IMPLICIT MULTIPLE TENSOR DECOMPOSITION *

KUNJING YANG†, LIBIN ZHENG‡, AND MINRU BAI§

Abstract. Recently, triple decomposition has attracted increasing attention for decomposing third-order tensors into three factor tensors. However, this approach is limited to third-order tensors and enforces uniformity in the lower dimensions across all factor tensors, which restricts its flexibility and applicability. To address these issues, we propose the Multiple decomposition, a novel framework that generalizes triple decomposition to arbitrary order tensors and allows the short dimensions of the factor tensors to differ. We establish its connections with other classical tensor decompositions. Furthermore, implicit neural representation (INR) is employed to continuously represent the factor tensors in Multiple decomposition, enabling the method to generalize to non-grid data. We refer to this INR-based Multiple decomposition as Implicit Multiple Tensor Decomposition (IMTD). Then, the Proximal Alternating Least Squares (PALS) algorithm is utilized to solve the IMTD-based tensor reconstruction models. Since the objective function in IMTD-based models often lacks the Kurdyka–Łojasiewicz (KL) property, we establish a KL-free convergence analysis for the algorithm. Finally, extensive numerical experiments further validate the effectiveness of the proposed method.

Key words. Triple decomposition, Multiple decomposition, arbitrary order tensors, implicit neural representation, non-grid data, convergence analysis.

MSC codes. 15A23, 15A69, 68U10

1. Introduction.

With the rapid development of big data processing in recent years, tensors have received significant attention and have become one of the most powerful tools for representing multi-dimensional data. Currently, tensors play a pivotal role in various fields, including remote sensing image processing [7, 6], computer vision [19, 37], deep learning [29], and signal processing [18].

Numerous real-world tensor datasets inherently exhibit low-rank structures, reflecting an underlying low-dimensional nature [21]. This property has been widely observed in regular grid-based data such as images [40], videos [5], and wireless channel measurements [8]. Recently, low-rank characteristics have also been identified in tensor representations of irregular, non-grid data, such as point cloud [44, 21]. Presently, various tensor decompositions and corresponding tensor ranks have been proposed to explore the underlying low-rank structure of data. Among the most classical are the CANDECOMP/PARAFAC (CP) decomposition and Tucker decomposition. The CP decomposition represents a tensor as a sum of rank-one tensor components, and the minimum number of these rank-one tensors is called the CP rank [15]. Under mild conditions, the CP decomposition is unique, a property that has enabled its widespread use in diverse fields, such as chemical component analysis [33] and spectral unmixing [11]. However, computing the exact CP rank is NP-hard [9], making its accurate determination computationally intractable in general. Tucker decomposition factorizes a tensor into a core tensor and a collection of factor matrices associated with each mode. The Tucker rank is the tuple of the matrix ranks of the tensor's mode-n unfoldings [15]. In fact, CP decomposition can be regarded as a special case of Tucker decomposition in which the core tensor is super-diagonal. Currently, Tucker

*Submitted to the editors DATE.

†The School of Mathematics, Hunan University, Changsha, Hunan 410082, P. R. China (kunjingyang@hnu.edu.cn).

‡The School of Mathematics, Hunan University, Changsha, Hunan 410082, P. R. China (fifholz301@hnu.edu.cn).

§ The Corresponding author with School of Mathematics, Hunan University, Changsha, Hunan 410082, P. R. China (minru-bai@hnu.edu.cn).

decomposition is widely used in applications such as data compression [42] and image restoration [17]. However, the core tensor usually contains a large number of parameters, resulting in high storage and computational complexity [25].

Another family is the tensor network decompositions, including Tensor Train (TT) [23], Tensor Ring (TR) [38] and Fully Connected Tensor Network (FCTN) [43] decompositions. These approaches have been successfully applied to image reconstruction [12] and data compression [1]. However, both TT and TR decompositions only model interactions between adjacent factor tensors, failing to capture long-range dependencies. Although FCTN decomposition allows full connectivity among all factors, it comes at the cost of substantially higher storage and computational complexity. In recent years, tensor singular value decomposition (t-SVD) [13] has attracted increasing attention. Within this framework, the tensor nuclear norm (TNN), as a convex relaxation of tensor tubal rank, has been widely applied in fields such as tensor completion [40] and robust principal component analysis [20]. To enhance flexibility, the transform-based tensor nuclear norm (TTNN) has also been proposed for low-rank approximation [26, 16]. Wang *et al.* [32] extended the t-SVD framework to functional settings for missing slice tensor recovery tasks. Despite its popularity, however, t-SVD is limited to the decomposition of third-order tensors [10].

More recently, Qi *et al.* [25] developed the triple decomposition, which decomposes a third-order tensor into three factor tensors. Each factor tensor has one dominant dimension, which is considered to inherit the features along the corresponding modes. Currently, triple decomposition has found widespread applications in areas such as traffic network recovery [22] and image fusion [35]. Nonetheless, in contrast to classical tensor decompositions such as CP and Tucker, triple decomposition is specifically designed for third-order tensors, which limits its applicability to higher-order data. Moreover, triple decomposition requires the short dimensions of the factor tensors to be equal, which limits its flexibility and interpretability.

To address these issues, we propose the Multiple decomposition, a method which decomposes an N th-order tensor ($N \geq 3$) into N factor tensors. Each factor tensor also has only one dominant dimension and is designed to inherit features from the corresponding mode of the original tensor. Unlike triple decomposition, multiple decomposition allows the sizes of the non-dominant dimensions in the factor tensors to be flexible and independently adjustable. Theoretically, we prove that this variability in short dimensions enables a higher compression ratio, improving storage and computational efficiency. We also establish the connections between Multiple decomposition and other classical tensor decompositions. In addition, it is worth noting that most existing tensor decomposition methods are designed for grid-structured tensor data, making them ill-suited for non-grid data formats. To further extend the applicability of Multiple decomposition, we incorporate implicit neural representation (INR) [30] to represent the factor tensors as continuous functions. Specifically, each factor tensor is modeled as an implicit function, parameterized by a neural network that maps a coordinate to the associated value in the tensor. We refer to this INR-based Multiple decomposition as Implicit Multiple Tensor Decomposition (IMTD), which enables the modeling of low-rank structures in irregular and non-grid data. Then, the Proximal Alternating Least Squares (PALS) algorithm is employed to solve the tensor reconstruction problem within the IMTD framework. Under mild conditions, we show that the implicit functions used to represent the target tensors in IMTD are Lipschitz continuous. Then, we provide a convergence analysis for the PALS algorithm that does not rely on the Kurdyka–Łojasiewicz (KL) property [3, 2]. This is important since the KL property can be difficult to verify for deep neural networks.

Finally, extensive numerical experiments are conducted to validate the effectiveness and broad applicability of the proposed method.

The main contributions of this paper are summarized as follows:

- We propose the novel Multiple decomposition, which decomposes a N th-order tensor ($N \geq 3$) into N factor tensors. In fact, multiple decomposition can be viewed as a generalization of triple decomposition, extending its applicability from third-order tensors to arbitrary order tensors. Moreover, Multiple decomposition allows the short dimensions of factor tensors to differ, enhancing flexibility and adaptability of the model.
- By integrating Multiple decomposition with implicit neural representation (INR), we develop the Implicit Multiple Tensor Decomposition (IMTD), a framework capable of representing the factor tensors as continuous implicit functions and performing decomposition on non-grid structured data.
- Theoretically, we establish the connections between the Multiple decomposition and other classical tensor decompositions. Under mild conditions, we prove that the IMTD-induced tensor function is Lipschitz continuous. Moreover, we provide a KL-free convergence analysis for the PALS algorithm. Finally, extensive numerical experiments are conducted to demonstrate the effectiveness of the proposed method.

The remaining sections of this paper are organized as follows. Section 2 outlines the essential notation and definitions. The proposed IMTD is introduced in Section 3. In Section 4, we present corresponding model, algorithm and convergence analysis. Finally, we provide extensive experiments to verify the effectiveness of IMTD in Section 5 and make a conclusion in Section 6.

2. Preliminary. In this paper, we adopt the following notation: scalars are denoted by lowercase letters, such as 'a'; vectors are denoted by bold lowercase letters such as '$\mathbf{a}$'; matrices are represented by bold uppercase letters such as '$\mathbf{A}$'; tensors are denoted by calligraphic letters such as '$\mathcal{A}$'; sets and fields are denoted by hollow letters such as '$\Re$'. The 'max', 'submax', and 'mid' denote the maximum value, the second largest element, and the median operation, respectively.

For N-order tensor $\mathcal{X} \in \Re^{I_1 \times I_2 \times \ldots \times I_N}$, we denote $\mathcal{X}_{(n)} \in \Re^{I_n \times (I_1 \ldots I_{n-1} I_{n+1} \ldots I_N)}$ as the mode-n unfolding matrix of $\mathcal{X}$. The Frobenius norm (F norm) of $\mathcal{X}$ is defined as $\|\mathcal{X}\|_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} |\mathcal{X}_{i_1 i_2 \ldots i_N}|^2}$, and its ℓ_1 norm is defined as $\|\mathcal{X}\|_{\ell_1} = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} |\mathcal{X}_{i_1 i_2 \ldots i_N}|$. In this paper, F norm is used if there is no special explanation. The mode-n product of a tensor $\mathcal{A} \in \Re^{I_1 \times I_2, \ldots, \times I_N}$ with a matrix $\mathbf{U} \in \Re^{J \times I_n}$ is denoted by $\mathcal{A} \times_n \mathbf{U}$ and is of size $I_1 \times, \ldots, \times I_{n-1} \times J \times I_{n+1} \times \ldots, I_N$. Elementwise, we have $(\mathcal{A} \times_n \mathbf{U})_{i_1, \ldots, i_{n-1} j i_{n+1}, \ldots, i_N} = \sum_{i_n}^{I_n} \mathbf{a}_{i_1 i_2 \ldots i_N} u_{j i_n}$. The mode-$n$ product can also be represented as matrix multiplication: $\mathcal{Y} = \mathcal{A} \times_n \mathbf{U} \iff \mathbf{Y}_{(n)} = \mathbf{U} \mathcal{A}_{(n)}$. The result of a series of multiplications in different modes is independent of the order of multiplication, i.e., $\mathcal{A} \times_n \mathbf{U} \times_m \mathbf{V} = \mathcal{A} \times_m \mathbf{V} \times_n \mathbf{U} \quad (m \neq n)$. If the modes are the same, then $\mathcal{A} \times_n \mathbf{U} \times_n \mathbf{V} = \mathcal{A} \times_n (\mathbf{V}\mathbf{U})$. Next, we introduce some classical tensor decomposition methods.

Definition 2.1. **(CP decomposition [15])** Suppose that $\mathcal{X} \in \Re^{I_1 \times I_2 \times \ldots \times I_N}$. Let $\mathbf{A}_n \in \Re^{I_n \times r}$, $n = 1, 2, \ldots, N$ be the factor matrices. If

$$\mathcal{X}_{i_1 i_2 \ldots i_N} = \sum_{p=1}^{r} (\mathbf{A}_1)_{i_1 p} (\mathbf{A}_2)_{i_2 p} \ldots (\mathbf{A}_N)_{i_N p} \tag{2.1}$$

for $i_n = 1, \ldots, I_n$ and $n = 1, \ldots, N$, then we say that $\mathcal{X}$ has a CP decomposition

$\mathcal{X} = [[\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_N]]$. The smallest integer r such that (2.1) holds is called the CP rank of $\mathcal{X}$, and is denoted as $\text{CPRank}(\mathcal{X}) = r$.

Definition 2.2. **(Tucker rank [15])** Suppose that the tensor $\mathcal{X} \in \Re^{I_1 \times I_2 \times ... \times I_N}$. We may unfold $\mathcal{X}$ to a matrix $\mathcal{X}_{(n)} \in \Re^{I_n \times I_1 ... I_{n-1} I_{n+1} ... I_N}$. Denote the matrix ranks of $\mathcal{X}_{(n)}$ as r_n, $n = 1, 2, ..., N$, respectively. Then the vector $(r_1, r_2, ..., r_N)$ is called the Tucker rank of $\mathcal{X}$, and is denoted as $\text{TuckRank}(\mathcal{X}) = (r_1, r_2, ..., r_N)$.

Definition 2.3. **(Tucker decomposition [15])** Suppose that the tensor $\mathcal{X} \in \Re^{I_1 \times I_2 \times ... \times I_N}$. Let $\mathbf{U}_n \in \Re^{I_n \times r_n}$, $n = 1, 2, ..., N$, and $\mathcal{C} \in \Re^{r_1 \times r_2 \times ... \times r_N}$. If

$$\mathcal{X}_{i_1 i_2 ... i_N} = \sum_{p_1=1}^{r_1} \sum_{p_2=1}^{r_2} ... \sum_{p_N=1}^{r_N} (\mathbf{U}_1)_{i_1 p_1} (\mathbf{U}_2)_{i_2 p_2} ... (\mathbf{U}_N)_{i_N p_N} \mathcal{C}_{p_1 p_2 ... p_N} \tag{2.2}$$

for $i_n = 1, \ldots, I_n$ and $n = 1, \ldots, N$, then we denote $\mathcal{X}$ has a Tucker decomposition $\mathcal{X} = [[\mathcal{C};\ \mathbf{U}_1, \mathbf{U}_2, ..., \mathbf{U}_N]]$. The matrices $\mathbf{U}_n$, $n = 1, 2, ..., N$ are called factor matrices of the Tucker decomposition, and $\mathcal{C}$ is called the Tucker core tensor. We may also denote the Tucker decomposition as $\mathcal{X} = \mathcal{D} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times ... \times_N \mathbf{U}_N$. The Tucker ranks r_n, $n = 1, 2, .., N$ are the smallest integers such that (2.2) holds.

Definition 2.4. **(Triple decomposition [25])** Let $\mathcal{X} = (x_{ijt}) \in \Re^{n_1 \times n_2 \times n_3}$ be a nonzero tensor, then $\mathcal{X}$ is said the triple product of a third-order horizontally square tensor $\mathcal{A} = (a_{ijt}) \in \Re^{n_1 \times r \times r}$, a third-order laterally square tensor $\mathcal{B} = (b_{pjs}) \in \Re^{r \times n_2 \times r}$, and a third-order frontally square tensor $\mathcal{C} = (b_{pqt}) \in \Re^{r \times r \times n_3}$ if

$$\mathcal{X}_{ijt} = \sum_{p=1}^{r} \sum_{q=1}^{r} \sum_{s=1}^{r} a_{iqs} b_{pjs} c_{pqt}, \tag{2.3}$$

for $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$ and $t = 1, \ldots, n_3$. For convenience, $\mathcal{X}$ is denoted as

$$\mathcal{X} = [\mathcal{A}\mathcal{B}\mathcal{C}], \tag{2.4}$$

and $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ are designated as factor tensors of $\mathcal{X}$. The smallest value of r such that (2.3) holds is called the triple rank of $\mathcal{X}$, and is denoted as $\text{TriRank}(\mathcal{X})$.

