d-2}\mathbf{V}_i^{\top}\mathbf{V}_i))\mathbf{G}^{(d)\top}\Big)^{\dagger}. \tag{39}$$

B.6 Theoretical Analysis

We assume that all equations in the main paper are written in the version that can handle a tensor of any order.

Theorem 1 (Compressed size of Light-IT). *The compressed size of Light-IT is $O(R(D + \sum_{i=1}^{d-2} M_i + K) + \log D \sum_{k=1}^{K} N_k)$.*

Proof. The matrix $\mathbf{P}$ requires $O(DR)$, $\{\mathbf{V}_i\}_{i=1}^{d-2}$ require $O(\sum_{i=2}^{d-2} M_i R)$, and $\mathbf{S}$ requires $O(KR)$. The mappings $\{\pi_k\}_{k=1}^{K}$ require $O(\log D)$ bits per index of the first mode when the integers follow a uniform distribution, which is the case that the compression result of Huffman encoding is the largest. Thus, $\{\pi_k\}_{k=1}^{K}$ requires $O(\log N_{max} \sum_{k=1}^{K} N_k)$ bits. Hence, the compressed size is $O(R(D + \sum_{i=1}^{d-2} M_i + K) + \log D \sum_{k=1}^{K} N_k)$. □

Theorem 2 (Compressed size of Light-IT^{++}). *The compressed size of Light-IT^{++} is $O(R(D + \sum_{i=1}^{d-2} M_i + K) + \log D \sum_{k=1}^{K} N_k + R^d)$.*

Proof. By Theorem 1, the sum of sizes of $\mathbf{P}$, $\{\mathbf{V}_i\}_{i=1}^{d-2}$, $\mathbf{S}$, $\{\pi_k\}_{k=1}^{K}$ is $O(R(D + \sum_{i=1}^{d-2} M_i + K) + \log D \sum_{k=1}^{K} N_k)$. The core tensor $\mathcal{G}$ takes $O(R^d)$ parameters. Therefore, the total number of parameters is $O(R(D + \sum_{i=1}^{d-2} M_i + K) + \log D \sum_{k=1}^{K} N_k + R^d)$. □

Theorem 3 (Compression time of Light-IT). *The compression time of Light-IT is $O\big((\prod_{i=1}^{d-2} M_i)(\sum_{k=1}^{K} N_k)R + (\sum_{k=1}^{K} N_k)DR\big)$.*

Proof. Computing $\pi_k(i)$ according to Eq. (3) for all k in $\{1,\cdots,K\}$ and i in $[N_k]$ takes $O((\sum_{k=1}^{K} N_k)DR)$ time. Given $\mathbf{P}'_k$ and $((\odot_{i=1}^{d-2}\mathbf{V}_i) \odot \mathbf{S}(k,:))$, computing $\mathbf{P}'_k((\odot_{i=1}^{d-2}\mathbf{V}_i) \odot \mathbf{S}(k,:))^{\top}$ in Eq. (26) for all $k \in \{1,\ldots,K\}$ is the most dominant part during a single epoch. Since the size of $\mathbf{P}'_k$ is $N_k \times R$ and the size of $((\odot_{i=1}^{d-2}\mathbf{V}_i) \odot \mathbf{S}(k,:))$ is $(\prod_{i=1}^{d-2} M_i) \times R$, computing $\mathbf{P}'_k((\odot_{i=1}^{d-2}\mathbf{V}_i) \odot \mathbf{S}(k,:))^{\top}$ takes $O(N_k (\prod_{i=1}^{d-2} M_i)R)$ time, and computing it for all k takes $O((\sum_{k=1}^{K} N_k) (\prod_{i=1}^{d-2} M_i)R)$. Hence, an epoch of Light-IT requires $O((\prod_{i=1}^{d-2} M_i) (\sum_{k=1}^{K} N_k)R + (\sum_{k=1}^{K} N_k)DR)$ time. □

[8] The tensor $\mathcal{G}$ is set to a diagonal tensor whose diagonal entries are all 1

Theorem 4 (Compression time of Light-IT for sparse irregular tensors). *The compression time of the spares version of Light-IT is* $O\big((\sum_{k=1}^{K} N_k + \sum_{i=1}^{d-2} M_i)R^2 + \sum_{k=1}^{K} nnz(\mathcal{X}_k)dR + (\sum_{k=1}^{K} N_k)DR\big)$.

