Tensor Decomposition Learning for Compression of Multidimensional Signals

Anastasia Aidini, Grigorios Tsagkatakis, and Panagiotis Tsakalides

***Abstract*—Multidimensional signals like multispectral images and color videos are becoming ubiquitous in modern times, constantly introducing challenges in data storage and transfer, and therefore demanding efficient compression strategies. Such high dimensional observations can be naturally encoded as tensors, exhibiting significant redundancies across dimensions. This property is exploited by tensor decomposition techniques that are being increasingly used for compactly encoding large multidimensional arrays. While efficient, these methods are incapable of utilizing prior information present in training data. In this paper, a novel tensor decomposition learning method is proposed for the compression of high dimensional signals. Specifically, instead of extracting independent bases for each example, our method learns an appropriate basis for each dimension from a set of training samples by solving a constrained optimization problem. As such, each sample is quantized and encoded into a reduced-size core tensor of coefficients that corresponds to the multilinear combination of the learned basis matrices. Furthermore, the proposed method employs a symbol encoding dictionary for binarizing the decomposition outputs. Experimental results on synthetic data and real satellite multispectral image sequences demonstrate the efficacy of our method, surpassing competing compression methods while offering the flexibility to handle arbitrary high dimensional data structures.**

***Index Terms*—Compression, multidimensional signals, Tucker decomposition, ADMM, learning**

I. Introduction

The continuous and excessive data growth that dominates the Big Data era leads to the generation of significant quantities of multidimensional observations. A natural way to represent such high dimensional signals is higher-order extensions of vectors and matrices, known as tensors, with entries indexed by several variables [1], [2]. A wide range of real-world data takes the format of tensors such as multispectral image and video sequences, as well as measurements from multiple sensors with different sensing modalities [3].

These massive multidimensional data introduce considerable challenges in terms of storage and communication, therefore effective data reduction, compact data representation, and compression techniques are of paramount importance. The need for compression becomes more apparent when one counts the number of bits needed to represent the information of a signal. The amount, though, of bits actually required to describe it, can be much smaller due to the existence of information redundancy [4].

Tensor-based approaches have recently been shown to be a powerful tool in compression of high dimensional data [5]–[10], as well as in many other signal processing applications, including multi-view clustering [11] and nonlocal image denoising [12], since they can retain the structure of the data. In contrary to the numerous transform-based data reduction methods [13], compression algorithms that involve tensor models can naturally exploit the correlations among variables.

Specifically, the Tucker decomposition [14] is an increasingly popular tensor-based technique for dimensionality reduction, which approximates a tensor data set by a multilinear combination of factor matrices weighted by the coefficients of a reduced-size core tensor. The factor matrices are crucial components of the decomposition since their column vectors represent the set of basis functions onto which the data is projected, and thus define the mapping between initial and compressed data and vice versa. While in transform coding methods, the bases are pre-defined and independent of the input data [15], tensor decompositions rely on data-dependent bases extracted directly from the input data itself. What is currently lacking, is a formal approach to use pre-defined, yet data-specific bases, for the representations. Following a machine learning paradigm, these bases should be extracted from training data that exhibit similar characteristics compared to the one under consideration (the testing data).

In this paper, we formulate the problem of tensor decomposition in a machine learning context, where training examples are used in order to populate the appropriate representation subspaces. This approach is in stark contrast to existing decomposition approaches, including both traditional Tucker and CANDECOMP/PARAFAC (CP) [3], where each example is decomposed independently from any others. Therefore, instead of extracting an independent representation for each example, we introduce a formal tensor decomposition learning method to learn a generic basis for each dimension that can efficiently represent every sample, by solving a constrained optimization problem using the Alternating Direction Method of Multipliers (ADMM) [16]. Specifically, we formulate the problem using a low-rank constraint in order to extract the correlations in the training data, and Tucker decomposing that can encode a larger set of interactions compared to CP decomposition. Given the learned decomposition matrices, each new sample is represented as a multilinear combination of them with coefficients in a reduced-sized core tensor.

As a specific application of the proposed tensor decompo-

A. Aidini is with the Department of Computer Science, University of Crete, Greece and the Institute of Computer Science - FORTH, Greece. Email: aidini@ics.forth.gr

G. Tsagkatakis is with the Institute of Computer Science - FORTH, Greece. E-mail: greg@ics.forth.gr

P. Tsakalides is with the Department of Computer Science, University of Crete, Greece and the Institute of Computer Science - FORTH, Greece. Email: tsakalid@ics.forth.gr