3. Implicit Multiple Tensor Decomposition. In this section, we first present the definition of Multiple decomposition and explore its relationship with other classical tensor decompositions. Then, we introduce the Implicit Multiple Tensor Decomposition (IMTD), along with its associated properties.

3.1. Multiple decomposition. Before formally introducing the proposed Multiple decomposition, we introduce the concept of generalized triple decomposition, which is a generalization of triple decomposition to arbitrary-order tensors.

Definition 3.1. **(Generalized triple decomposition)** Let $\mathcal{X} \in \Re^{I_1 \times I_2 \times ... \times I_N}$ be a nonzero tensor, then $\mathcal{X}$ is said the generalized triple product of N-order tensors $\mathcal{A}_1 \in \Re^{I_1 \times r \times ... \times r}$, $\mathcal{A}_2 \in \Re^{r \times I_2 \times r \times ... \times r}$, ..., and $\mathcal{A}_N \in \Re^{r \times r \times ... \times r \times I_N}$ if

$$\mathcal{X}_{i_1 i_2 ... i_N} = \sum_{p_1=1}^{r} \sum_{p_2=1}^{r} ... \sum_{p_N=1}^{r} (\mathcal{A}_1)_{i_1 p_2 ... p_N} (\mathcal{A}_2)_{p_1 i_2 p_3 ... p_N} ... (\mathcal{A}_N)_{p_1 p_2 ... p_{N-1} i_N}, \tag{3.1}$$

for $i_n = 1, \ldots, I_n$ and $n = 1, \ldots, N$. For convenience, $\mathcal{X}$ is denoted as

$$\mathcal{X} = [\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_N], \tag{3.2}$$

and $\mathcal{A}_1$, $\mathcal{A}_2$,..., $\mathcal{A}_N$ are designated as factor tensors of $\mathcal{X}$. The smallest r such that (3.1) holds is called the generalized triple rank of $\mathcal{X}$, and is denoted as $\text{GTriRank}(\mathcal{X})$. For a zero tensor, its generalized triple rank is defined as zero.

Next, the well-definedness of Definition 3.1 and an upper bound on the generalized triple rank are established in the following theorem.

THEOREM 3.1. *Generalized triple decomposition and generalized triple rank are well defined. A nonzero tensor* $\mathcal{X} \in \Re^{I_1 \times I_2 \times \ldots \times I_N}$ *always has a generalized triple decomposition (3.1), satisfying* $\text{GTriRank}(\mathcal{X}) \leq \text{submax}\{I_1, I_2, ..., I_N\}$.

Proof. Without loss of generality, we may assume that we have a nonzero tensor $\mathcal{X} \in \Re^{I_1 \times I_2 \times \ldots \times I_N}$ and $I_1 \geq I_2 \geq ... \geq I_N \geq 1$. Thus, $\text{submax}\{I_1, I_2, ..., I_N\} = I_2$. Let $r = I_2$. Define tensors $\mathcal{A}_1 \in \Re^{I_1 \times r \times \ldots \times r}$, $\mathcal{A}_2 \in \Re^{r \times I_2 \times \ldots \times r}$,..., and $\mathcal{A}_N \in \Re^{r \times r \times \ldots \times I_N}$, such that $(\mathcal{A}_1)_{i_1 p_2 \ldots p_N} = x_{i_1 p_N p_{N-1} \ldots p_3 p_2}$ for $p_3 \leq I_3, p_4 \leq I_4, ..., p_N \leq I_N$ and $(\mathcal{A}_1)_{i_1 p_2 \ldots p_N} = 0$ otherwise, $(\mathcal{A}_2)_{1 i_2 \ldots p_N} = \delta_{i_2 p_3 \ldots p_N}$ and $(\mathcal{A}_2)_{p_1 i_2 \ldots p_N} = 0$ for $p_1 > 1$, ..., $(\mathcal{A}_N)_{1 p_2 \ldots i_N} = \delta_{p_2 \ldots p_{N-1} i_N}$ and $(\mathcal{A}_N)_{p_1 p_2 \ldots i_N} = 0$ for $p_1 > 1$, where $\delta_{p_2 \ldots p_{N-1} i_N}$,..., and $\delta_{p_2 \ldots p_{N-1} i_N}$ are the Kronecker symbol such that $\delta = 1$ when all subscripts are equal, and 0 otherwise. Then, we get

$$\begin{aligned} &\sum_{p_1=1}^{r} \sum_{p_2=1}^{r} \ldots \sum_{p_N=1}^{r} (\mathcal{A}_1)_{i_1 p_2 \ldots p_N} (\mathcal{A}_2)_{p_1 i_2 \ldots p_N} \ldots (\mathcal{A}_N)_{p_1 p_2 \ldots i_N} \\ &= \sum_{p_2=1}^{r} \ldots \sum_{p_N=1}^{r} x_{i_1 p_N \ldots p_2} \delta_{i_2 p_3 \ldots p_N} \ldots \delta_{p_2 \ldots p_{N-1} i_N} = \mathcal{X}_{i_1 i_2 \ldots i_N}, \end{aligned}$$

for $i_n = 1, \ldots, I_n$ and $n = 1, \ldots, N$, i.e., (3.1) holds for the above choices of factor tensors $\mathcal{A}_1$, $\mathcal{A}_2$,..., and $\mathcal{A}_N$. Thus, generalized triple decomposition always exists with $\text{GTriRank}(\mathcal{X}) \leq r := I_2 = \text{submax}\{I_1, I_2, ..., I_N\}$. ∎

Although the generalized triple decomposition is well-defined, it requires the short dimensions of the factor tensors to be identical, which limits its flexibility and applicability. Therefore, we proceed to introduce the proposed Multiple decomposition.

Definition 3.2. **(Multiple decomposition)** Let tensor $\mathcal{X} \in \Re^{I_1 \times I_2 \times \ldots \times I_N}$ be a nonzero tensor, then $\mathcal{X}$ is said the Multiple product of N-order tensors $\mathcal{A}_1 \in \Re^{I_1 \times r_2 \times \ldots \times r_N}$, $\mathcal{A}_2 \in \Re^{r_1 \times I_2 \times r_3 \times \ldots \times r_N}$, ..., and $\mathcal{A}_N \in \Re^{r_1 \times r_2 \times \ldots \times r_{N-1} \times I_N}$ if

$$\mathcal{X}_{i_1 i_2 \ldots i_N} = \sum_{p_1=1}^{r_1} \sum_{p_2=1}^{r_2} \ldots \sum_{p_N=1}^{r_N} (\mathcal{A}_1)_{i_1 p_2 \ldots p_N} (\mathcal{A}_2)_{p_1 i_2 p_3 \ldots p_N} \ldots (\mathcal{A}_N)_{p_1 p_2 \ldots i_{N-1} i_N}, \tag{3.3}$$

for $i_n = 1, \ldots, I_n$ and $n = 1, \ldots, N$. For convenience, we still use (3.2) to represent $\mathcal{X}$, i.e., $\mathcal{X} = [\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_N]$, and $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_N$ are designated as factor tensors of $\mathcal{X}$. If the vector $\mathbf{r} := (r_1, r_2, \ldots, r_N)$ satisfies the following conditions:

1. $r_i \leq \text{GTriRank}(\mathcal{X})$ for all $i = 1, 2, \ldots, N$;
2. $\mathbf{r}$ has the smallest ℓ_1 norm among all such vectors satisfying (3.3). In the case of Multiple vectors sharing the same minimal ℓ_1 norm, $\mathbf{r}$ is selected to be the one with the smallest lexicographical order;

then we refer to $\mathbf{r}$ as the *Multiple rank* of $\mathcal{X}$, and denote it by $\text{MtpRank}(\mathcal{X})$.

The Multiple rank of tensor $\mathcal{X}$ always exists, since we can at least choose $r_i = \text{GTriRank}(\mathcal{X})$ for $i = 1, 2, ..., N$. Note that if the vector $(r_1, r_2, \ldots, r_N)$ satisfies (3.3), then any vector $(s_1, s_2, \ldots, s_N)$ with $s_i \geq r_i$ can also satisfy equation (3.3), simply by zero-padding the corresponding factor tensors. Therefore, condition 2 is imposed to ensure the uniqueness of the Multiple rank. The following theorem establishes the relationship between $\text{GTriRank}(\mathcal{X})$ and $\text{MtpRank}(\mathcal{X})$.

THEOREM 3.2. *If* $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *is a nonzero tensor with* $\text{MtpRank}(\mathcal{X}) = (r_1, r_2, \ldots, r_N)$*, then* $\text{GTriRank}(\mathcal{X}) = \max\{r_1, r_2, \ldots, r_N\}$.

Proof. Due to $\text{MtpRank}(\mathcal{X}) = (r_1, r_2, ..., r_N)$, there exists $\mathcal{A}_1 \in \Re^{I_1 \times r_2 \times \ldots \times r_N}$, $\mathcal{A}_2 \in \Re^{r_1 \times I_2 \times \ldots \times r_N}$, ..., and $\mathcal{A}_N \in \Re^{r_1 \times r_2 \times \ldots \times r_{N-1} \times I_N}$, satisfying $\mathcal{X} = [\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_N]$. Let $r := \max\{r_1, r_2, ..., r_N\}$. We expand $\mathcal{A}_1$, $\mathcal{A}_2$, ..., and $\mathcal{A}_N$ to $\widehat{\mathcal{A}}_1 \in \mathbb{R}^{I_1 \times r \times \cdots \times r}$, $\widehat{\mathcal{A}}_2 \in \mathbb{R}^{r \times I_2 \times \cdots \times r}$, ..., and $\widehat{\mathcal{A}}_N \in \mathbb{R}^{r \times r \times \cdots \times r \times I_N}$ by zero-padding. Then, we have

$$\begin{aligned}
&\sum_{p_1=1}^{r} \sum_{p_2=1}^{r} \ldots \sum_{p_N=1}^{r} (\widehat{\mathcal{A}}_1)_{i_1 p_2 \ldots p_N} (\widehat{\mathcal{A}}_2)_{p_1 i_2 \ldots p_N} \ldots (\widehat{\mathcal{A}}_N)_{p_1 p_2 \ldots i_N} \\
&= \sum_{p_1=1}^{r_1} \sum_{p_2=1}^{r_2} \ldots \sum_{p_N=1}^{r_N} (\mathcal{A}_1)_{i_1 p_2 \ldots p_N} (\mathcal{A}_2)_{p_1 i_2 \ldots p_N} \ldots (\mathcal{A}_N)_{p_1 p_2 \ldots i_N} = \mathcal{X}_{i_1 i_2 \ldots i_N}.
\end{aligned}$$

Therefore, we also have $\mathcal{X} = [\widehat{\mathcal{A}}_1 \widehat{\mathcal{A}}_2 ... \widehat{\mathcal{A}}_N]$, which implies $\text{GTriRank}(\mathcal{X}) \leq r$. According to the definition of Multiple rank, we obtain $\text{GTriRank}(\mathcal{X}) \geq r$, which means $\text{GTriRank}(\mathcal{X}) = \max\{r_1, r_2, ..., r_N\}$. ∎

Remark 3.1. Theorem 3.2 implies that the requirement of equal auxiliary dimensions in the factor tensors may introduce redundancy, as they are uniformly set to $\max\{r_1, \ldots, r_N\}$. Therefore, Multiple decomposition, by allowing flexible rank configurations, can achieve a higher compression ratio and improved efficiency.

Next, we introduce some operational properties of the Multiple product.

THEOREM 3.3. *Suppose that tensor* $\mathcal{A}_1 \in \Re^{I_1 \times r_2 \times \ldots \times r_N}$, $\mathcal{A}_2 \in \Re^{r_1 \times I_2 \times \ldots \times r_N}$, ..., $\mathcal{A}_N \in \Re^{r_1 \times r_2 \times \ldots \times r_{N-1} \times I_N}$ *and* $\widehat{\mathcal{A}}_1 \in \Re^{I_1 \times r_2 \times \ldots \times r_N}$, $\widehat{\mathcal{A}}_2 \in \Re^{r_1 \times I_2 \times \ldots \times r_N}$, ..., $\widehat{\mathcal{A}}_N \in \Re^{r_1 \times r_2 \times \ldots \times I_N}$ *are N-order tensors, then the following properties hold:*

1. $[(\mathcal{A}_1 - \widehat{\mathcal{A}}_1)\mathcal{A}_2 ... \mathcal{A}_N] = [\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_N] - [\widehat{\mathcal{A}}_1 \mathcal{A}_2 ... \mathcal{A}_N]$,
   $[\mathcal{A}_1(\mathcal{A}_2 - \widehat{\mathcal{A}}_2)... \mathcal{A}_N] = [\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_N] - [\mathcal{A}_1 \widehat{\mathcal{A}}_2 ... \mathcal{A}_N]$,
   ...
   $[\mathcal{A}_1 \mathcal{A}_2 ...(\mathcal{A}_N - \widehat{\mathcal{A}}_N)] = [\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_N] - [\mathcal{A}_1 \mathcal{A}_2 ... \widehat{\mathcal{A}}_N]$.
2. *If* $\mathcal{X} = [\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_N]$*, then we have*

$$|\mathcal{X}_{i_1 i_2 \ldots i_N}| \leq \|\mathcal{A}_1(i_1, :, ..., :)\|_{\ell_1} \|\mathcal{A}_2(:, i_2, ..., :)\|_{\ell_1} ... \|\mathcal{A}_N(:, :, ..., i_N)\|_{\ell_1}.$$