Proof. In the proof, we only mention the most dominant parts of computations. First, Computing $\pi_k(i)$ according to Eq. (3) for all k in $\{1, \cdots, K\}$ and i in $[N_k]$ takes $O((\sum_{k=1}^{K} N_k)DR)$ time. let's consider Eq. (27). Computing $(\prod_{i=1}^{d-2} \mathbf{V}_i^\top \mathbf{V}_i)$ consumes $O(\sum_{i=1}^{d-2} M_i R^2)$ time as the size of $\mathbf{V}_i$ is $M_i \times R$. Computing $(\sum_{k=1}^{K} \mathbf{S}(k, r_1)\ \mathbf{S}(k, r_2)\ \mathbf{P}'_k(:, r_1)^\top \mathbf{P}'_k(:, r_2))$ for all r_1, r_2 in $\{1, \ldots, R\}$ and k in $\{1, \ldots, K\}$ consumes $O(\sum_{k=1}^{K} N_k R^2)$ time as the sizes of $\mathbf{P}'_k(:, r_1)$ and $\mathbf{P}'_k(:, r_2)$ are both $1 \times N_k$. Approximation of a single entry by Light-IT requires $O(dR)$ time, and approximating whole non-zero entries requires $\sum_{k=1}^{K} O(nnz(\mathcal{X}_k)dR)$ time. It is equivalent to the time complexity for computing the second term in the final formula of Eq. (7). To sum up, the running time for an epoch of Light-IT when compressing a sparse irregular tensor is $O((\sum_{k=1}^{K} N_k + \sum_{i=1}^{d-2} M_i)R^2 + \sum_{k=1}^{K} nnz(\mathcal{X}_k)dR + (\sum_{k=1}^{K} N_k)DR)$. □

Theorem 5 (Compression time of Light-IT^{++}). *The compression time of Light-IT^{++} is* $O\big(R^{3(d-2)} + KR^4 + R^6 + R^{2d-1}D + d(\sum_{k=1}^{K} N_k) \prod_{i=1}^{d-2} M_i)R^{d-1} + KR^{2d-1}\big)$.

Proof. Here, the complexities for the most dominant parts are listed. In Eq. (35), given $(\otimes_{i=1}^{d-2}\mathbf{V}_i^\top \mathbf{V}_i)$, computing its pseudoinverse requires $O(R^{3(d-2)})$. Computing $\mathbf{P}_k^\top \mathbf{P}_k \otimes \mathbf{S}(k,:)^\top \mathbf{S}(k,:)$ from $\mathbf{P}_k^\top \mathbf{P}_k$ and $\mathbf{S}(k,:)^\top \mathbf{S}(k,:)$ for all k in $\{1, \ldots, K\}$ needs $O(KR^4)$ time. Computing its pseudoinverse takes $O(R^6)$ time. In Eq. (37), earning $\mathbf{G}^{(1)}\Big((\otimes_{i=1}^{d-2}\mathbf{V}_i^\top \mathbf{V}_i) \otimes \big(\sum_{(j,k)\in T_i} \mathbf{S}(k,:)^\top \mathbf{S}(k,:)\big)\Big)\mathbf{G}^{(1)\top}$ from $\mathcal{G}$ and $(\otimes_{i=1}^{d-2}\mathbf{V}_i^\top \mathbf{V}_i) \otimes \sum_{(j,k)\in T_i} \mathbf{S}(k,:)^\top \mathbf{S}(k,:)$ spends $O(R^{2d-1}D)$ time due to their sizes. In Eq. (21), computing $\mathbf{X}_k^{(i)}(\mathbf{P}_k \otimes \mathbf{S}(k,:) \otimes (\otimes_{j\neq i}\mathbf{V}_j))$ from $\mathcal{X}_k$ and $\mathbf{P}_k \otimes \mathbf{S}(k,:) \otimes (\otimes_{j\neq i}\mathbf{V}_j)$ for all k requires $O((\sum_{k=1}^{K} N_k)\ (\prod_{i=1}^{d-2} M_i)\ R^{d-1})$ time due to their sizes. It takes $O(d(\sum_{k=1}^{K} N_k)\ (\prod_{i=1}^{d-2} M_i)\ R^{d-1})$ in total since Eq. (21) is conducted for $d-2$ times. In Eq. (39), computing $(\mathbf{P}_k^\top \mathbf{P}_k \otimes (\otimes_{i=1}^{d-2}\mathbf{V}_i^\top \mathbf{V}_i))$ from $\mathbf{P}_k^\top \mathbf{P}_k$ and $\otimes_{i=1}^{d-2}\mathbf{V}_i^\top \mathbf{V}_i$ for all k requires $O(KR^{2d-1})$ time. In conclusion, total running time for an epoch is $O(R^{3(d-2)} + KR^4 + R^6 + R^{2d-1}D + d(\sum_{k=1}^{K} N_k) \prod_{i=1}^{d-2} M_i)R^{d-1} + KR^{2d-1})$. □