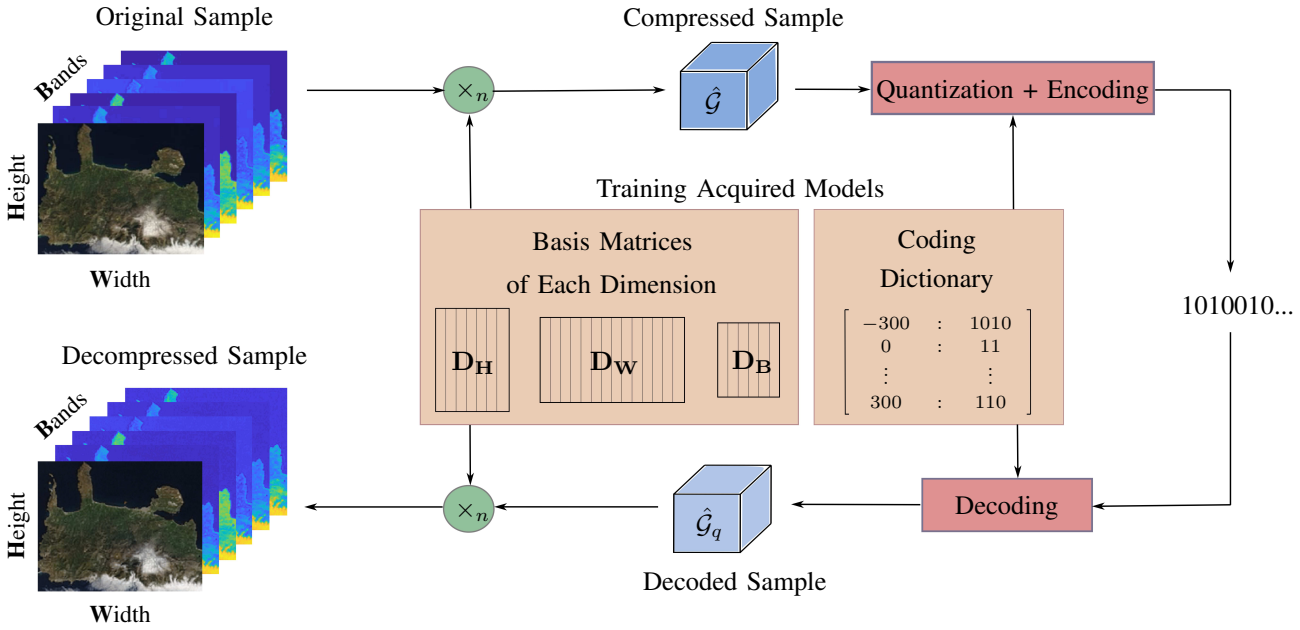


Fig. 1: Flowchart of the proposed compression method in the three dimensional case. The system takes as input a multidimensional signal, like a multispectral image, and produces a compact binary sequence in order to transmit it. During training, a basis matrix for each dimension, $\mathbf{D}_H, \mathbf{D}_W, \mathbf{D}_B$, and a symbol coding dictionary are learned from training samples. During run-time, the testing sample is represented as a multilinear combination of the learned basis matrices using the mode-n product ($\times_n$), with coefficients in the reduced-size core tensor $\hat{\mathcal{G}}$. Then, the compressed data $\hat{\mathcal{G}}$ are quantized and encoded using the learned coding dictionary. The decompressed sample can be easily obtained by synthesizing the quantized and decoded core tensor of coefficients $\hat{\mathcal{G}}_q$ with the learned basis matrices.

sition learning method we consider the compression of high dimensional data. Compression is achieved by transmitting the reduced dimension core tensor of coefficients instead of the full-dimension input tensor sample. To further increase the compression rate, the derived coefficients are quantized and encoded before transmission, using a symbol encoding dictionary that is also learned in the training process. A flowchart of the proposed compression scheme in the three dimensional case is shown in Fig. 1.

Overall, the key contributions of this paper can be summarized as follows:

- We formulate the problem of tensor decomposition in a machine learning context, where training examples are used in order to populate the appropriate representation subspaces, such that each new sample can be written as a multilinear combination of them.
- We propose an end-to-end compression algorithm for high dimensional observations, that includes both quantization and encoding to bitstreams, making it directly applicable in real-world applications.
- Although we consider 3D and 4D observations, the proposed method can handle arbitrary high dimensional data, since the structure of the data is preserved.
- The presented compression process has been developed for achieving high compression ratios with the correlations among all variables simultaneously removed at a low computational cost.
- We report the efficacy of the proposed method on synthetic 4D data, as well as on actual 3D and 4D remote sensing multispectral image sequences.

II. Related Work

Data compression algorithms can be generally classified into lossless and lossy methods. Lossless compression algorithms have been traditionally preferred to preserve all the information present in the data for scientific purposes despite their limited compression ratio. Nevertheless, the increment in the data-rate of the new-generation sensors is making more critical the necessity of obtaining higher compression ratios, making it necessary to use near-lossless and/or lossy compression techniques [17].