3. *If* $\mathcal{X} = [\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_N]$*, then for* $n = 1, 2, ..., N$*, one has*

$$\mathcal{X}_{(n)} = (\mathcal{A}_n)_{(n)} \mathcal{E}_{n[N-1]}^{\top},$$

where $\mathcal{E}_n$ *is a* $(2N-2)$*-order tensor with size* $I_1 \times ... \times I_{n-1} \times I_{n+1} \times ... \times I_N \times r_1 \times \ldots \times r_{n-1} \times r_{n+1} \times ... \times r_N$*, and* $(\mathcal{E}_n)_{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N p_1 \ldots p_{n-1} p_{n+1} \ldots p_N} = \sum_{p_n=1}^{r_n} (\mathcal{A}_1)_{i_1 p_2 \ldots p_N} ... (\mathcal{A}_{n-1})_{p_1 \ldots i_{n-1} p_n \ldots p_N} \ (\mathcal{A}_{n+1})_{p_1 \ldots p_n i_{n+1} \ldots p_N} ... (\mathcal{A}_N)_{p_1 p_2 \ldots i_N}$.
The $\mathcal{E}_{n[N-1]}$ *is a matrix by collapsing the first* $N-1$ *dimensions of* $\mathcal{E}$ *and reshaping the remaining dimensions into a single dimension.*

Proof. Properties 1 and 2 follow directly from the definition of multiple product. We focus on proving Property 3. To clarify the procedure, we present the mode-1 unfolding of the tensor as an illustrative example, noting that the unfoldings along

other dimensions follow analogously:

$$\begin{aligned}\mathcal{X}_{i_1 i_2 \ldots i_N} &= \sum_{p_1=1}^{r_1}\sum_{p_2=1}^{r_2}\ldots\sum_{p_N=1}^{r_N}(\mathcal{A}_1)_{i_1 p_2\ldots p_N}(\mathcal{A}_2)_{p_1 i_2\ldots p_N}\ldots(\mathcal{A}_N)_{p_1 p_2\ldots i_N}\\ &= \sum_{p_2=1}^{r_2}\ldots\sum_{p_N=1}^{r_N}(\mathcal{A}_1)_{i_1 p_2\ldots p_N}(\mathcal{A}_2)_{1 i_2\ldots p_N}\ldots(\mathcal{A}_N)_{1 p_2\ldots i_N}\\ &+ \sum_{p_2=1}^{r_2}\ldots\sum_{p_N=1}^{r_N}(\mathcal{A}_1)_{i_1 p_2\ldots p_N}(\mathcal{A}_2)_{2 i_2\ldots p_N}\ldots(\mathcal{A}_N)_{2 p_2\ldots i_N}\\ &\ldots\\ &+ \sum_{p_2=1}^{r_2}\ldots\sum_{p_N=1}^{r_N}(\mathcal{A}_1)_{i_1 p_2\ldots p_N}(\mathcal{A}_2)_{r_1 i_2\ldots p_N}\ldots(\mathcal{A}_N)_{r_1 p_2\ldots i_N}.\end{aligned} \tag{3.4}$$

Notice that each term shares the common factor $(\mathcal{A}_1)_{i_1 p_2\ldots p_N}$. Therefore, one has

$$\mathcal{X}_{i_1 i_2\ldots i_N} = \sum_{p_2=1}^{r_2}\ldots\sum_{p_N=1}^{r_N}[(\mathcal{A}_1)_{i_1 p_2\ldots p_N}\mathbf{e}_{i_2\ldots i_N p_2\ldots p_N}], \tag{3.5}$$

where $\mathbf{e}_{i_2\ldots i_N p_2\ldots p_N} := \sum_{p_1=1}^{r_1}(\mathcal{A}_2)_{p_1 i_2 p_3\ldots p_N}\ldots(\mathcal{A}_N)_{p_1 p_2\ldots p_{N-1} i_N}$. We can rewrite (3.5) as:

$$\mathcal{X}_{i_1 i_2\ldots i_N} = \sum_{k=1}^{K}[(\mathcal{A}_1)_{i_1 k}\mathbf{e}_{i_2\ldots i_N k}], \quad \text{with} \quad K = r_2 r_3\ldots r_N. \tag{3.6}$$

The right-hand side of equation (3.6) can be viewed as a matrix multiplication:

$$\mathcal{X}_{i_1 i_2\ldots i_N} = [(\mathcal{A}_1)_{(1)}\mathcal{E}_{[N-1]}^{\top}]_{i_1(i_2\ldots i_N)}. \tag{3.7}$$

Since $\mathcal{X}_{i_1 i_2\ldots i_N}$ corresponds to $(\mathcal{X}_{(1)})_{i_1(i_2\ldots i_N)}$, we can obtain

$$\mathcal{X}_{(1)} = (\mathcal{A}_1)_{(1)}\mathcal{E}_{[N-1]}^{\top}.$$

The computation of other mode unfoldings can be derived similarly to that of the first mode, which completes the proof. ∎

Remark 3.2. Property 1 establishes that the Multiple product satisfies the distributive law, while Property 2 provides a useful upper bound for this operation. Both follow directly from the definition of Multiple product. Property 3 presents a computational formula for unfolding the Multiple product of factor tensors across different modes, which can be used to solve the Multiple decomposition by iteratively updating the factor tensors within the block coordinate descent (BCD) framework.

3.1.1. The relation between the Multiple and CP decompositions. In fact, Multiple decomposition can be viewed as an extension of CP decomposition, where each factor matrix in CP decomposition is generalized to a factor tensor in Multiple decomposition, as illustrated in Figure 1. The formal relationship can be established in the following theorem.

THEOREM 3.4. *For $\mathcal{X} \in \Re^{I_1\times I_2\times\ldots\times I_N}$ with MtpRank($\mathcal{X}$)=$(r_1, r_2, ..., r_N)$, we have*

$$\max\{r_1, r_2, ..., r_N\} \leq \text{CPRank}(\mathcal{X}) \leq r_1 r_2\ldots r_N. \tag{3.8}$$

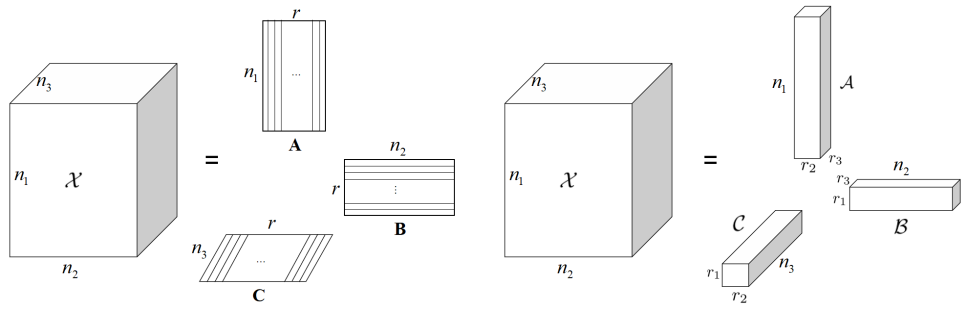


Fig. 1: An intuitive comparison between CP decomposition (left) and Multiple decomposition (right) for a third-order tensor.

Proof. We denote $r := \text{CPRank}(\mathcal{X})$. Suppose $\mathcal{X} = [[\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_N]]$ is a CP decomposition of tensor $\mathcal{X}$, where $\mathbf{A}_1 = (a^{(1)}_{i_1 p}) \in \Re^{I_1 \times r}$, $\mathbf{A}_2 = (a^{(2)}_{i_2 p}) \in \Re^{I_2 \times r}$,..., and $\mathbf{A}_N = (a^{(N)}_{i_n p}) \in \Re^{I_N \times r}$ are factor matrices. Denote $\mathcal{A}_1 \in \Re^{I_1 \times r \times ... \times r}$, $\mathcal{A}_2 \in \Re^{r \times I_2 \times ... \times r}$, ... , and $\mathcal{A}_N \in \Re^{r \times r \times ... \times I_N}$ with $(\mathcal{A}_1)_{i_1 p_2 ... p_N} = a^{(1)}_{i_1 p_2}$ for $p_2 = p_3 = ... = p_N$ and $(\mathcal{A}_1)_{i_1 p_2 ... p_N} = 0$ otherwise; $(\mathcal{A}_2)_{p_1 i_2 ... p_N} = a^{(2)}_{i_2 p_1}$ for $p_1 = p_3 = ... = p_N$ and $(\mathcal{A}_2)_{p_1 i_2 ... p_N} = 0$ otherwise; ... ; $(\mathcal{A}_N)_{p_1 p_2 ... I_N} = a^{(N)}_{i_N p_1}$ for $p_1 = p_2 = ... = p_{N-1}$. Then for all $i_n = 1, \ldots, I_n$ and $n = 1, \ldots, N$, there holds

$$\begin{aligned}[\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_N]_{i_1 i_2 ... i_N} &= \sum_{p_1=1}^{r} \sum_{p_2=1}^{r} ... \sum_{p_N=1}^{r} (\mathcal{A}_1)_{i_1 p_2 ... p_N} (\mathcal{A}_2)_{p_1 i_2 ... p_N} ... (\mathcal{A}_N)_{p_1 p_2 ... i_N} \\ &= \sum_{p=1}^{r} a^{(1)}_{i_1 p} a^{(2)}_{i_2 p} ... a^{(N)}_{i_N p} = \mathcal{X}_{i_1 i_2 ... i_N}.\end{aligned}$$

This means $\mathcal{X} = [\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_N]$ and $\text{GTriRank}(\mathcal{X}) \leq \text{CPRank}(\mathcal{X})$. According to Theorem 3.2, we have $\max\{r_1, r_2, ..., r_N\} \leq \text{CPRank}(\mathcal{X})$. On the other hand, since $\text{MtpRank}(\mathcal{X})=(r_1, r_2, ..., r_N)$, There exists $\mathcal{B}_1 \in \Re^{I_1 \times r_2 \times ... \times r_N}$, $\mathcal{B}_2 \in \Re^{r_1 \times I_2 \times ... \times r_N}$, ..., and $\mathcal{B}_N \in \Re^{r_1 \times r_2 \times ... \times r_{N-1} \times I_N}$, s.t.

$$\mathcal{X}_{i_1 i_2 ... i_N} = \sum_{p_1=1}^{r_1} \sum_{p_2=1}^{r_2} ... \sum_{p_N=1}^{r_N} (\mathcal{B}_1)_{i_1 p_2 ... p_N} (\mathcal{B}_2)_{p_1 i_2 ... p_N} ... (\mathcal{B}_N)_{p_1 p_2 ... i_N}.$$

Therefore, $\mathcal{X}$ can be represented as a sum of $r_1 r_2 ... r_N$ rank-one tensors, and the last inequality in the theorem holds. ∎

The second inequality in (3.8) can hold with equality (see Example 3.5 in [25]), in which case the CP rank can be much larger than the Multiple rank.

3.1.2. The relation between the Multiple and Tucker decompositions. If tensor $\mathcal{X} = [\mathcal{A}_1 \mathcal{A}_2 ... \mathcal{A}_N]$ with $\mathcal{A}_1 = \mathcal{F}_1 \times_1 \mathbf{U}_1$, $\mathcal{F}_1 \in \Re^{R_1 \times r_2 \times ... \times r_N}$, $\mathbf{U}_1 \in \Re^{I_1 \times R_1}$, $\mathcal{A}_2 = \mathcal{F}_2 \times_2 \mathbf{U}_2$, $\mathcal{F}_2 \in \Re^{r_1 \times R_2 \times ... \times r_N}$, $\mathbf{U}_2 \in \Re^{I_2 \times R_2}$, ..., and $\mathcal{A}_N = \mathcal{F}_N \times_N \mathbf{U}_N$,

$\mathcal{F}_N \in \Re^{r_1 \times r_2 \times \ldots \times R_N}$, $\mathbf{U}_N \in \Re^{I_N \times R_N}$, then we have

$$
\begin{aligned}
\mathcal{X}_{i_1 i_2 \ldots i_N} &= \sum_{p_1=1}^{r_1} \sum_{p_2=1}^{r_2} \ldots \sum_{p_N=1}^{r_N} (\mathcal{A}_1)_{i_1 p_2 \ldots p_N} (\mathcal{A}_2)_{p_1 i_2 \ldots p_N} \ldots (\mathcal{A}_N)_{p_1 p_2 \ldots p_{N-1} i_N} \\
&= \sum_{j_1=1}^{R_1} \sum_{j_2=1}^{R_2} \ldots \sum_{j_N=1}^{R_N} (\mathbf{U}_1)_{i_1 j_1} (\mathbf{U}_2)_{i_2 j_2} \ldots (\mathbf{U}_N)_{i_N j_N} \\
&\quad \cdot \underbrace{\sum_{p_1=1}^{r_1} \sum_{p_2=1}^{r_2} \ldots \sum_{p_N=1}^{r_N} (\mathcal{F}_1)_{j_1 p_2 \ldots p_N} (\mathcal{F}_2)_{p_1 j_2 \ldots p_N} \ldots (\mathcal{F}_N)_{p_1 p_2 \ldots p_{N-1} j_N}}_{([\mathcal{F}_1 \mathcal{F}_2 \ldots \mathcal{F}_N])_{j_1 j_2 \cdots j_N}},
\end{aligned} \tag{3.9}
$$

which corresponds to the formulation in (2.2). This implies that

$$
[(\mathcal{F}_1 \times_1 \mathbf{U}_1)(\mathcal{F}_2 \times_2 \mathbf{U}_2) \ldots (\mathcal{F}_N \times_N \mathbf{U}_N)] = [\mathcal{F}_1 \mathcal{F}_2 \ldots \mathcal{F}_N] \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \ldots \times_n \mathbf{U}_N. \tag{3.10}
$$

Based on (3.10), we can derive the following conclusions.