Theorem 6 (Compression time of Light-IT^{++} for spare irregular tensors). *The compression time of the sparse version of Light-IT^{++} is* $O\big(\prod_{i=1}^{d-2} M_i\, R^4 + \sum_{k=1}^{K} nnz(\mathcal{X}_k)\, R^d d + R^{3(d-2)} + (\prod_{i=1}^{d-2} M_i)R^d + KR^4 + R^6 + R^{2d-1}D + \sum_{i=1}^{d-2} M_i K R^d + d(\sum_{k=1}^{K} N_k)R^2 + KR^{2d-1}\big)$.

Proof. Again, we focus on the most dominant parts of the computations. In Eq. (33), computing $(\otimes_{i=1}^{d-2}\mathbf{V}_i)\mathbf{G}^{(2,\cdots,d-1)} \sum_{k=1}^{K}(\mathbf{P}_k^\top \mathbf{P}_k \otimes \mathbf{S}(k,:)^\top \mathbf{S}(k,:))$ from $(\otimes_{i=1}^{d-2}\mathbf{V}_i)$ and $\mathbf{G}^{(2,\cdots,d-1)} \sum_{k=1}^{K}(\mathbf{P}_k^\top \mathbf{P}_k \otimes \mathbf{S}(k,:)^\top \mathbf{S}(k,:))$ requires $O(\prod_{i=1}^{d-2} M_i R^4)$. Computing the second term in Eq. (29) using Eq. (19) requires $O(\sum_{k=1}^{K} nnz(\mathcal{X}_k)\ R^d d)$ as approximating a single entry takes $O(R^d d)$. In Eq. (37), earning $\mathbf{G}^{(1)}\Big((\otimes_{i=1}^{d-2}\mathbf{V}_i^\top \mathbf{V}_i) \otimes \big(\sum_{(j,k)\in T_i} \mathbf{S}(k,:)^\top \mathbf{S}(k,:)\big)\Big)\mathbf{G}^{(1)\top}$ from $\mathcal{G}$ and $(\otimes_{i=1}^{d-2}\mathbf{V}_i^\top \mathbf{V}_i) \otimes \sum_{(j,k)\in T_i} \mathbf{S}(k,:)^\top \mathbf{S}(k,:)$ spends $O(R^{2d-1}D)$ time due to their sizes. In Eq. (21), computing $\mathbf{X}_k^{(i)}(\mathbf{P}_k \otimes (\otimes_{j\neq i}\mathbf{V}_j) \otimes \mathbf{S}(k,:))\mathbf{G}^{(i)\top}$ from $\mathbf{X}_k^{(i)}(\mathbf{P}_k \otimes (\otimes_{j\neq i}\mathbf{V}_j) \otimes \mathbf{S}(k,:))$ and $\mathcal{G}$ for all k and i takes $O(\sum_{i=1}^{d-2} M_i K R^d)$ due to their sizes. In Eq. (37), computing $\mathbf{P}_k^\top \mathbf{P}_k$ for all k requires $O(\sum_{k=1}^{k} N_k R^2)$ due to the size of $\mathbf{P}_k$. Since Eq. (37) is repeated for $d-2$ times, this part requires $O(d \sum_{k=1}^{k} N_k R^2)$ in total. In Eq. (39), computing $(\mathbf{P}_k^\top \mathbf{P}_k \otimes (\otimes_{i=1}^{d-2}\mathbf{V}_i^\top \mathbf{V}_i))$ from $\mathbf{P}_k^\top \mathbf{P}_k$ and $\otimes_{i=1}^{d-2}\mathbf{V}_i^\top \mathbf{V}_i$ for all k requires $O(KR^{2d-1})$ time. To sum up, total running time of Light-IT^{++} for an epoch when it is executed on a sparse irregular tensor is $O(\prod_{i=1}^{d-2} M_i R^4 + \sum_{k=1}^{K} nnz(\mathcal{X}_k)\ R^d d + R^{3(d-2)} + (\prod_{i=1}^{d-2} M_i)R^d + KR^4 + R^6 + R^{2d-1}D + \sum_{i=1}^{d-2} M_i K R^d + d(\sum_{k=1}^{K} N_k)R^2 + KR^{2d-1})$. □