Many effective data-reduction and lossy compression methods are based on transform coding approaches, which first perform a data domain transformation followed by (vector) quantization or coefficient thresholding, and often conclude with an entropy coding of the remaining data coefficients. Well known examples include Fourier transform, Karhunen-Loeve Transform (KLT), Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT). Typical transformation methods for the compression of multispectral images apply DWT [18]–[21], KLT [22], [23], or Principal Components Analysis (PCA) [24] for introducing spectral decorrelation, followed by JPEG2000 [25], [26] for decorrelating the spatial information and performing the quantization stage and the entropy coding.

Nevertheless, in order to compress high dimensional data, the correlations among all dimensions must be simultaneously removed. To address this challenge, many tensor-based approaches have recently been developed. Specifically, a hyperspectral image compression algorithm based on DWT for each spectral band, and Tucker decomposition applied to the four wavelet sub-images in order to achieve more compression ratio, is introduced in [27]–[30]. A similar method based on a pair-wise multilevel grouping approach for the non-negative Tucker decomposition in a transform domain is described in [31] to overcome the high computational cost.

Instead of processing the hyperspectral image separately by spectral channel or by pixel, a patch-based low-rank tensor decomposition method is presented in [5], [6]. In this approach, the similar tensor patches are grouped by clustering and are approximately decomposed to a coefficient tensor and three dictionary matrices, which lead to a low-rank tensor representation of both the spatial and spectral modes, since the grouped tensor is assumed to be redundant. In addition, in [7], [8], the original tensor is decomposed into a factor matrix along each mode and a core tensor with reduced dimensions, which control the compression ratio. A similar approach is presented in [32], but in this compression technique, the Block Coordinate Descent method is used to find the optimal decomposition, which is initialized by using Compressed Sensing.

A better way to choose the size for each basis is presented in the Tucker compression alternative method proposed in [33] that is based on coefficient thresholding and zigzag traversal, followed by logarithmic quantization on both the transformed core tensor and its factor matrices. However, instead of fixing the number of quantization bits per transform coefficient, or the transform basis size, the lossy compressor based on the Tucker decomposition described in [9], improves the compression ratio-accuracy trade-off curve by compressing bit planes of progressively less importance.

Besides Tucker decomposition, the CP decomposition that represents a tensor as a linear combination of rank-1 tensors, has also been used for compression purposes. Specifically, a nonnegative CP decomposition algorithm has been proposed in [34] for compression of hyperspectral image time series where the input data is represented by computing their CP approximation with compressed versions of the original factor matrices. In addition, a compression algorithm based on the CP decomposition is also introduced in [10], in which a data tensor is represented by a small number of rank-1 tensors.

However, the above models require either the transmission of all rank-1 tensors with their coefficients or the core tensor and the factor matrices. An attempt to limit these requirements is presented in [35], where the rank-1 tensors are derived from a given image time series, requiring only the transmission of the coefficient vector corresponding to the learned CP decomposition. Nevertheless, in that approach only one training sample is used in the learning process, with the compression performance getting worse over time.

An extension of the above method in order to train the model using more training samples is introduced in [36], where a set of full-sized orthonormal dictionary pairs is learned for compact representation of image patches. Similarly, a tensor dictionary learning method based on CP decomposition is presented in [37] for the compression of satellite-derived image time series. However, in dictionary learning methods, a small number of dictionary elements is selected for each sample using sparse coefficients that must be calculated during run-time. Therefore, an optimization problem must be solved during the compression process for the estimation of the sparse coefficients of each new sample, resulting in high computational complexity and execution time. On the contrary, instead of extracting independent bases for each example, the proposed tensor decomposition learning method learns a generic basis for each dimension that can efficiently represent every sample at a low computational cost.

III. Tensor Preliminaries

A mathematical way to represent high dimensional data, whose entries are indexed by several variables, are *tensors*. Formally, an N-way or Nth-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is defined as a multidimensional array, whereby the order of a tensor is the number of its dimensions. Generalized from matrix format, a tensor is able to contain more structural information, being a powerful tool for dealing with multi-modal and multi-relational data [38].

A common framework for tensor computations is to turn the tensor into a matrix, which is called matricization or unfolding of the tensor [39] and allows the use of algorithms designed on matrices. Specifically, the mode-n matricization of $\mathcal{X}$ is denoted as $\text{unfold}_n(\mathcal{X}) = \mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times \prod_{i \neq n} I_i}$ and corresponds to a matrix with columns being the vectors obtained by fixing all indices of $\mathcal{X}$ except the n-th index. However, the structure of the data is not preserved in this way and the high dimensional relationships, e.g., across neighboring pixels or time instances, are lost.