THEOREM 3.5. *Suppose* $\mathcal{X} \in \Re^{I_1 \times I_2 \times \ldots \times I_N}$ *with* $\text{TuckRank}(\mathcal{X}) = (t_1, t_2, ..., t_N)$. *Let* $\mathcal{X} = \mathcal{D} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times \ldots \times_N \boldsymbol{U}_N$ *be a Tucker decomposition of* $\mathcal{X}$ *with* $\mathcal{D} \in \Re^{t_1 \times t_2 \times \ldots \times t_N}$ *and factor matrices* $\boldsymbol{U}_1 \in \Re^{I_1 \times t_1}$, $\boldsymbol{U}_2 \in \Re^{I_2 \times t_2}$,..., $\boldsymbol{U}_N \in \Re^{I_N \times t_N}$. *If* $\boldsymbol{U}_i^\top \boldsymbol{U}_i$, $i = 1, 2, ..., N$ *are invertible, then one has*

$$
\text{MtpRank}(\mathcal{X}) = \text{MtpRank}(\mathcal{D}) \leq \text{submax}\{t_1, t_2, ..., t_N\}. \tag{3.11}
$$

Proof. Assume $\mathcal{D} = [\widetilde{\mathcal{A}}_1 \widetilde{\mathcal{A}}_2 \ldots \widetilde{\mathcal{A}}_N]$ with $\widetilde{\mathcal{A}}_1 \in \Re^{t_1 \times r_2 \times \ldots \times r_N}$, $\widetilde{\mathcal{A}}_2 \in \Re^{r_1 \times t_2 \times \ldots \times r_N}$, ..., and $\widetilde{\mathcal{A}}_N \in \Re^{r_1 \times r_2 \times \ldots \times t_N}$. Then from (3.10), one has

$$
\begin{aligned}
\mathcal{X} &= [\widetilde{\mathcal{A}}_1 \widetilde{\mathcal{A}}_2 \ldots \widetilde{\mathcal{A}}_N] \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \ldots \times_n \mathbf{U}_N \\
&= [(\widetilde{\mathcal{A}}_1 \times_1 \mathbf{U}_1)(\widetilde{\mathcal{A}}_2 \times_2 \mathbf{U}_2) \ldots (\widetilde{\mathcal{A}}_N \times_N \mathbf{U}_N)].
\end{aligned} \tag{3.12}
$$

Clearly, $\widetilde{\mathcal{A}}_1 \times_1 \mathbf{U}_1 \in \Re^{I_1 \times r_2 \times \ldots \times r_N}$, $\widetilde{\mathcal{A}}_2 \times_2 \mathbf{U}_2 \in \Re^{r_1 \times I_2 \times \ldots \times r_N}$,..., and $\widetilde{\mathcal{A}}_N \times_N \mathbf{U}_N \in \Re^{r_1 \times r_2 \times \ldots \times I_N}$. Since $\mathbf{U}_i^\top \mathbf{U}_i$, $i = 1, 2, ..., N$ are invertible, one has

$$
\begin{aligned}
&\mathcal{X} \times_1 (\mathbf{U}_1^\top \mathbf{U}_1)^{-1} \mathbf{U}_1^\top \times_2 (\mathbf{U}_2^\top \mathbf{U}_2)^{-1} \mathbf{U}_2^\top \times \ldots \times_N (\mathbf{U}_N^\top \mathbf{U}_N)^{-1} \mathbf{U}_N^\top \\
&= \mathcal{D} \times_1 (\mathbf{U}_1^\top \mathbf{U}_1)^{-1} (\mathbf{U}_1^\top \mathbf{U}_1) \times_2 (\mathbf{U}_2^\top \mathbf{U}_2)^{-1} (\mathbf{U}_2^\top \mathbf{U}_2) \times \ldots \times_N (\mathbf{U}_N^\top \mathbf{U}_N)^{-1} (\mathbf{U}_N^\top \mathbf{U}_N) \\
&= \mathcal{D} \times_1 \mathbf{I}_{r_1} \times_2 \mathbf{I}_{r_2} \times \ldots \times_N \mathbf{I}_{r_N} = \mathcal{D}.
\end{aligned}
$$

Assume that $\text{MtpRank}(\mathcal{X}) = (R_1, R_2, ..., R_N)$, then there exists $\mathcal{A}_1 \in \Re^{I_1 \times R_2 \times \ldots \times R_N}$, $\mathcal{A}_2 \in \Re^{R_1 \times I_2 \times \ldots \times R_N}$, ..., and $\mathcal{A}_N \in \Re^{R_1 \times R_2 \times \ldots \times R_{N-1} \times I_N}$, s.t., $\mathcal{X} = [\mathcal{A}_1 \mathcal{A}_2 \ldots \mathcal{A}_N]$. Hence, it holds that

$$
\begin{aligned}
\mathcal{D} &= [\mathcal{A}_1 \mathcal{A}_2 \ldots \mathcal{A}_N] \times_1 (\mathbf{U}_1^\top \mathbf{U}_1)^{-1} \mathbf{U}_1^\top \times_2 (\mathbf{U}_2^\top \mathbf{U}_2)^{-1} \mathbf{U}_2^\top \times \ldots \times_N (\mathbf{U}_N^\top \mathbf{U}_N)^{-1} \mathbf{U}_N^\top \\
&= [(\mathcal{A}_1 \times_1 (\mathbf{U}_1^\top \mathbf{U}_1)^{-1} \mathbf{U}_1^\top)(\mathcal{A}_2 \times_2 (\mathbf{U}_2^\top \mathbf{U}_2)^{-1} \mathbf{U}_2^\top) \ldots (\mathcal{A}_N \times_N (\mathbf{U}_N^\top \mathbf{U}_N)^{-1} \mathbf{U}_N^\top)].
\end{aligned}
$$

We can find that $\mathcal{A}_1 \times_1 (\mathbf{U}_1^\top \mathbf{U}_1)^{-1} \mathbf{U}_1^\top \in \Re^{t_1 \times R_2 \times \ldots \times R_N}$, $\mathcal{A}_2 \times_2 (\mathbf{U}_2^\top \mathbf{U}_2)^{-1} \mathbf{U}_2^\top \in \Re^{R_1 \times t_2 \times \ldots \times R_N}$, and $\mathcal{A}_N \times_N (\mathbf{U}_N^\top \mathbf{U}_N)^{-1} \mathbf{U}_N^\top \in \Re^{R_1 \times R_2 \times \ldots \times t_N}$. Combined with (3.12), it implies that the factor tensors of $\mathcal{D}$ and those of $\mathcal{X}$ can be transformed into each other. Therefore, we have $\text{MtpRank}(\mathcal{X}) = \text{MtpRank}(\mathcal{D})$. ∎

In fact, there is also a close connection between the tensor ranks of $\mathcal{X}$ and those of its factor tensors in generalized triple decomposition.

THEOREM 3.6. *Let $\mathcal{X} = [\mathcal{A}_1\mathcal{A}_2...\mathcal{A}_N]$ be the generalized triple decomposition; then for $n = 1, 2, ..., N$, one has*

$$\text{TuckRank}(\mathcal{X})_n \leq \text{TuckRank}(\mathcal{A}_n)_n \leq \text{GTriRank}(\mathcal{A}_n)^{N-1} \leq \text{GTriRank}(\mathcal{X})^{N-1}.$$

Proof. Suppose $\text{GTriRank}(\mathcal{X}) = r$, $\text{TuckRank}(\mathcal{A}_1)_1 = r_1$, $\text{TuckRank}(\mathcal{A}_2)_2 = r_2$,..., and $\text{TuckRank}(\mathcal{A}_N)_N = r_N$. Let $\mathcal{A}_1 = \mathcal{F} \times_1 \mathbf{U}_1 \times_2 \mathbf{V}_2 \times ... \times_N \mathbf{V}_n$ be a Tucker decomposition of $\mathcal{A}_1$ with core tensor $\mathcal{F} \in \Re^{r_1 \times s_2 \times ... \times s_N}$ and factor matrices $\mathbf{U}_1 \in \Re^{I_1 \times r_1}$, $\mathbf{V}_2 \in \Re^{r \times s_2}$,..., $\mathbf{V}_N \in \Re^{r \times s_N}$. Denote $\bar{\mathcal{A}}_1 = \mathcal{F} \times_2 \mathbf{V}_2 \times ... \times_N \mathbf{V}_N \in \Re^{r_1 \times r \times ... \times r}$. Then $\mathcal{A}_1 = (\mathcal{F} \times_2 \mathbf{V}_2 \times ... \times_N \mathbf{V}_N) \times_1 \mathbf{U}_1 = \bar{\mathcal{A}}_1 \times_1 \mathbf{U}_1$. Similarly, there exist $\bar{\mathcal{A}}_2 \in \Re^{r \times r_2 \times ... \times r}$,..., $\bar{\mathcal{A}}_N \in \Re^{r \times r \times ... \times r_N}$, and $\mathbf{U}_2 \in \Re^{I_2 \times r_2}$,..., $\mathbf{U}_N \in \Re^{I_N \times r_N}$ such that $\mathcal{A}_1 = \bar{\mathcal{A}}_1 \times_1 \mathbf{U}_1$, $\mathcal{A}_2 = \bar{\mathcal{A}}_2 \times_2 \mathbf{U}_2$, ..., $\mathcal{A}_N = \bar{\mathcal{A}}_N \times_N \mathbf{U}_N$. Hence, $\mathcal{X} = [\mathcal{A}_1\mathcal{A}_2...\mathcal{A}_N] = ([\bar{\mathcal{A}}_1\bar{\mathcal{A}}_2...\bar{\mathcal{A}}_N]) \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times ... \times_N \mathbf{U}_N$ according to (3.5). From the definition of Tucker rank, for $n = 1, 2, ..., N$, we have

$$\text{TuckRank}(\mathcal{X})_n \leq \text{TuckRank}(\mathcal{A}_n)_n. \tag{3.13}$$

Assume that $\text{GTriRank}(\mathcal{A}_1) = \bar{r}$. Then there are $\hat{\mathcal{B}}_1 \in \Re^{I_1 \times \bar{r} \times ... \times \bar{r}}$, $\hat{\mathcal{B}}_2 \in \Re^{\bar{r} \times r \times ... \times \bar{r}}$,..., and $\hat{\mathcal{B}}_N \in \Re^{\bar{r} \times \bar{r} \times ... \times r}$ such that $\mathcal{A}_1 = [\hat{\mathcal{B}}_1\hat{\mathcal{B}}_2...\hat{\mathcal{B}}_N]$. Replacing $\mathcal{X}$ and $\mathcal{A}_1$ in the first inequality of (3.13) by $\mathcal{A}_1$ and $\hat{\mathcal{B}}_1$, we have $\text{TuckRank}(\mathcal{A}_1)_1 \leq \text{TuckRank}(\hat{\mathcal{B}}_1)_1$. Note that $\hat{\mathcal{B}}_1 \in \Re^{n_1 \times \bar{r} \times ... \times \bar{r}}$. By the definition of the Tucker rank, $\text{TuckRank}(\hat{\mathcal{B}}_1)_1$ is the rank of an $n_1 \times \bar{r}^{N-1}$ matrix. Hence, $\text{TuckRank}(\hat{\mathcal{B}}_1)_1 \leq \bar{r}^{N-1}$. In a similar way, we can prove

$$\text{TuckRank}(\mathcal{A}_n)_n \leq \text{GTriRank}(\mathcal{A}_n)^{N-1}. \tag{3.14}$$

Since $\mathcal{A}_1 = [\hat{\mathcal{B}}_1\hat{\mathcal{B}}_2...\hat{\mathcal{B}}_N]$ and $\mathcal{A}_1 \in \Re^{n_1 \times r \times ... \times r}$, it follows from Theorem 3.1 that $\text{GTriRank}(\mathcal{A}_1) \leq r = \text{GTriRank}(\mathcal{X})$. Then for $n = 2, 3, ..., N$, we can prove in a similar way. Therefore, the third inequality of Theorem 3.6 holds. ∎

Next, we introduce the concept of hybrid multiple decomposition.

Definition 3.3. **(Hybrid multiple decomposition)** Suppose the tensor $\mathcal{X} \in \Re^{I_1 \times I_2 \times ... \times I_N}$ has Multiple rank $\text{MtpRank}(\mathcal{X}) = (R_1, R_2, ..., R_N)$ and Tucker rank $\text{TuckRank}(\mathcal{X}) = (r_1, r_2, ..., r_N)$. Consider a Tucker decomposition of $\mathcal{X}$: $\mathcal{X} = \mathcal{D} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times ... \times_N \mathbf{U}_N$, where $\mathcal{D} \in \Re^{r_1 \times r_2 \times ... \times r_N}$ and $\mathbf{U}_1 \in \Re^{I_1 \times r_1}$, $\mathbf{U}_2 \in \Re^{I_2 \times r_2}$,..., $\mathbf{U}_N \in \Re^{I_N \times r_N}$. Furthermore, if $\mathbf{U}_i^\top \mathbf{U}_i$, $i = 1, 2, ..., N$ are invertible and the core tensor $\mathcal{D}$ has Multiple decomposition $\mathcal{D} = [\mathcal{A}_1\mathcal{A}_2...\mathcal{A}_N]$ with $\mathcal{A}_1 \in \Re^{r_1 \times R_2 \times ... \times R_N}$, $\mathcal{A}_2 \in \Re^{R_1 \times r_2 \times ... \times R_N}$,..., and $\mathcal{A}_N \in \Re^{R_1 \times R_2 \times ... \times r_N}$, then we call

$$\mathcal{X} = [\mathcal{A}_1\mathcal{A}_2...\mathcal{A}_N] \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times ... \times_N \mathbf{U}_N = [(\mathcal{A}_1 \times_1 \mathbf{U}_1)(\mathcal{A}_2 \times_2 \mathbf{U}_2)...(\mathcal{A}_N \times_N \mathbf{U}_N)]$$

a *hybrid multiple decomposition* of $\mathcal{X}$.

The hybrid multiple decomposition can further exploit the latent correlations along the long dimensions of the factor tensors in Multiple decomposition. This is especially effective when strong correlations exist within individual modes.

3.1.3. The relation between Multiple decomposition and tensor network decompositions. Next, we compare Multiple decomposition with several classical tensor network decompositions, including Tensor Train (TT) decomposition [23],

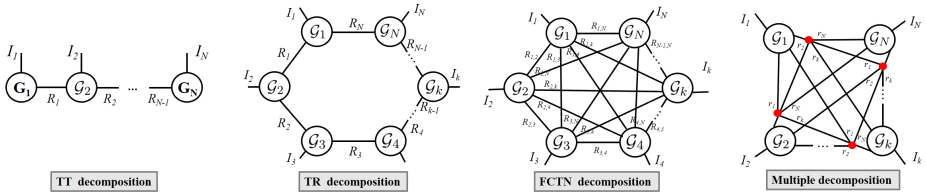


Fig. 2: Illustration of the TT, TR, FCTN and the proposed Multiple decomposition.

Tensor Ring (TR) decomposition [38] and Fully Connected Tensor Network (FCTN) decomposition [43]. Figure 2 illustrates the schematic diagrams of these decompositions. From the figure, we can find that

- TT and TR decompositions rely on sequential, nearest-neighbor interactions, which may hinder direct modeling of long-range or cross-mode correlations. Although Multiple decomposition does not establish explicit connections between every pair of factor tensors, it enables global interactions through shared virtual indices in the summation process.
- The FCTN decomposition establishes interactions between every pair of factor tensors, enabling it to fully capture the correlations among all modes. However, when decomposing an N-th order tensor, the FCTN rank involves a vector with $N(N-1)/2$ elements, leading to significantly increased computational complexity. In contrast, Multiple decomposition captures global correlations via implicit coupling through shared latent dimensions, while the Multiple rank takes the form of an N-dimensional vector, resulting in lower storage and computational costs for high-order tensors.