C SUPPLEMENTS FOR EXPERIMENTS

C.1 Metrics for Compressed Sizes

All methods, except COPA, saved their resulting matrices or tensors in a dense format. For COPA, we chose either a sparse format (i.e., COO) or a dense format, selecting the one with minimal space cost for saving each of the outputs. For the sparse format, we further compressed integer indices using Huffman encoding.

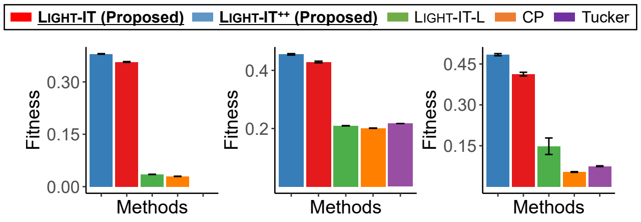
C.2 Training Details

When training Light-IT, we used 500 epochs for all datasets. We measured the fitness of Light-IT every 10 epochs and reported the highest obtained fitness. We tuned the learning rate for Light-IT by varying it in {1, 0.1, 0.01, 0.001}. As a result, the learning rate of Light-IT is set to 10^{-2} except in US-stock. In US-stock, 10^{-1} is used as the learning rate when the ranks are 4 and 8, and 10^{-2} is used when the ranks are 12 and 16. The initial parameters of Light-IT^{++} were set to the parameters of Light-IT, and the core tensor was initialized to a diagonal tensor whose diagonal entries were all 1. We updated the parameters of Light-IT^{++} until the increase of fitness is smaller than 10^{-4}. We used the same convergence condition for PARAFAC2-ALS. When training COPA on sparse tensors, we imposed non-negative constraints to $\mathbf{H}$, $\mathbf{W}$ and a sparsity constraint to $\mathbf{V}$, following one of the scenarios outlined in the original paper. For dense tensors, we gave non-negative constraints to all $\mathbf{H}$, $\mathbf{W}$, and $\mathbf{V}$ since running with sparsity constraints on $\mathbf{V}$ was failed. We set the convergence tolerance to 10^{-4} and the maximum number of iterations to 1000 when executing SPARTan, as provided in the authors' code. The convergence tolerance was set to 10^{-4} when running COPA, and this was the setting provided in the authors' code. We also set the convergence tolerance for PARAFAC2-ALS to 10^{-4}. We set the maximum number of iterations to 32 for DPar2 and RD-ALS, as the authors of DPar2 did. For REPAIR, we set the convergence tolerance to 10^{-3}, which is the setting in the authors' code.

C.3 Semantics of Data and Preprocessing Steps

CMS:[9] Centers for Medicare and Medicaid Services (CMS) data are synthesized data of Medicare beneficiaries and their claims from 2008 to 2010. We built a tensor to count the diagnoses that each patient has received in the form of (visits, diagnoses, patients; counts).

[9] https://www.cms.gov/data-research/statistics-trends-and-reports/medicare-claims-synthetic-public-use-files/cms-2008-2010-data-entrepreneurs-synthetic-public-use-file-de-synpuf



(a) CMS (b) Korea-stock (c) Enron

Figure 10: Our ideas on vocabulary-based compression and extension with core tensor are effective for compression.

Table 3: Sizes of tensors and their number of non-zeros in Fig. 11(a) and 8(a)

Order	Density	Order	Density
3	10^{-2}	4	10^{-3}
	2×10^{-2}		2×10^{-3}
	4×10^{-2}		4×10^{-3}
	8×10^{-2}		8×10^{-3}
	1.6×10^{-1}		1.6×10^{-2}

We categorized the diagnosis codes into Clinical Classification Software (CCS)[10], and assumed that each claim id indicates a unique visit. We only considered the patients who visited clinics more than once.