A more appropriate approach involves expressing a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ as a core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ multiplied by a matrix $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ along each mode, i.e.,

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_N \mathbf{A}^{(N)}, \tag{1}$$

known as *Tucker decomposition* [14]. The Tucker decomposition is in general not unique since the factor matrices are rotation invariant. However, the subspaces defined by the factor matrices in this decomposition are unique, while the bases in these subspaces may be chosen arbitrarily and compensated for within the core tensor. The orthogonality constraint though, on the factor matrices, can help to find unique basis vectors in a Tucker representation that has several interesting and useful properties [2].

The mode-n product $\times_n$ referred in (1) denotes the tensor-times-matrix operation, which essentially projects data onto given basis factors. Formally, the mode-n product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n \times \cdots \times I_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$, which is denoted by $\mathcal{X} \times_n \mathbf{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$, has elements

$$(\mathcal{X} \times_n \mathbf{A})_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_n \ldots i_N} \cdot a_{j i_n}. \tag{2}$$

This product can also be expressed in terms of unfolded tensors as

$$\mathcal{Y} = (\mathcal{X} \times_n \mathbf{A}) \Leftrightarrow \mathbf{Y}_{(n)} = \mathbf{A} \cdot \mathbf{X}_{(n)}. \tag{3}$$

Note that the order of the multiplication in a series of distinct mode-n multiplications, is irrelevant, i.e., $\mathcal{X} \times_m \mathbf{A} \times_n \mathbf{B} = \mathcal{X} \times_n \mathbf{B} \times_m \mathbf{A}$ $(m \neq n)$, but if the modes are the same, then $\mathcal{X} \times_n \mathbf{A} \times_n \mathbf{B} = \mathcal{X} \times_n (\mathbf{B} \cdot \mathbf{A})$.

A fundamental property of tensors is the tensor rank, a generalization of matrix rank. Specifically, considering that a rank-1 Nth-order tensor is the outer product of N vectors with elements the product of the corresponding vector elements, the rank of a tensor is defined as the minimum number of rank-1 tensors needed to produce the original tensor. Unfortunately, there is no straightforward algorithm to determine the rank of a given tensor; in fact, the problem is NP-hard [40].

However, another definition for the rank of an N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the multilinear rank which is the N-tuple $(R_1, R_2, ..., R_N)$, with R_n be the column rank of $\mathbf{X}_{(n)}$ for $n = 1, .., N$ $(R_n \leq I_n)$. In other words, the multilinear rank is the rank of each mode of the tensor that can also be considered as the dimensions of the core tensor in the exact Tucker decomposition with orthogonal factor matrices.

IV. Proposed Compression Method

The proposed compression method is composed of two parts, the training and the run-time phase. During training, available data is utilized for developing the appropriate tensor representation model which is subsequently utilized for compression of new samples.

A. Training Process

Given a set of training samples, each of them being an Nth-order tensor $\mathcal{X}^j \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, we can obtain an $(N+1)$th-order training tensor $\mathcal{X} = (\mathcal{X}^1, \mathcal{X}^2, ..., \mathcal{X}^S) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times S}$, where S is the number of training samples.

To compress a multidimensional sample, we learn a basis for each dimension, $\mathbf{D}_1, .., \mathbf{D}_N$, such that each sample can be written as a multilinear combination of them, i.e.,

$$\mathcal{X}^j = \mathcal{G}^j \times_1 \mathbf{D}_1 \times_2 \cdots \times_N \mathbf{D}_N, \tag{4}$$

for $j = 1, .., S$. To achieve this, we solve the proposed optimization problem

$$\min_{\mathcal{G}, \mathbf{D}_1, ., \mathbf{D}_{N+1}} \tfrac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_N \mathbf{D}_N \times_{N+1} \mathbf{D}_{N+1}\|_F^2$$

$$\text{subject to } \|\mathbf{D}_{N+1}\|_* \leq \lambda,\ \mathbf{D}_n^T \cdot \mathbf{D}_n = \mathbf{I}_{R_n},\ n = 1, .., N \tag{5}$$

where $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N \times S}$, $\mathbf{D}_n \in \mathbb{R}^{I_n \times R_n}$, with $R_n \leq I_n$ for $n = 1, .., N$, $\mathbf{D}_{N+1} \in \mathbb{R}^{S \times S}$, and $\mathbf{I}_{R_n}$ is the identity matrix with dimensions $R_n \times R_n$. We impose a low-rank constraint on the factor matrix of the sample-variable $\mathbf{D}_{N+1}$ in order to extract the correlations in the training samples. The nuclear norm $\|\mathbf{D}_{N+1}\|_*$ of $\mathbf{D}_{N+1}$ in equation (5), is a convex relaxation of the rank minimization, since it is the sum of its singular values, with the parameter $\lambda > 0$ used to control its rank. In addition, the orthogonality constraint $\mathbf{D}_n^T \cdot \mathbf{D}_n = \mathbf{I}_{R_n}$ on the other factor matrices is used to obtain an orthogonal basis along each mode.