For third-order tensors, the TR, FCTN, and Multiple decompositions share structural similarities. Triple decomposition imposes the constraint that the short-side dimensions of the factor tensors must be identical, which can be viewed as a constrained instance. In the case of tensors with order four or above, the TR, FCTN, and Multiple decompositions exhibit distinct structural and representational characteristics. Due to the distinct modeling mechanism, Multiple decomposition is not a special case of TR or FCTN, nor is it fully encompassed by them in higher-order settings.

3.2. Implicit Multiple Tensor Decomposition. Non-grid data, such as point clouds, may also possess intrinsic low-rank structures [44]. Current tensor decompositions are primarily designed for regular grid-structured data and cannot be directly applied to non-grid data. To address this issue, we integrate Multiple decomposition with implicit neural representation (INR), which models data as a continuous function of coordinates. This leads to the proposed Implicit Multiple Tensor Decomposition (IMTD). Specifically, we regard the data $\mathcal{X}$ as a bounded tensor function

$$f_{\mathcal{X}}(\cdot) : D_{\mathcal{X}} := \Omega_1 \times \Omega_2 \times ... \times \Omega_N \to \Re.$$

When $\mathcal{X} \in \Re^{I_1 \times I_2 \times \cdots \times I_N}$ is a regular grid tensor, each index set Ω_n can be defined as $\Omega_n := \{1, 2, \ldots, I_n\}$ for $n = 1, 2, \ldots, N$. In the case of non-grid data, Ω_n are continuous subsets of $\Re$, and observations are given at irregularly sampled coordinates within $D_{\mathcal{X}}$. Then, we represent each factor tensor $\mathcal{A}_n (n = 1, 2, ..., N)$ as a continuous function $\mathcal{A}_{\Theta_n}(\cdot)$ via a neural network parameterized by Θ_n. Once the parameters $\Theta_1, \Theta_2, ..., \Theta_N$ are trained, we can obtain the factor tensor for the desired dimension by inputting the coordinates of that dimension. Taking $\mathcal{A}_{\Theta_1}(\cdot)$ as an example, it can

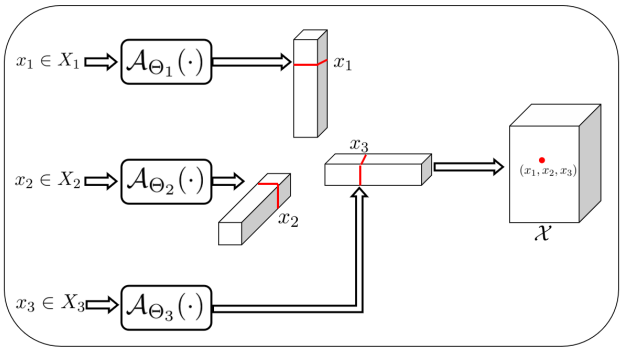


Fig. 3: The flowchart of the IMTD (taking a third-order tensor as an example).

be formulated as

$$\mathcal{A}_{\Theta_1}(x_1) := \text{reshape}(\mathbf{H}_d(\sigma(\mathbf{H}_{d-1} \cdots \sigma(\mathbf{H}_1 x_1)))) : \Omega_1 \to \Re^{1 \times r_2 \times r_3 \times \ldots \times r_N}, \tag{3.15}$$

where $\Omega_1 \subset \Re$ is the definition domain in the first dimension, $\sigma(\cdot)$ is the nonlinear activation function, the 'reshape' operation reorganizes the vector into a tensor of the specified size, and $\Theta_1 := \{\mathbf{H}_i\}_{i=1}^d$ are learnable weight matrices of the Multilayer Perceptron (MLP). The MLP-parameterized IMTD is formulated as

$$f_{\mathcal{X}}(\mathbf{x}) = [\mathcal{A}_{\Theta_1}(x_1)\mathcal{A}_{\Theta_2}(x_2)...\mathcal{A}_{\Theta_N}(x_N)] \in \Re, \tag{3.16}$$

where $\mathbf{x} = (x_1, x_2, ..., x_N) \in \Re^n$ is the coordinate of the observation point. The flowchart of the IMTD for the third-order tensor case is shown in Figure 3. Through the above procedure, we can solve the IMTD by optimizing the network parameters.

Remark 3.3. By leveraging Property 3 of Theorem 3.3, the factor tensors can be decoupled and optimized within Block Coordinate Descent (BCD) framework, with convergence guarantees established following the analysis in [25]. However, this approach relies on grid-structured factor tensors. In contrast, IMTD represents the factor tensors as continuous implicit functions, enabling the model to generalize across arbitrary coordinates. Therefore, IMTD can also capture complex nonlinear patterns and uncover the underlying low-rank structure in non-grid data.

THEOREM 3.7. *Let the mappings* $\mathcal{A}_{\Theta_1}(\cdot) : \Omega_1 \to \mathbb{R}^{1 \times r_2 \times r_3 \times \ldots \times r_N}$, $\mathcal{A}_{\Theta_2}(\cdot) : \Omega_2 \to \mathbb{R}^{r_1 \times 1 \times r_3 \times \ldots \times r_N}$,..., *and* $\mathcal{A}_{\Theta_N}(\cdot) : \Omega_N \to \mathbb{R}^{r_1 \times r_2 \times \ldots \times r_{N-1} \times 1}$ *be* N *MLPs structured as in (3.15) with parameters* Θ_1, Θ_2,..., Θ_N, *where* $\Omega_1, \Omega_2, ..., \Omega_N \subset \mathbb{R}$ *are bounded. Let the MLPs share the same activation function* $\sigma(\cdot)$ *and depth* d. *We assume that*

- $\sigma(\cdot)$ *is Lipschitz continuous with Lipschitz constant* κ.
- *The* ℓ_1 *norm of each weight matrix* $\mathbf{H}_i$ *in the MLPs is bounded by* $\omega > 0$.

Define the function $f(\cdot) : D_f = \Omega_1 \times \Omega_2 \times ... \times \Omega_N \to \mathbb{R}$ *as* $f(\cdot) := [\mathcal{A}_{\Theta_1}\mathcal{A}_{\Theta_2}...\mathcal{A}_{\Theta_N}](\cdot)$. *Then, for any* $\boldsymbol{x} = (x_1, x_2, ..., x_N), \boldsymbol{y} = (y_1, y_2, ..., y_N) \in D_f$, *one has*

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq \delta \|\boldsymbol{x} - \boldsymbol{y}\|,$$

i.e., f *is Lipschitz continious in* D_f, *where* $\delta = \sqrt{2}\omega^{Nd}\kappa^{Nd-N}\zeta^{N-1}$ *and* $\zeta = \|\boldsymbol{x}\|_\infty$.

Proof. For any $(x_1, x_2, ..., x_N), (y_1, y_2, ..., y_N) \in D_f$, direct calculation yields

$$\begin{aligned}
&|f(x_1, x_2, ..., x_N) - f(y_1, x_2, ..., x_N)| \\
&= |[\mathcal{A}_{\Theta_1}(x_1)\mathcal{A}_{\Theta_2}(x_2)...\mathcal{A}_{\Theta_N}(x_N)] - [\mathcal{A}_{\Theta_1}(y_1)\mathcal{A}_{\Theta_2}(x_2)...\mathcal{A}_{\Theta_N}(x_N)]| \\
&= |[(\mathcal{A}_{\Theta_1}(x_1) - \mathcal{A}_{\Theta_1}(y_1))\mathcal{A}_{\Theta_2}(x_2)...\mathcal{A}_{\Theta_N}(x_N)]| \\
&\leq \|\mathcal{A}_{\Theta_1}(x_1) - \mathcal{A}_{\Theta_1}(y_1)\|_{\ell_1}\|\mathcal{A}_{\Theta_2}(x_2)\|_{\ell_1}...\|\mathcal{A}_{\Theta_N}(x_N)\|_{\ell_1}.
\end{aligned}$$

Note that $\sigma(\cdot)$ is Lipschitz continuous, i.e., $|\sigma(x) - \sigma(y)| \leq \kappa|x - y|$ holds for any x, y, and letting $y = 0$ derives $|\sigma(x)| \leq \kappa|x|$ since $\sigma(0) = \sin(0) = 0$. Thus, we have

$$\begin{aligned}
\|\mathcal{A}_{\Theta_2}(x_2)\|_{\ell_1} &= \|\mathbf{H}_d^{(2)}(\sigma(\mathbf{H}_{d-1}^{(2)} \cdots \mathbf{H}_2^{(2)}(\sigma(\mathbf{H}_1^{(2)}x_2))))\|_{\ell_1} \\
&\leq \omega\|\sigma(\mathbf{H}_{d-1}^{(2)} \cdots \mathbf{H}_2^{(2)}(\sigma(\mathbf{H}_1^{(2)}x_2)))\|_{\ell_1} \\
&\leq \omega\kappa\|\mathbf{H}_{d-1}^{(2)} \cdots \mathbf{H}_2^{(2)}(\sigma(\mathbf{H}_1^{(2)}x_2))\|_{\ell_1} \\
&... \\
&\leq \omega^d\kappa^{d-1}|x_2|,
\end{aligned}$$

where $\{\mathbf{H}_i^{(2)}\}_{i=1}^d$ are the weight matrices of $\mathcal{A}_{\Theta_2}(\cdot)$. Similarly, we have $\|\mathcal{A}_{\Theta_n}(x_n)\|_{\ell_1} \leq \omega^d\kappa^{d-1}|x_n|$ for $n = 3, 4, ..., N$. Meanwhile, it holds that

$$\begin{aligned}
&\|\mathcal{A}_{\Theta_1}(x_1) - \mathcal{A}_{\Theta_1}(y_1)\|_{\ell_1} \\
&= \|\mathbf{H}_d^{(1)}(\sigma(\mathbf{H}_{d-1}^{(1)} \cdots \mathbf{H}_2^{(1)}(\sigma(\mathbf{H}_1^{(1)}x_1)))) - \mathbf{H}_d^{(1)}(\sigma(\mathbf{H}_{d-1}^{(1)} \cdots \mathbf{H}_2^{(1)}(\sigma(\mathbf{H}_1^{(1)}y_1))))\|_{\ell_1} \\
&= \|\mathbf{H}_d^{(1)}(\sigma(\mathbf{H}_{d-1}^{(1)} \cdots \mathbf{H}_2^{(1)}(\sigma(\mathbf{H}_1^{(1)}x_1))) - \sigma(\mathbf{H}_{d-1}^{(1)} \cdots \mathbf{H}_2^{(1)}(\sigma(\mathbf{H}_1^{(1)}y_1))))\|_{\ell_1} \\
&\leq \omega\|\sigma(\mathbf{H}_{d-1}^{(1)} \cdots \mathbf{H}_2^{(1)}(\sigma(\mathbf{H}_1^{(1)}x_1))) - \sigma(\mathbf{H}_{d-1}^{(1)} \cdots \mathbf{H}_2^{(1)}(\sigma(\mathbf{H}_1^{(1)}y_1)))\|_{\ell_1} \\
&\leq \omega\kappa\|\mathbf{H}_{d-1}^{(1)} \cdots \mathbf{H}_2^{(1)}(\sigma(\mathbf{H}_1^{(1)}x_1)) - \mathbf{H}_{d-1}^{(1)} \cdots \mathbf{H}_2^{(1)}(\sigma(\mathbf{H}_1^{(1)}y_1))\|_{\ell_1} \\
&\quad ... \\
&\leq \omega^d\kappa^{d-1}|x_1 - y_1|,
\end{aligned}$$

where $\{\mathbf{H}_i^{(1)}\}_{i=1}^d$ denote the weight matrices of $\mathcal{A}_{\Theta_1}(\cdot)$. Combining the above inequalities, we have

$$\begin{aligned}
|f(x_1, x_2, ..., x_N) - f(y_1, x_2, ..., x_N)| &\leq \omega^{Md}\kappa^{Nd-N}|x_2|...|x_N||x_1 - x_2| \\
&\leq \omega^{Nd}\kappa^{Nd-N}\zeta^{N-1}|x_1 - y_1|.
\end{aligned}$$

In a similar way, we can show that

$$|f(x_1, x_2, ..., x_N) - f(x_1, y_2, ..., x_N)| \leq \omega^{Nd}\kappa^{Nd-N}\zeta^{N-1}|x_2 - y_2|,$$

$$|f(x_1, x_2, ..., x_N) - f(x_1, x_2, ..., y_N)| \leq \omega^{Nd}\kappa^{Nd-N}\zeta^{N-1}|x_N - y_N|.$$

Based on the above inequalities, we can deduce

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq \delta\|\mathbf{x} - \mathbf{y}\|,$$

for any $\mathbf{x} = (x_1, x_2, ..., x_N), \mathbf{y} = (y_1, y_2, ..., y_N) \in D_f$, which completes the proof. ∎

4. Model, algorithm and convergence analysis. In this section, we investigate the tensor recovery models within the IMTD framework, along with corresponding algorithms and convergence analysis.

4.1. The tensor reconstruction problems within IMTD framework. We first consider the case where the model involves only a single variable. Within the IMTD framework, the models can be generally formulated as

$$\min_{\Theta} \ F(\mathcal{A}_\Theta(\mathbf{x}); \mathcal{M}) + \lambda\phi(\mathcal{A}_\Theta(\mathbf{x})), \tag{4.1}$$

where $\mathcal{M}$ is the observed data, $\mathcal{A}_\Theta(\cdot) := [\mathcal{A}_{\Theta_1}\mathcal{A}_{\Theta_2}...\mathcal{A}_{\Theta_N}](\cdot)$ is a tensor function representing the reconstructed tensor in IMTD form, $\Theta := (\Theta_1, \Theta_2, ..., \Theta_N)$ denotes the network parameter, $\mathbf{x}$ is the coordinates of tensor, $F(\cdot)$ represents the fidelity term and $\phi(\cdot)$ is the regularization term, $\lambda > 0$ is a weight parameter. We also consider the tensor reconstruction problem with two variates, which can be formulated as:

$$\min_{\Theta,\mathcal{E}} \ G(\Theta, \mathcal{E}) := F(\mathcal{A}_\Theta(\mathbf{x}), \mathcal{E}; \mathcal{M}) + \lambda\phi(\mathcal{A}_\Theta(\mathbf{x})) + \gamma\psi(\mathcal{E}), \tag{4.2}$$

where $\mathcal{E}$ is typically treated as a perturbation variable, $\psi(\mathcal{E})$ is a regularization term, and $\gamma > 0$ is a weight parameter. We assume that F is continuously differentiable, ϕ is lower semi-continious (LSC), ψ is LSC and coercive. Next, we present two classical reconstruction problems under the framework of (4.1) and (4.2).