MIMIC-III [15]:[11] MIMIC-III is an intensive care unit (ICU) dataset collected between 2001 and 2012. The tensor we built saves the medications that each patient has prescribed and is composed of (dates, drugs, patients; counts). We removed the dates that didn't include any medications and removed patients whose lengths of dates were less than 2. We only used 1000 types of drugs which were most frequently prescribed.

Korea-stock, US-stock [14]:[12] Stock datasets are the collections of stocks on the South Korea and US stock markets. Each dataset is represented as a tensor of (dates, feature types, stocks; feature values). We used the tensors preprocessed by the authors of [14].

Enron [26, 27]:[13] Enron stores the counts of words in emails in the form of (dates, senders, words, receivers; counts). We only used senders and receivers whose IDs were smaller than 1001, and we checked the 1000 most frequently used words. We made the irregularity in the tensor by removing the dates in each receiver when no email had arrived.

Delicious [8, 27]:[14] The Delicious dataset is a collection of tags from the Delicious website and consists of (dates, items, tags, users; binary indicators for tags). We sampled the most frequently appeared items and tags when building a tensor. We got rid of dates that don't include any tag for each user, forming irregularity of the tensor.

Table 4: Sizes of tensors in Fig. 11(b)

Order	N_{max}	N_{avg}	Size (except the 1st mode)
3	255	136.52	512×256
	255	136.52	512×512
	510	273.04	512×512
	510	261.71	1024×512

Table 5: Sizes of tensors in Fig. 8(b)

Order	N_{max}	N_{avg}	Size (except the 1st mode)
3	63	31.55	$64 \times 64 \times 64$
	126	63.09	$64 \times 64 \times 64$
	126	63.09	$128 \times 64 \times 64$
	126	63.09	$128 \times 128 \times 64$
	126	63.09	$128 \times 128 \times 128$

C.4 Hyperparameter Settings

The hyperparameters of the methods in Section 6.2 are listed in the section. The ranks of Light-IT are set to 5, 10, 20, 30 in CMS. MIMIC-III, Enron, and Delicious. In the cases of Korea-stock and US-stock, 4, 8, 12, 16 are used for the ranks of Light-IT. For Light-IT^{++}, we set the ranks to 5, 10, 20, 30 in CMS and MIMIC-III, 4, 8, 12, 15 in Korea-stock and US-stock, 5, 9. 12, 15 for Enron, and 5, 11, 17, 23 in Delicious.

For all competitors except REPAIR, we set the ranks to 3, 6, 9, 12, 15 in CMS, 5, 8, 11, 14, 17, 20 in MIMIC-III, 1, 2, 4, 6, 8 in Korea-stock, and 1, 2, 3, 4, 5, 6, 7 in US-stock. We set the ranks of REPAIR to 5, 8, 11, 14, 17, 20 in MIMIC-III. For REPAIR, we set the number of bases to 253, following the one of the settings in the paper.

C.5 Supplements for the Ablation Study

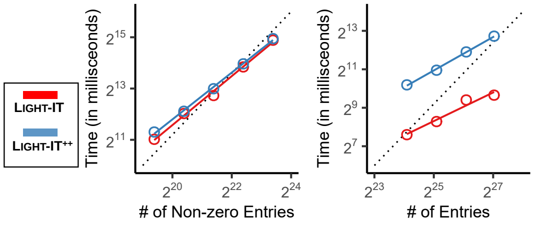
The ablation studies for the remaining datasets are in Fig. 10. Still, our ideas help increase accuracy, and regular tensor decompositions are inaccurate. Note that out-of-memory occurred when running Tucker decomposition on the CMS dataset. For the methods in Fig. 6 of Section 6.3, we set the rank to 30 in CMS and MIMIC-III, 16 in Korea-stock and US-stock, 15 in Enron, and 23 in Delicious. We tuned the learning rates of Light-IT and Light-IT-L from $\{1, 10^{-1}$ $, 10^{-2}, 10^{-3}\}$. The convergence tolerances for the CP and Tucker decompositions are set to 10^{-4}.