The problem in (5) can also be formulated as

$$\min_{\mathcal{G}, \mathbf{D}_1, ., \mathbf{D}_{N+1}} \tfrac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_N \mathbf{D}_N \times_{N+1} \mathbf{D}_{N+1}\|_F^2 + \lambda \|\mathbf{D}_{N+1}\|_*$$

$$\text{subject to } \mathbf{D}_n^T \cdot \mathbf{D}_n = \mathbf{I}_{R_n},\ n = 1, .., N. \tag{6}$$

The main task of the above problem is to recover the factor matrices $\mathbf{D}_1, .., \mathbf{D}_N, \mathbf{D}_{N+1}$ with their corresponding coefficients in the core tensor $\mathcal{G}$, as it is illustrated in Fig. 2 in the case $N = 3$, under the nuclear norm and the orthogonality constraints. To simplify the solution, we introduce the auxiliary variable $\mathbf{A} \in \mathbb{R}^{S \times S}$, such that $\mathbf{A} = \mathbf{D}_{N+1}$, in order to impose the low-rank constraint on it. Therefore, we can reformulate the minimization problem in (6) as

$$\min_{\mathcal{G}, \mathbf{D}_1, ., \mathbf{D}_{N+1}, \mathbf{A}} \tfrac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_N \mathbf{D}_N \times_{N+1} \mathbf{D}_{N+1}\|_F^2 + \lambda \|\mathbf{A}\|_*$$

$$\text{subject to } \mathbf{A} = \mathbf{D}_{N+1},\ \mathbf{D}_n^T \cdot \mathbf{D}_n = \mathbf{I}_{R_n},\ n = 1, .., N \tag{7}$$

and we can apply the Alternating Direction Method of Multipliers (ADMM) to solve the problem. In more detail, the ADMM scheme takes into account the separate structure of each variable in (7), relying on the minimization of its unconstrained augmented Lagrangian function which is given by

$$\begin{gathered}\mathcal{L}(\mathcal{G}, \mathbf{D}_1, .., \mathbf{D}_N, \mathbf{D}_{N+1}, \mathbf{A}, \mathbf{Y}) = \\ \tfrac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_{N+1} \mathbf{D}_{N+1}\|_F^2 + \lambda \|\mathbf{A}\|_* + \\ \mathbf{Y} \cdot (\mathbf{A} - \mathbf{D}_{N+1}) + \tfrac{p}{2} \|\mathbf{A} - \mathbf{D}_{N+1}\|_F^2,\end{gathered} \tag{8}$$

where $\mathbf{Y} \in \mathbb{R}^{S \times S}$ stands for the Lagrange multiplier matrix, while $p > 0$ denotes the step size parameter. Following the general algorithmic strategy of the ADMM scheme, we seek for the stationary point solving iteratively for each individual variable while keeping the others fixed. As a result, we create the following sequence of update rules at each iteration l, until a maximum number of iterations is reached or the decrease in the objective function between consecutive iterations is smaller than a predefined threshold.

- For the auxiliary variable $\mathbf{A}$, we solve the problem

$$\begin{gathered}\mathbf{A} = \text{argmin}_{\mathbf{A}} \mathcal{L} = \\ \text{argmin}_{\mathbf{A}} (\lambda \|\mathbf{A}\|_* + \mathbf{Y} \cdot (\mathbf{A} - \mathbf{D}_{N+1}) + \tfrac{p}{2} \|\mathbf{A} - \mathbf{D}_{N+1}\|_F^2) \Leftrightarrow \\ \mathbf{A} = \text{argmin}_{\mathbf{A}} (\mathbf{Y} \cdot (\mathbf{A} - \mathbf{D}_{N+1}) + \tfrac{p}{2} \|\mathbf{A} - \mathbf{D}_{N+1}\|_F^2) \\ \text{subject to } \|\mathbf{A}\|_* \leq \lambda.\end{gathered}$$