1) Robust Tensor Completion (RTC). This is a classical problem on the original meshgrid tensor, aiming to recover the underlying low rank tensor from partial observations. Specifically, given an observed tensor $\mathcal{M}$ corrupted by sparse noise and an index set Ω of observed entries, the RTC problem can be formulated within the framework (4.2) as follows:

$$\min_{\Theta,\mathcal{E}} \ \|P_\Omega(\mathcal{A}_\Theta(\mathbf{x}) + \mathcal{E} - \mathcal{M})\|_F^2 + \lambda\|\mathcal{A}_\Theta(\mathbf{x})\|_{\mathrm{TV}\ell_1} + \gamma\|\mathcal{E}\|_{\ell_1}, \tag{4.3}$$

where $\|\cdot\|_{\mathrm{TV}\ell_1} := \|\mathcal{D}(\cdot)\|_{\ell_1}$ denotes the Total Variation (TV) regularization in the ℓ_1 sense, $\mathcal{D}(\cdot)$ represents a discrete difference operator that computes the first-order differences along each mode of the tensor [4].

2) Point Cloud Upsampling (PCU) refers to the process of converting a sparse point cloud into a dense one, which can benefit subsequent applications [21]. For a sparse point cloud $\mathbf{P} \in \mathbb{R}^{p\times 3}$, where p denotes the number of points, we use $\Omega := \{\mathbf{P}_{(m,:)}\}_{m=1}^p$ to denote the observed set, and adopt signed distance function (SDF) [24] to learn a continuous representation from the sparse point cloud. The PCU model within the IMTD framework can be expressed as

$$\begin{aligned} &\min_s \ \sum_{\mathbf{x}\in\Omega} |s(\mathbf{x})| + \lambda \int_{\mathbb{R}^3} \left| \left\| \frac{\partial s(\mathbf{x})}{\partial \mathbf{x}} \right\|_F^2 - 1 \right| d\mathbf{x} + \gamma \int_{\mathbb{R}^3\setminus\Omega} \exp(-|s(\mathbf{x})|) d\mathbf{v}, \\ &s.t. \ \ s(\mathbf{x}) = \mathcal{A}_\Theta(\mathbf{x}) := [\mathcal{A}_{\Theta_1}\mathcal{A}_{\Theta_2}...\mathcal{A}_{\Theta_N}](\mathbf{x}), \end{aligned}$$

where $s(\cdot) : \mathbb{R}^3 \to \mathbb{R}$ denotes the SDF, λ and γ are weight parameters. The first term enforces the SDF to be zero on the observed points, which represents the underlying surface of the point cloud. The second term encourages the magnitude of the SDF gradients to be unity everywhere, promoting well-behaved geometry. The third term penalizes deviations of the SDF values from zero outside the observed point set [30, 21]. To generate a dense point cloud, we sample points $\mathbf{v}$ uniformly across the space such that $|s(\mathbf{v})| < \tau$, where τ is a predefined threshold. These sampled points constitute the desired upsampling result.

4.2. Algorithm and convergence analysis. Since Θ is the true optimization variable in $\mathcal{A}_\Theta(\mathbf{x})$, we denote $\mathcal{A}_\Theta(\mathbf{x})$ as $\mathcal{A}_\mathbf{x}(\Theta)$ or $\mathcal{A}_\Theta$. To solve the model (4.1), we can define the loss function as $F(\mathcal{A}_\Theta;\mathcal{M}) + \lambda\phi(\mathcal{A}_\Theta)$, and directly optimize Θ using adaptive moment estimation (Adam) [14] algorithm. However, for the model (4.2), directly optimizing a multi-variable loss function may cause conflicts, leading to degraded performance. Therefore, we employ the Proximal Alternating Least Squares (PALS) algorithm to solve the model. The iterative scheme is

$$\Theta^{k+1} \in \arg\min_{\Theta} \; F(\mathcal{A}_\Theta, \mathcal{E}^k; \mathcal{M}) + \lambda\phi(\mathcal{A}_\Theta) + \frac{\eta}{2}\|\mathcal{A}_\Theta - \mathcal{A}_{\Theta^k}\|_F^2, \tag{4.4}$$

$$\mathcal{E}^{k+1} \in \arg\min_{\mathcal{E}} \; \gamma\psi(\mathcal{E}) + F(\mathcal{A}_{\Theta^{k+1}}, \mathcal{E}; \mathcal{M}) + \frac{\eta}{2}\|\mathcal{E} - \mathcal{E}^k\|_F^2, \tag{4.5}$$

where $\eta > 0$ is a proximal parameter. It is worth noting that we add $\frac{\eta}{2}\|\mathcal{A}_\Theta - \mathcal{A}_{\Theta^k}\|_F^2$ rather than $\frac{\eta}{2}\|\Theta - \Theta^k\|_F^2$ to the Θ subproblem. The pseudocode of PAO algorithm for solving the model (4.2) is summarized in Algorithm 4.1.

Algorithm 4.1 PAO algorithm for solving the model (4.2)

Input: $\mathcal{M}$, λ, γ, η.
Initialize: Θ^0, $\mathcal{E}^0$. Set $k = 0$.
Step1: Compute Θ^{k+1} by solving the subproblem (4.4);
Step2: Compute $\mathcal{E}^{k+1}$ by solving the subproblem (4.5);
Step3: If a termination criterion is not met, set $k = k + 1$ and return to step 1;
Output: Θ^{k+1}, $\mathcal{A}_{\Theta^{k+1}}$.

Next, we provide a convergence analysis for Algorithm 4.1. Since the objective function in MITD-based models may lack the Kurdyka–Łojasiewicz (KL) property[2, 3] due to complex landscapes induced by deep architectures and nonlinear activations, we adopt a KL-free analysis framework to ensure broader applicability. Define

$$\widehat{G}(\mathcal{A}, \mathcal{E}) := F(\mathcal{A}, \mathcal{E}; \mathcal{M}) + \lambda\phi(\mathcal{A}) + \gamma\psi(\mathcal{E}).$$

Then $G(\Theta, \mathcal{E})$ is the composite function of $\widehat{G}(\mathcal{A}, \mathcal{E})$ and $\mathcal{A} = \mathcal{A}_\mathbf{x}(\Theta)$.

THEOREM 4.1. *Assume the hypotheses of Theorem 3.7 hold. Then, any limit point $(\Theta^*, \mathcal{E}^*)$ of the sequence $\{(\Theta^k, \mathcal{E}^k)\}$ generated by Algorithm 4.1 is a critical point of G. If the map $\Theta \mapsto \mathcal{A}_\Theta$ is a submersion at Θ^*, i.e., the Jacobian has full row rank, then $(\mathcal{A}_{\Theta^*}, \mathcal{E}^*)$ is also a critical point of $\widehat{G}$, and $\{(\mathcal{A}_{\Theta^k}, \mathcal{E}^k)\}$ cannot oscillate between distinct critical points of $\widehat{G}$. Moreover, if all critical points of $\widehat{G}$ are isolated, the sequence $\{(\mathcal{A}_{\Theta^k}, \mathcal{E}^k)\}$ converges globally to a single critical point of $\widehat{G}$.*

Proof. Since Θ^{k+1} and $\mathcal{E}^{k+1}$ are optimal solutions of (4.4) and (4.5), we have

$$\begin{aligned} G(\mathcal{A}_{\Theta^{k+1}}, \mathcal{E}^k) &\le G(\mathcal{A}_{\Theta^k}, \mathcal{E}^k) - \frac{\eta}{2}\|\mathcal{A}_{\Theta^{k+1}} - \mathcal{A}_{\Theta^k}\|_F^2, \\ G(\mathcal{A}_{\Theta^{k+1}}, \mathcal{E}^{k+1}) &\le G(\mathcal{A}_{\Theta^{k+1}}, \mathcal{E}^k) - \frac{\eta}{2}\|\mathcal{E}^{k+1} - \mathcal{E}^k\|_F^2. \end{aligned} \tag{4.6}$$

For convenience, we denote $G^k = G(\mathcal{A}_{\Theta^k}, \mathcal{E}^k)$. Then, from (4.6), one has

$$G^{k+1} \le G^k - \frac{\eta}{2}(\|\mathcal{A}_{\Theta^{k+1}} - \mathcal{A}_{\Theta^k}\|_F^2 + \|\mathcal{E}^{k+1} - \mathcal{E}^k\|_F^2). \tag{4.7}$$

Define the Lyapunov function: $V_k := G^k + \frac{\eta}{2}a_k + \frac{\eta}{2}e_k$, where $a_k = \|\mathcal{A}_{\Theta^k} - \mathcal{A}_{\Theta^{k-1}}\|_F^2$, $e_k = \|\mathcal{E}^k - \mathcal{E}^{k-1}\|_F^2$. Then, by combining with (4.7), we analyze the difference:

$$V_{k+1} - V_k = G^{k+1} - G^k + \frac{\eta(a_{k+1} - a_k + e_{k+1} - e_k))}{2} \le -\frac{\eta(a_k + e_k)}{2} < 0. \tag{4.8}$$

This implies that V_k is monotonically decreasing. Therefore, V_k converges to some finite limit V^* since it is bounded below. By summing (4.8) over k, we obtain $\sum_{k=1}^{\infty}(a_k+e_k) \leq \frac{2}{\eta}(V_1-V^*) < \infty$, which implies $a_k+e_k \to 0$, i.e., $\|\mathcal{A}_{\Theta^{k+1}}-\mathcal{A}_{\Theta^k}\|_F \to 0$, $\|\mathcal{E}^{k+1}-\mathcal{E}^k\|_F \to 0$. From the definition of V^k, we have $G^k \to V^*$. According to the assumption of Theorem 3.7, the network parameter Θ is bounded. Thus, the sequence $\{(\Theta^k, \mathcal{E}^k)\}$ generated by Algorithm 4.1 is also bounded since G is coercive in $\mathcal{E}$. Therefore, there exists a subsequence $\{(\Theta^t, \mathcal{E}^t)\}$ that converges to a limit point, denoted as $(\Theta^*, \mathcal{E}^*)$. From (4.4) and (4.5), one has:

$$\begin{aligned} 0 &\in [\nabla_{\mathcal{A}_\Theta} F(\mathcal{A}_{\Theta^{t+1}}, \mathcal{E}^t) + \lambda \partial\phi(\mathcal{A}_{\Theta^{t+1}}) + \eta \mathcal{R}^t]^\top \mathcal{J}_{t+1}, \\ 0 &\in \nabla_{\mathcal{E}} F(\mathcal{A}_{\Theta^{t+1}}, \mathcal{E}^{t+1}) + \gamma \partial\psi(\mathcal{E}^{t+1}) + \eta(\mathcal{E}^{t+1} - \mathcal{E}^t), \end{aligned} \tag{4.9}$$

where $\mathcal{J}_{t+1} := \frac{\partial \mathcal{A}_\Theta}{\partial \Theta}\big|_{\Theta=\Theta^{t+1}}$, $\mathcal{R}^t := \mathcal{A}_{\Theta^{t+1}} - \mathcal{A}_{\Theta^t}$. The derivative computation is performed in the convention of vectorizing the tensor variables. Then we obtain:

$$\begin{aligned} [\nabla_{\mathcal{A}_\Theta} F(\mathcal{A}_{\Theta^{t+1}}, \mathcal{E}^{t+1}) - \nabla_{\mathcal{A}_\Theta} F(\mathcal{A}_{\Theta^{t+1}}, \mathcal{E}^t) + \eta \mathcal{R}^t]^\top \mathcal{J}_{t+1} &\in \partial G_\Theta(\mathcal{A}_{\Theta^{t+1}}, \mathcal{E}^{t+1}), \\ \eta(\mathcal{E}^t - \mathcal{E}^{t+1}) &\in \partial G_{\mathcal{E}}(\mathcal{X}_f^{t+1}, \mathcal{E}^{t+1}). \end{aligned}$$

Taking $k \to +\infty$ on both sides, we obtain $0 \in \partial G_\Theta(\Theta^*, \mathcal{E}^*)$, and $0 \in \partial G_{\mathcal{E}}(\Theta^*, \mathcal{E}^*)$, which implies that $(\Theta^*, \mathcal{E}^*)$ is a critical point of G. Since $\mathcal{A}_\Theta$ is a submersion at Θ^*, i.e., $\mathcal{J}_* := \frac{\partial \mathcal{A}_\Theta}{\partial \Theta}\big|_{\Theta=\Theta^*}$ is full row rank, we can also obtain $0 \in \partial \widehat{G}_{\mathcal{A}}(\mathcal{A}_{\Theta^*}, \mathcal{E}^*)$ and $0 \in \partial \widehat{G}_{\mathcal{E}}(\mathcal{A}_{\Theta^*}, \mathcal{E}^*)$. Therefore, $(\mathcal{A}_{\Theta^*}, \mathcal{E}^*)$ is also a critical point of $\widehat{G}$.

Next, we prove that the sequence $\{(\mathcal{A}_{\Theta^k}, \mathcal{E}^k)\}$ does not oscillate among limit points. By contradiction, if it did, then $a_k + e_k$ would remain greater than some constant C indefinitely. From (4.8), we know that in each iteration, V_{k+1} decreases by at least $C\eta/2$, which contradicts the fact that V_k is bounded below. If critical points are all isolated, the sequence eventually enters and stays in a neighborhood of one such point, i.e., it converges globally to that critical point. ∎

Remark 4.1. The parameter sequence $\{\Theta^k\}$ may fail to converge due to the non-uniqueness of network parameters, i.e., Multiple Θ can yield the same $\mathcal{A}_\Theta$. However, our primary interest lies in the convergence of the output $\{\mathcal{A}_{\Theta^k}\}$. Even if Θ^k oscillates among equivalent parameterizations, the corresponding outputs remain stable, which is sufficient for practical performance and structural recovery.

5. Experiments. In this section, we conducted extensive numerical experiments to verify the effectiveness of the proposed IMTD, including robust tensor completion (grid-structured data) and point cloud upsampling (non-grid data). The IMTD is trained on a NVIDIA GeForce RTX 4060 GPU, and we run the compared codes in MATLAB R2023a on a 12th Gen Intel® Core™ i5-12450H processor at 2.00 GHz.