When conducting the experiments for Fig. 7 of Section 6.3, we sampled the first few slices from real-world sparse tensors (1000 for CMS and MIMIC-III, 20 for Enron, and 30 for Delicious) since total tensors in dense formats are too large. For all datasets in Fig. 7, we set the number of epochs of Light-IT to 500, learning rate of Light-IT to 10^{-2}, rank to 10, and convergence tolerance of Light-IT^{++} to 10^{-4}.

C.6 Supplments for the Scalability Test

The results for 3-order tensors are in Fig. 11. The (minimum, maximum, average) number of rows of the 3-order sparse irregular tensors in Section 6.4 is (2, 510, 261.71), and the mode lengths of the remaining modes are all 512. The (minimum, maximum, average) number of rows of the 4-order sparse irregular tensors in Section 6.4 is (2, 126, 70.38), and the mode lengths of the remaining modes are all 128. The densities of the sparse irregular tensors are in Table 3.

[10] https://hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp
[11] https://physionet.org/content/mimiciii/1.4/
[12] https://datalab.snu.ac.kr/dpar2/
[13] https://frostt.io/tensors/enron/
[14] https://frostt.io/tensors/delicious/

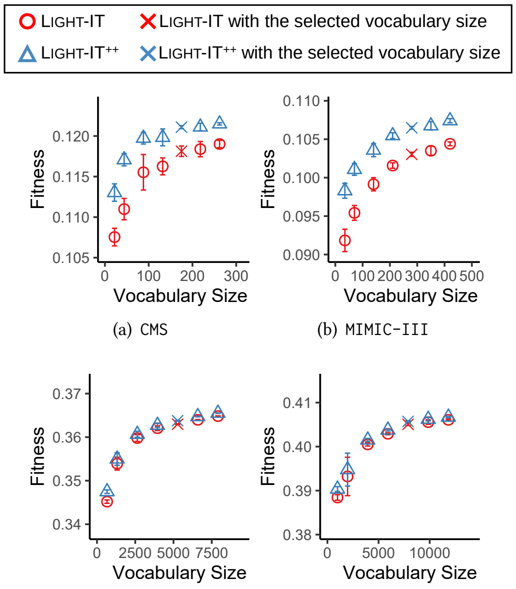


(a) 3-order sparse tensors (b) 3-order dense tensors

Figure 11: Compression time of our methods increases (sub-)linearly to the number of non-zero entries or entries.

Table 6: Ranks and fitness of the methods in Fig. 9

Dataset		Light-IT	Light-IT^{++}	SPARTan	COPA	PARAFAC2-ALS	Dpar2	RD-ALS	REPAIR
CMS	Rank	10	10	9	12	9	12	12	O.O.M
	Fitness	0.202	0.209	0.194	0.205	0.194	0.192	0.197	O.O.M
MIMIC-III	Rank	5	5	5	8	5	5	5	11
	Fitness	0.103	0.107	0.105	0.113	0.106	0.949	0.967	0.108
Korea-stock	Rank	4	4	4	4	4	4	4	O.O.M
	Fitness	0.362	0.363	0.372	0.351	0.372	0.358	0.358	O.O.M
US-stock	Rank	4	4	4	5	4	4	4	O.O.M
	Fitness	0.405	0.405	0.412	0.403	0.412	0.405	0.405	O.O.M
Enron	Rank	5	5	only applicable to 3-order irregular tensors					
	Fitness	0.263	0.279						
Delicious	Rank	5	5						
	Fitness	0.005	0.006						



(a) CMS (b) MIMIC-III

(c) Korea-stock (d) US-stock

Figure 12: Setting the vocabulary size to the maximum mode length of the first mode is sufficient.

The sizes of the dense irregular tensors used in Section 6.4 are in Table 4 and 5

C.7 Supplements for the Time Experiment

The ranks and fitness of all methods used in Section 6.5 are in Table 6.

C.8 Vocabulary Size

We investigate how the accuracies of our models change depending on the size of the vocabulary, and the results are depicted in Fig. 12. The ranks of the modes are fixed to 5 in the CMS and MIMIC-III datasets, and 4 in the Korea-stock and US-stock datasets. The accuracies of the models are nearly saturated when the vocabulary sizes are near the maximum mode length of the first mode. Thus, the settings in Section 6.2 are empirically validated.