  We firstly minimize the augmented Lagrangian function with respect to $\mathbf{A}$ by setting $\nabla_{\mathbf{A}} \mathcal{L} = 0$ without regard for its nuclear norm constraint, i.e.,

$$\nabla_{\mathbf{A}} (\mathbf{Y} \cdot (\mathbf{A} - \mathbf{D}_{N+1}) + \tfrac{p}{2} \|\mathbf{A} - \mathbf{D}_{N+1}\|_F^2) = 0 \Rightarrow$$

$$\mathbf{Y} + p \cdot (\mathbf{A} - \mathbf{D}_{N+1}) = 0$$

  and we update

$$\hat{\mathbf{A}} \leftarrow \mathbf{D}_{N+1} - \frac{\mathbf{Y}}{p}. \tag{9}$$

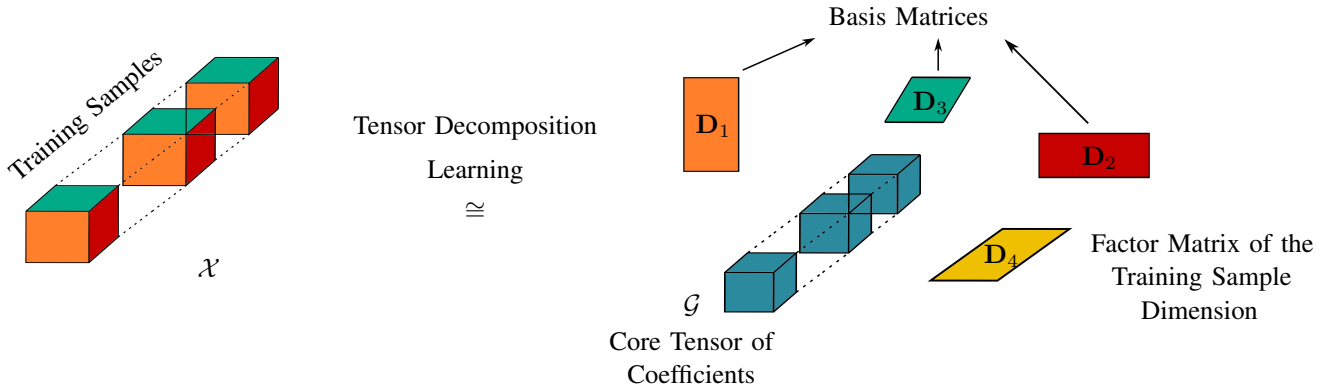


Fig. 2: Tensor decomposition learning method in the case of three dimensional samples. The tensor $\mathcal{X}$ of the training samples is represented as a multilinear combination of factor matrices of each dimension (with respective colors), $\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3, \mathbf{D}_4$, with coefficients in the core tensor $\mathcal{G}$, by solving a constrained optimization problem using ADMM. Given the learned basis matrices $\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3$, each new sample can be expressed in terms of them.

To impose the nuclear norm constraint on $\hat{\mathbf{A}}$, we take its Singular Value Decomposition (SVD) and we hold some of the corresponding singular values, depending on the parameter λ, i.e.,

$$\begin{aligned}(\mathbf{U}, \mathbf{S}, \mathbf{V}) &\leftarrow \text{svd}(\hat{\mathbf{A}}) \\ \mathbf{s} &\leftarrow P_\lambda(\text{diag}(\mathbf{S})) \\ \mathbf{A} &\leftarrow \mathbf{U} \cdot \text{diag}(\mathbf{s}) \cdot \mathbf{V}^T\end{aligned}$$

where P_λ denotes the projection onto the l_1 ball with radius λ and diag$(\cdot)$ gets diagonal elements of a matrix, or creates a diagonal matrix with the elements of the input vector on the main diagonal. In our experiments, we keep 90% of the information of the singular values.

- For the basis matrices $\mathbf{D}_n, n = 1, .., N$, we solve the problem

$$\begin{gathered}\mathbf{D}_n = \text{argmin}_{\mathbf{D}_n} \mathcal{L} = \\ \text{argmin}_{\mathbf{D}_n} (\tfrac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_{N+1} \mathbf{D}_{N+1}\|_F^2) \\ \text{subject to } \mathbf{D}_n^T \cdot \mathbf{D}_n = \mathbf{I}_{R_n}.\end{gathered}$$

Similarly, we firstly update each $\mathbf{D}_n$ by setting $\nabla_{\mathbf{D}_n} \mathcal{L} = 0$ without regard for its orthogonal constraint, i.e.,

$$\nabla_{\mathbf{D}_n} (\tfrac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_{N+1} \mathbf{D}_{N+1}\|_F^2) = 0 \Rightarrow$$

$$\begin{gathered}(\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_N \mathbf{D}_N \times_{N+1} \mathbf{D}_{N+1})_{(n)} \cdot \\ \cdot (\mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_{n-1} \mathbf{D}_{n-1} \times_{n+1} \\ \mathbf{D}_{n+1} \times_{n+2} \cdots \times_N \mathbf{D}_N \times_{N+1} \mathbf{D}_{N+1})_{(n)}^T = 0.\end{gathered}$$