5.1. Parameter settings. The crucial parameters are the Multiple rank in IMTD. Although the Multiple rank of a tensor $\mathcal{X} \in \Re^{I_1 \times I_2 \times \cdots \times I_N}$ cannot be determined directly, a reasonable range can be estimated. According to Theorem 3.2, we recommend first roughly determining GTriRank($\mathcal{X}$), and then gradually adjust it downward to select the Multiple rank. Specifically, we propose the following lower and upper bounds for estimating GTriRank($\mathcal{X}$):

$$r_{min} := \max\{I_1^{\frac{1}{N-1}}, I_2^{\frac{1}{N-1}}, ..., I_N^{\frac{1}{N-1}}\}/3, \quad r_{max} := \frac{2}{3}\sqrt[N-1]{\frac{I_1 I_2 ... I_N}{I_1 + I_2 + ... + I_N}}.$$

For a point cloud dataset $\mathcal{X}$ consisting of N points, i.e., $\mathcal{X} \in \Re^{N \times 3}$, we recommend $r = N^{\frac{1}{3}}$ as an upper bound for the choice of MtpRank($\mathcal{X}$).

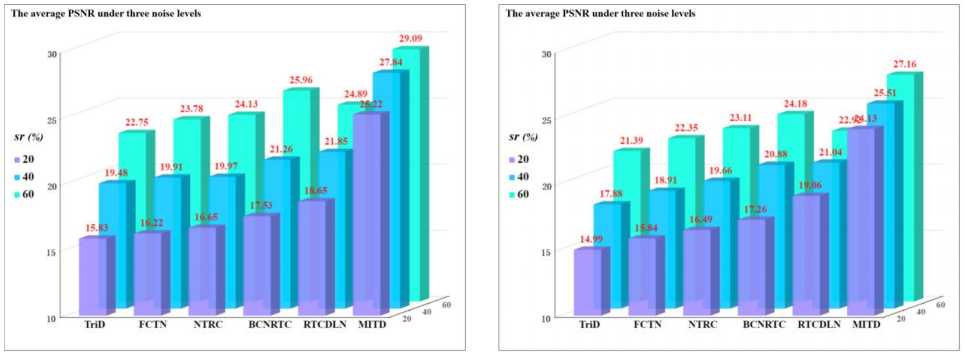
5.2. Comparison on robust tensor completion (RTC) task. In this subsection, we compare IMTD with several RTC methods, including: TriD [25], FCTN [43], NTRC [28], BCNRTC [41] and RTCDLN [27]. All these methods are evaluated on both color images and color videos. Specifically, the experiments are conducted on two color images: Pepper ($512 \times 512 \times 3$) and Flower ($321 \times 481 \times 3$), along with two color videos: Ocean ($112 \times 160 \times 3 \times 32$) and Man ($144 \times 176 \times 3 \times 51$). All images and videos are corrupted by (i) randomly removing a fraction of entries (i.e., missing data) and (ii) adding salt-and-pepper noise to the observed entries. We set the sampling rates (sr) to $0.2, 0.4$, and 0.6, and consider noise levels of $\sigma \in \{0.2, 0.4, 0.6\}$.

5.2.1. Color images. Table 1 presents the reconstruction performance of various methods for the color images. As shown in the table, the IMTD-based RTC

Table 1: PSNR (dB) values for restoring results of different methods for colour images corrupted by sample loss and salt-and-pepper noise.

Sample rates (%)	Noise level (%)	Pepper					
		TriD	FCTN	NTRC	BCNRTC	RTCDLN	IMTD
20	20	18.86	19.62	19.76	21.17	22.86	**28.65**
	40	16.03	16.36	16.95	17.49	18.83	**25.13**
	60	12.62	12.89	13.24	13.92	14.26	**21.89**
40	20	23.58	24.27	23.98	25.96	26.58	**31.27**
	40	19.24	20.16	19.70	21.74	22.17	**28.08**
	60	15.63	15.87	16.24	16.09	16.81	**24.17**
60	20	27.75	28.83	29.36	30.92	29.85	**32.86**
	40	22.47	23.72	23.90	26.16	25.97	**29.62**
	60	18.05	18.79	19.14	20.79	18.85	**24.79**
Sample rates (%)	Noise level (%)	Flower					
		TriD	FCTN	NTRC	BCNRTC	RTCDLN	IMTD
20	20	19.15	20.71	21.35	22.07	22.60	**26.75**
	40	15.20	15.45	16.71	17.23	19.82	**24.67**
	60	10.63	11.37	11.42	12.48	14.76	**20.98**
40	20	22.49	23.89	24.64	26.02	25.54	**28.91**
	40	17.71	18.74	19.51	21.27	21.66	**25.48**
	60	13.46	14.10	14.84	15.35	15.94	**22.14**
60	20	25.01	26.95	27.55	29.07	28.12	**30.42**
	40	21.90	22.43	23.42	24.17	23.20	**27.38**
	60	17.15	17.80	18.37	19.31	17.45	**23.69**

method consistently outperforms all competing methods across all test settings. Notably, its performance advantage becomes even more significant under conditions of low sampling rates and high noise levels. Taking the 'Pepper' dataset as an example, when the sampling rate is 0.2 and the noise level is 0.6, the PSNR of the image reconstructed by IMTD is 7.6 dB higher than that achieved by the second-best method, RTCDLN. This is mainly attributed to the robustness of the Multiple decomposition structure and the strong representation capability of the implicit neural network for the data. Moreover, TriD, FCTN and NTRC achieve very similar performance on both datasets, primarily because these methods exhibit comparable expressiveness for third-order data. TriD further restricts the short sides of the factor tensors to be identical, reducing flexibility and resulting in performance inferior to other methods. As the sampling rate increases and the noise level decreases, the observed data become more informative, allowing traditional low-rank RTC models to better recover the underlying structure. Consequently, their performance improves significantly. Both



(a) Pepper (b) Flower

Fig. 4: The average PSNR under three noise levels at different sampling rates.

BCNRTC and RTCDLN are based on the tensor nuclear norm (TNN): the former enhances TNN through a non-convex correction, while the latter introduces a transform domain. Their superior performance demonstrates the effectiveness of TNN-based RTC methods. However, their performance still lags behind that of IMTD, particularly under strong interference conditions. The average PSNR under three noise levels are displayed in Figure 4, from which we can more intuitively perceive the advantages of the IMTD-based RTC method. Figure 5 shows the recovered images, where the region marked by blue box is magnified and displayed in the upper left corner.

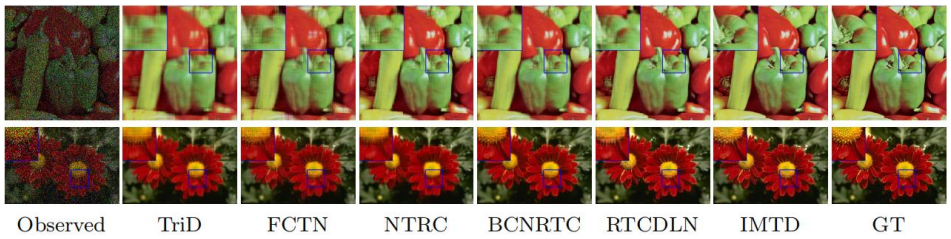
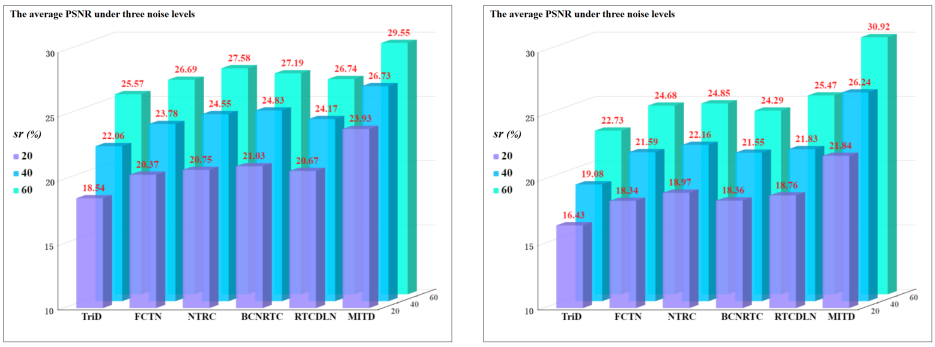


Fig. 5: The visual comparison of restoration results on the 'Pepper' (sr=0.4, σ=0.2) and 'Flower' (sr=0.6, σ=0.2) datasets for each method.

As shown in the figure, the IMTD-based method recovers corrupted color images more effectively than the other compared approaches, producing visually clearer results. For example, in the blue-marked region of the 'pepper' image that contains relatively complex textures, the low-rank RTC methods tend to oversmooth these intricate details when filling in missing values and suppressing noise. This results in distortions in the complex regions of the image. In contrast, the IMTD-based approach can better preserve and recover fine image details by implicitly learning the underlying functional distribution of the data. For the 'Flower' dataset, BCNRTC also demonstrates strong reconstruction performance in localized regions. This is primarily due to the non-convex surrogate of the TNN and ℓ_1 norm, which allows for better preservation of the image's high-frequency details. However, achieving such performance typically requires careful and intricate parameter tuning.

5.2.2. Color videos. A color video can be represented as a fourth-order tensor of dimensions $W \times H \times 3 \times S$, where W and H denote the spatial width and height of each frame, 3 corresponds to the three color channels (e.g., RGB), and S represents the total number of frames in the video. For methods that can only handle third-



(a) Ocean (b) Man

Fig. 6: The average PSNR under three noise levels at different sampling rates.

order tensors (TriD, BCNRTC, and RTCDLN), we reshape the data into a third-order tensor with size $W \times H \times (3S)$, where 3S combines the color channels across all frames. Figure 6 presents the average PSNR values of the reconstructed images across three noise levels in various sampling rates.

From the figure, the advantages of BCNRTC and RTCDLN become less pronounced and even inferior to those of FCTN and NTRC. This is primarily attributed to the merging of the fourth-order tensor into a third-order tensor, which disrupts the intrinsic correlations within the data, thereby diminishing the effectiveness of TNN-based RTC methods in restoring color video data. Moreover, NTRC performs better than FCTN on color video data. A possible reason is that although FCTN can establish direct connections among arbitrary factors, it inevitably introduces excessive complexity into the process, leading to degraded performance. Overall, the IMTD-based RTC method consistently achieves significant performance advantages. On one hand, it can directly handle high-order tensors with flexible rank adaptability. On the other hand, its implicit learning-based functional representation enables more effective capture and recovery of fine details in video and image data.

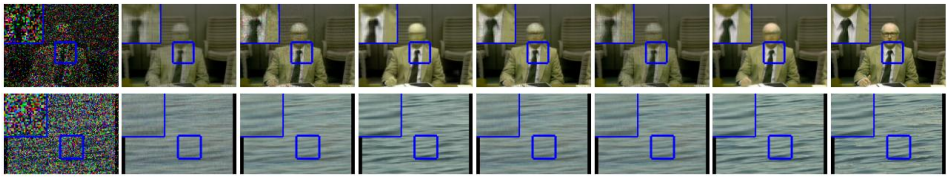


Fig. 7: The visual comparison of restoration results on the 'Man' dataset(sr=0.4, σ=0.2) and 'Ocean' dataset(sr=0.6, σ=0.2) for each compared method.

Figure 7 shows the performance of compared methods on video datasets. The reconstructed video of TriD exhibits obvious stripes and noise, primarily since it is only applicable to third-order tensors and lacks flexibility in rank selection. Both NTRC and FCTN achieved competitive performance compared to BCNRTC and RTCDLN, which indicates that the reconstruction performance of TNN-based approaches for fourth-order tensors is degraded. In contrast, our method still achieves optimal performance on video datasets. Furthermore, IMTD significantly outperforms TriD, which clearly demonstrates the advantages of our IMTD in terms of both modeling capability and computational performance.

5.3. Comparison on point cloud upsampling (PCU) task. Next, we evaluate our method on the PCU task to demonstrate its effectiveness on irregular, non-grid data. Several benchmark datasets are selected, including star, balls and heart [39]. We randomly sampled 5% or 10% of the points as input and applied the compared methods to upsample them. Then, We compare IMTD with Snowflake [34], SAPCU [39], and LRTFR [21], using Chamfer Distance (CD) [36] and F-Score [31] as evaluation metrics. Table 2 records the corresponding quantitative results.

Table 2: The CD and F-score for three datasets upsampled by compared methods.

Dataset	Star		Ball		Heart	
Methods	CD	F-Score	CD	F-Score	CD	F-Score
Snowflake	0.1274	0.8496	0.0746	0.9351	0.0839	0.7862
SAPCU	0.0789	0.9033	0.0527	0.9649	0.0775	0.9136
LRTFR	0.0647	0.9368	0.0442	0.9840	0.0627	0.9615
IMTD	**0.0581**	**0.9642**	**0.0435**	**0.9866**	**0.0542**	**0.9767**

As shown in Table 2, the proposed IMTD consistently achieves superior performance across all three datasets. This implies that the point clouds predicted by IMTD are closer to the ground truth and exhibit better local structure and completeness. This can be primarily attributed to IMTD's architectural flexibility and its capability of exploiting multi-dimensional data features. The corresponding visual results are presented in Figure 8. From the figure, Snowflake's upsampling introduces additional

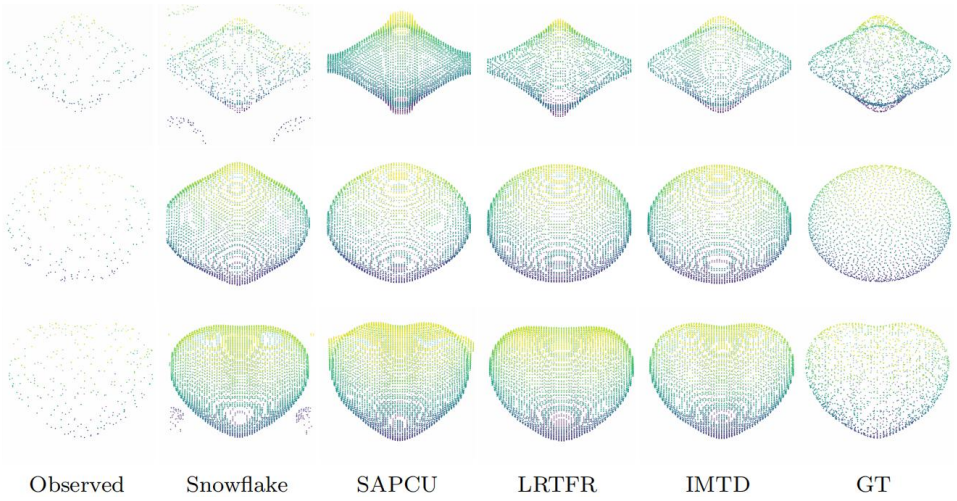
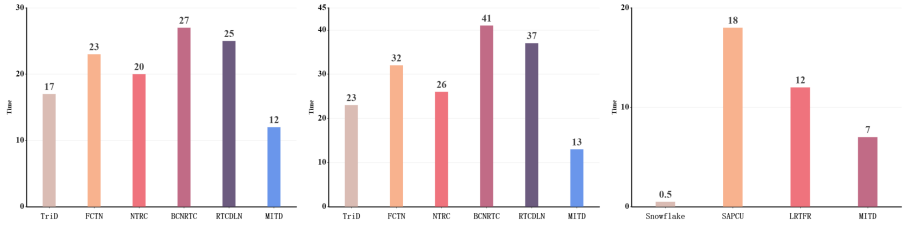


Fig. 8: The visual comparison of compared methods on point cloud upsampling. From top to bottom: Star ($sr = 0.1$), Balls ($sr = 0.1$) and Heart ($sr = 0.05$).

erroneous points on both 'Star' and 'Heart' datasets, which may be attributed to the use of deconvolution. Moreover, its reconstruction of the 'ball' dataset exhibits noticeable geometric distortions. This may be attributed to limited generalization, as Snowflake is trained on specific shape categories. SAPCU, LRTFR, and IMTD are all based on implicit neural representations. SAPCU does not exploit the intrinsic low-rank structure of the data, resulting in noticeable deviations in its upsampling results. LRTFR adopts Tucker decomposition and parameterizes the factor matrices, achieving competitive performance. In comparison, IMTD achieves greater preservation in overall shape and fine-grained geometric details. This advantage stems not

only from IMTD's architectural flexibility, but also from the parameterization of all factor tensors, which enables the model to capture more refined geometric features.