By setting $\mathcal{C}_n = \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_{n-1} \mathbf{D}_{n-1} \times_{n+1} \mathbf{D}_{n+1} \times_{n+2} \cdots \times_{N+1} \mathbf{D}_{N+1}$, we have

$$(\mathcal{X} - \mathcal{C}_n \times_n \mathbf{D}_n)_{(n)} \cdot \mathbf{C}_{n_{(n)}}^T = 0$$

and we update

$$\hat{\mathbf{D}}_n \leftarrow (\mathbf{X}_{(n)} \cdot \mathbf{C}_{n_{(n)}}^T) \cdot (\mathbf{C}_{n_{(n)}} \cdot \mathbf{C}_{n_{(n)}}^T)^{-1}, \qquad (10)$$

where $\mathbf{B}^{-1}$ is denoted the pseudoinverse of a matrix $\mathbf{B}$. Subsequently, we apply a *QR* factorization on each $\hat{\mathbf{D}}_n$ in order to impose the orthogonality constraint on them. Then, the basis matrix $\mathbf{D}_n$ of each mode of the tensor is constructed as the R_n columns of the unitary matrix $\mathbf{Q}$ of the corresponding *QR* factorization, i.e.,

$$\begin{aligned}(\mathbf{Q}, \mathbf{R}) &\leftarrow QR(\hat{\mathbf{D}}_n) \\ \mathbf{D}_n &\leftarrow \mathbf{Q}(:, 1 : R_n)\end{aligned}$$

for $n = 1, .., N$.

- The update of the factor matrix $\mathbf{D}_{N+1}$ that corresponds to the sample-variable is different from the other factor matrices. Specifically, we solve the unconstrained problem

$$\begin{gathered}\mathbf{D}_{N+1} = \text{argmin}_{\mathbf{D}_{N+1}} \mathcal{L} = \\ \text{argmin}_{\mathbf{D}_{N+1}} (\tfrac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_{N+1} \mathbf{D}_{N+1}\|_F^2 \\ + \mathbf{Y} \cdot (\mathbf{A} - \mathbf{D}_{N+1}) + \tfrac{p}{2} \|\mathbf{A} - \mathbf{D}_{N+1}\|_F^2).\end{gathered}$$

By setting $\nabla_{\mathbf{D}_{N+1}} \mathcal{L} = 0$ and $\mathcal{C}_{N+1} = \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_N \mathbf{D}_N$, we have

$$\begin{gathered}\nabla_{\mathbf{D}_{N+1}} (\tfrac{1}{2} \|\mathcal{X} - \mathcal{C}_{N+1} \times_{N+1} \mathbf{D}_{N+1}\|_F^2 \\ + \mathbf{Y} \cdot (\mathbf{A} - \mathbf{D}_{N+1}) + \tfrac{p}{2} \|\mathbf{A} - \mathbf{D}_{N+1}\|_F^2) = 0 \Rightarrow\end{gathered}$$

$$\begin{gathered}-(\mathcal{X} - \mathcal{C}_{N+1} \times_{N+1} \mathbf{D}_{N+1})_{(N+1)} \cdot \mathbf{C}_{N+1_{(N+1)}}^T - \mathbf{Y} \\ -p \cdot (\mathbf{A} - \mathbf{D}_{N+1}) = 0 \Rightarrow\end{gathered}$$

$$\begin{gathered}(\mathcal{C}_{N+1} \times_{N+1} \mathbf{D}_{N+1})_{(N+1)} \cdot \mathbf{C}_{N+1_{(N+1)}}^T + p \cdot \mathbf{D}_{N+1} = \\ = \mathbf{X}_{(N+1)} \cdot \mathbf{C}_{N+1_{(N+1)}}^T + \mathbf{Y} + p \cdot \mathbf{A}\end{gathered}$$

and we update

$$\begin{gathered}\mathbf{D}_{N+1} \leftarrow (\mathbf{X}_{(N+1)} \cdot \mathbf{C}_{N+1_{(N+1)}}^T + \mathbf{Y} + p \cdot \mathbf{A}) \cdot \\ \cdot (\mathbf{C}_{N+1_{(N+1)}} \cdot \mathbf{C}_{N+1_{(N+1)}}^T + p \cdot \mathbf{I}_S)^{-1},\end{gathered} \qquad (11)$$

where $\mathbf{I}_S$ is the identity matrix with dimensions $S \times S$.