5.4. Running time. Next, we compare the execution of each method, as shown in Figure 9. For the RTC task, IMTD exhibits increased computational efficiency than the tensor decomposition-based approaches owing to the utilization of GPU acceleration. The TNN-based methods incur higher computational cost, due to the expensive and repeated SVD calculations during the optimization process. In the PCU task, all the compared methods leverage GPU acceleration during training. Snowflake adopts a supervised learning framework, leading to lengthy training times but enables fast inference. SAPCU, LRTFR, and IMTD are unsupervised approaches. SAPCU directly represents point clouds using implicit neural networks, resulting in higher computational cost. LRTFR reduces dimensionality through Tucker decomposition but suffers from expensive core tensor updates. Overall, our approach demonstrates superior efficiency in both tasks.



(a) RTC (Pepper) (b) RTC (Ocean) (c) PDU (Heart)

Fig. 9: The running time of the compared methods for different tasks.

6. Conclusion. In this paper, we propose the Implicit Multiple Tensor Decomposition (IMTD), a novel framework that generalizes triple decomposition to tensors of arbitrary order. In addition, it allows the short dimensions of the factor tensors to differ. By representing the factor tensors as implicit functions, IMTD enables continuous modeling of tensor data and supports decomposition of irregular and non-grid data. Theoretically, we investigate the connections between IMTD and classical tensor decompositions, and develop a computational algorithm for addressing the reconstruction problem within the IMTD framework. A KL-free convergence analysis is also provided. Finally, extensive numerical experiments further validate the effectiveness and broad applicability of the proposed method.

Acknowledgments. We would like to acknowledge the assistance of volunteers in putting together this example manuscript and supplement.

REFERENCES

[1] D. Aksoy, D. J. Gorsich, S. Veerapaneni, and A. A. Gorodetsky, *An incremental tensor train decomposition algorithm*, SIAM J. Sci. Comput., 46 (2024), pp. A1047–A1075, https://doi.org/10.1137/22m1537734.

[2] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran, *Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-łojasiewicz inequality*, Math. Oper. Res., 35 (2010), pp. 438–457, https://doi.org/10.1287/moor.1100.0449.

[3] H. Attouch, J. Bolte, and B. F. Svaiter, *Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized*

gauss–seidel methods, Math. Program., 137 (2011), pp. 91–129, https://doi.org/10.1007/s10107-011-0484-9.

[4] M. Bai, X. Zhang, and Q. Shao, *Adaptive correction procedure for tvl1 image deblurring under impulse noise*, Inverse Problems, 32 (2016), p. 085004, https://doi.org/10.1088/0266-5611/32/8/085004.

[5] J. A. Bengua, H. N. Phien, H. D. Tuan, and M. N. Do, *Efficient tensor completion for color image and video recovery: Low-rank tensor train*, IEEE Trans. Image Process., 26 (2017), pp. 2466–2479, https://doi.org/10.1109/tip.2017.2672439.

[6] L. Bungert, D. A. Coomes, M. J. Ehrhardt, J. Rasch, R. Reisenhofer, and C.-B. Schönlieb, *Blind image fusion for hyperspectral imaging with the directional total variation*, Inverse Problems, 34 (2018), p. 044003, https://doi.org/10.1088/1361-6420/aaaf63.

[7] Y. Chen, T.-Z. Huang, W. He, X.-L. Zhao, H. Zhang, and J. Zeng, *Hyperspectral image denoising using factor group sparsity-regularized nonconvex low-rank approximation*, IEEE Trans. Geosc. Remote Sens., 60 (2022), pp. 1–16, https://doi.org/10.1109/tgrs.2021.3110769.

[8] G. T. de Araujo, A. L. F. de Almeida, and R. Boyer, *Channel estimation for intelligent reflecting surface assisted mimo systems: A tensor modeling approach*, IEEE J. Sel. Top. Signal Process., 15 (2021), pp. 789–802, https://doi.org/10.1109/jstsp.2021.3061274.

[9] V. de Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127, https://doi.org/10.1137/06066518x.

[10] L. Feng, C. Zhu, Z. Long, J. Liu, and Y. Liu, *Multiplex transformed tensor decomposition for multidimensional image recovery*, IEEE Trans. Image Process., 32 (2023), pp. 3397–3412, https://doi.org/10.1109/tip.2023.3284673.

[11] T. Imbiriba, R. A. Borsoi, and J. C. M. Bermudez, *Low-rank tensor modeling for hyperspectral unmixing accounting for spectral variability*, IEEE Trans. Geosci. Remote Sens., 58 (2020), pp. 1833–1842, https://doi.org/10.1109/tgrs.2019.2949543.

[12] D. Jin, J. Liu, J. Yang, and Z. Wu, *High-order coupled fully connected tensor network decomposition for hyperspectral image super-resolution*, IEEE Geosci. Remote Sens. Let., 19 (2022), pp. 1–5, https://doi.org/10.1109/lgrs.2022.3207548.

[13] M. E. Kilmer and C. D. Martin, *Factorization strategies for third-order tensors*, Linear Algebra Appli., 435 (2011), pp. 641–658, https://doi.org/10.1016/j.laa.2010.09.020.

[14] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv:1412.6980, (2014), https://arxiv.org/abs/1412.6980.

[15] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500, https://doi.org/10.1137/07070111x.

[16] B.-Z. Li, X.-L. Zhao, T.-Y. Ji, X.-J. Zhang, and T.-Z. Huang, *Nonlinear transform induced tensor nuclear norm for tensor completion*, J. Sci. Comput., 92 (2022), https://doi.org/10.1007/s10915-022-01937-1.

[17] S. Li, R. Dian, L. Fang, and J. M. Bioucas-Dias, *Fusing hyperspectral and multispectral images via coupled sparse tensor factorization*, IEEE Trans. Image Process., 27 (2018), pp. 4118–4130, https://doi.org/10.1109/tip.2018.2836307.

[18] Y. Lin, S. Jin, M. Matthaiou, and X. You, *Tensor-based channel estimation for millimeter wave mimo-ofdm with dual-wideband effects*, IEEE Trans. Commun., 68 (2020), pp. 4218–4232, https://doi.org/10.1109/tcomm.2020.2983673.

[19] J. Liu, P. Musialski, P. Wonka, and J. Ye, *Tensor completion for estimating missing values in visual data*, IEEE Trans. Pattern Anal. Mach. Intell., 35 (2013), pp. 208–220, https://doi.org/10.1109/tpami.2012.39.

[20] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, *Tensor robust principal component analysis with a new tensor nuclear norm*, IEEE Trans. Pattern Anal. Mach. Intell., 42 (2020), pp. 925–938, https://doi.org/10.1109/tpami.2019.2891760.

[21] Y. Luo, X. Zhao, Z. Li, M. K. Ng, and D. Meng, *Low-rank tensor function representation for multi-dimensional data recovery*, IEEE Trans. Pattern Anal. Mach. Intell., 46 (2024), pp. 3351–3369, https://doi.org/10.1109/tpami.2023.3341688.

[22] Z. Ming, Z. Qin, L. Zhang, Y. Xu, and L. Qi, *Network traffic recovery from link-load measurements using tensor triple decomposition strategy for third-order traffic tensors*, J. Comput. Appl. Math., 447 (2024), p. 115901, https://doi.org/10.1016/j.cam.2024.115901.

[23] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317, https://doi.org/10.1137/090752286.

[24] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, *Deepsdf: Learning continuous signed distance functions for shape representation*, (2019), https://doi.org/10.48550/ARXIV.1901.05103, https://arxiv.org/abs/1901.05103.

[25] L. Qi, Y. Chen, M. Bakshi, and X. Zhang, *Triple decomposition and tensor recovery of third order tensors*, SIAM J. Matrix Anal Appl., 42 (2021), pp. 299–329, https://doi.org/10.1137/20m1323266.

[26] D. Qiu, M. Bai, M. K. Ng, and X. Zhang, *Robust low transformed multi-rank tensor methods for image alignment*, J. Sci. Comput., 87 (2021), https://doi.org/10.1007/s10915-021-01437-8.

[27] D. Qiu, B. Yang, and X. Zhang, *Robust tensor completion via dictionary learning and generalized nonconvex regularization for visual data recovery*, IEEE Trans. Circ, Syst, Video Tech,, 34 (2024), pp. 11026–11039, https://doi.org/10.1109/tcsvt.2024.3413992.

[28] Y. Qiu, G. Zhou, Q. Zhao, and S. Xie, *Noisy tensor completion via low-rank tensor ring*, IEEE Trans. Neural Net. Learn. Syst., 35 (2024), pp. 1127–1141, https://doi.org/10.1109/tnnls.2022.3181378.

[29] V. Saragadam, R. Balestriero, A. Veeraraghavan, and R. G. Baraniuk, *Deeptensor: Low-rank tensor decomposition with deep network priors*, IEEE Trans. Pattern Anal. Mach. Intell., 46 (2024), pp. 10337–10348, https://doi.org/10.1109/tpami.2024.3450575.

[30] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, *Implicit neural representations with periodic activation functions*, licit neural representations with periodic activationfunctions," in Advances in Neural Information Processing Systems(NeurIPS), 33 (2020), pp. 7462–7473.

[31] A. A. Taha and A. Hanbury, *Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool*, BMC Medical Imaging, 15 (2015), https://doi.org/10.1186/s12880-015-0068-x.

[32] C. Wang, X.-L. Zhao, Y.-B. Zheng, B.-Z. Li, and M. K. Ng, *Functional tensor singular value decomposition*, SIAM J. Sci. Comput., 47 (2025), pp. A2180–A2204, https://doi.org/10.1137/24m1644687.

[33] Z. Wang and M. Bai, *An inexactly accelerated algorithm for nonnegative tensor cp decomposition with the column unit constraints*, Comput. Opt. Appl., 88 (2024), pp. 923–962, https://doi.org/10.1007/s10589-024-00574-8.

[34] P. Xiang, X. Wen, Y.-S. Liu, Y.-P. Cao, P. Wan, W. Zheng, and Z. Han, *Snowflake point deconvolution for point cloud completion and generation with skip-transformer*, IEEE Trans. Pattern Anal. Mach. Intell., (2022), pp. 1–18, https://doi.org/10.1109/tpami.2022.3217161.

[35] K. Yang, M. Bai, R. Dian, and T. Lu, *Subspace-based coupled tensor decomposition for hyperspectral blind fusion*, Inverse Problems and Imaging, 19 (2025), pp. 560–591, https://doi.org/10.3934/ipi.2024045.

[36] Y. Yu, J. Choi, Y. Kim, K. Yoo, S.-H. Lee, and G. Kim, *Supervising neural attention models for video captioning by human gaze data*, in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, July 2017, https://doi.org/10.1109/cvpr.2017.648.

[37] X. Zhang, D. Wang, Z. Zhou, and Y. Ma, *Robust low-rank tensor recovery with rectification and alignment*, IEEE Trans. Pattern Anal. Mach. Intell., 43 (2021), pp. 238–255, https://doi.org/10.1109/tpami.2019.2929043.

[38] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, *Tensor ring decomposition*, (2016), https://doi.org/10.48550/ARXIV.1606.05535, https://arxiv.org/abs/1606.05535.

[39] W. Zhao, X. Liu, D. Zhai, J. Jiang, and X. Ji, *Self-supervised arbitrary-scale implicit point clouds upsampling*, IEEE Trans. Pattern Anal. Mach.Intell., 45 (2023), pp. 12394–12407, https://doi.org/10.1109/tpami.2023.3287628.

[40] X. Zhao, M. Bai, and M. K. Ng, *Nonconvex optimization for robust tensor completion from grossly sparse observations*, J. Sci. Comput., 85 (2020), https://doi.org/10.1007/s10915-020-01356-0.

[41] X. Zhao, M. Bai, D. Sun, and L. Zheng, *Robust tensor completion: Equivalent surrogates, error bounds, and algorithms*, SIAM J. Imaging Sci., 15 (2022), pp. 625–669, https://doi.org/10.1137/21m1429539.

[42] L. Zheng, Z. Wang, M. Bai, Z. Tan, and Q. Zhu, *Beamforming tensor compression for massive mimo fronthaul*, IEEE Trans. Commun., 73 (2025), pp. 3894–3908, https://doi.org/10.1109/tcomm.2024.3496749.

[43] Y.-B. Zheng, T.-Z. Huang, X.-L. Zhao, Q. Zhao, and T.-X. Jiang, *Fully-connected tensor network decomposition and its application to higher-order tensor completion*, AAAI Conference on Artificial Intelligence, 35 (2021), pp. 11071–11078, https://doi.org/10.1609/aaai.v35i12.17321.

[44] D. Zhu, H. Chen, W. Wang, H. Xie, G. Cheng, M. Wei, J. Wang, and F. L. Wang, *Nonlocal low-rank point cloud denoising for 3-d measurement surfaces*, IEEE Trans. Instrum, Meas., 71 (2022), pp. 1–14, https://doi.org/10.1109/tim.2021.3139686.