- For the coefficients of the updated basis matrices, we solve the unconstrained problem

$$\begin{gathered}\mathcal{G} = \text{argmin}_{\mathcal{G}} \mathcal{L} = \\ \text{argmin}_{\mathcal{G}} (\tfrac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_{N+1} \mathbf{D}_{N+1}\|_F^2).\end{gathered}$$

By setting $\nabla_{\mathcal{G}}\mathcal{L} = 0$, we have

$$\nabla_{\mathcal{G}}(\tfrac{1}{2}\|\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_{N+1} \mathbf{D}_{N+1}\|_F^2) = 0 \Rightarrow$$

$$(\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_{N+1} \mathbf{D}_{N+1}) \times_1 \mathbf{D}_1^{-1} \times_2 \cdots \cdots \times_N \mathbf{D}_N^{-1} \times_{(N+1)} \mathbf{D}_{N+1}^{-1} = 0 \Rightarrow$$

$$\mathcal{X} \times_1 \mathbf{D}_1^T \times_2 \cdots \times_N \mathbf{D}_N^T \times_{(N+1)} \mathbf{D}_{N+1}^{-1} - \mathcal{G} \times_1 (\mathbf{D}_1^T \cdot \mathbf{D}_1) \times_2 \cdots \times_N (\mathbf{D}_N^T \cdot \mathbf{D}_N) \times_{N+1} (\mathbf{D}_{N+1}^{-1} \cdot \mathbf{D}_{N+1}) = 0$$

and we update

$$\mathcal{G} \leftarrow \mathcal{X} \times_1 \mathbf{D}_1^T \times_2 \cdots \times_N \mathbf{D}_N^T \times_{N+1} \mathbf{D}_{N+1}^{-1}, \quad (12)$$

where $\mathbf{D}_n^{-1} = \mathbf{D}_n^T$ and $\mathbf{D}_n^T \cdot \mathbf{D}_n = \mathbf{I}_{R_n}$ for the orthogonal matrices $\mathbf{D}_n, n = 1, .., N$.

- Finally, we update the Lagrangian multiplier matrix as

$$\mathbf{Y}^{(l)} \leftarrow \mathbf{Y}^{(l-1)} + p \cdot (\mathbf{A} - \mathbf{D}_{N+1}), \quad (13)$$

at each iteration l. In our setup, we set $p = 0.01$.

The overall algorithm for learning the basis matrix of each dimension is summarized in Algorithm 1.

Besides the basis matrices, we also obtain a symbol encoding dictionary in order to represent the compressed data in a binary format for transmission or storage. Specifically, the learned coding dictionary maps a binary number to a discrete set of symbols K, using fewer bits for the most common symbols, using Huffman coding. In our approach, the symbols of this dictionary are defined as the quantization levels of the Lloyd-max quantization of the core tensor obtained from the proposed tensor decomposition learning method, into a given number of bits. Note that we apply the Lloyd-max algorithm, which is a non-uniform quantizer, since the range of values of the core tensor is usually very large and we need smaller quantization steps where values appear most frequently, in order to reduce the quantization error.

B. Compression and Decompression

In the compression process, each new sample $\hat{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ can be written as a multilinear combination of the learned basis matrices $\mathbf{D}_1, .., \mathbf{D}_N$, i.e.,

$$\hat{\mathcal{X}} = \hat{\mathcal{G}} \times_1 \mathbf{D}_1 \times_2 \cdots \times_N \mathbf{D}_N. \quad (14)$$

Therefore, instead of the sample $\hat{\mathcal{X}}$, we only need to transmit the reduced-size core tensor of coefficients $\hat{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ that is given by the mode-n multiplication of the data with the transposed basis matrices, i.e.,

$$\hat{\mathcal{G}} = \hat{\mathcal{X}} \times_1 \mathbf{D}_1^T \times_2 \cdots \times_N \mathbf{D}_N^T. \quad (15)$$

Note that the core tensor $\hat{\mathcal{G}}$ has a significantly smaller size than the original sample, allowing a high compression ratio with a low computational cost, since it is required only a mode-n multiplication to obtain the compressed data.

However, a crucial step for data transmission or storage is the need for representation of the compressed data in a binary format. To achieve this, we quantize the coefficient tensor $\hat{\mathcal{G}}$ to a given number of bits b, using the set of symbols $K = \{q_1, q_2, ..., q_{2^b}\}$ of the learned encoding dictionary as the quantization levels and the middle of the quantization levels

Algorithm 1 Tensor Decomposition Learning Algorithm

Input: The training tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times S}$, the reduced dimensions of the core tensor $(R_1, .., R_N)$, the step size parameter p, the maximum number of iterations l_{max} and the tolerance tol for the stopping criterion.
Output: The basis matrices $\mathbf{D}_n \in \mathbb{R}^{I_n \times R_n}, n = 1, .., N$ and the core tensor of coefficients $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N \times S}$ for the training samples.

Initialization: Set $\mathbf{D}_1, .., \mathbf{D}_{N+1}$ randomly where $\mathbf{D}_{N+1} \in \mathbb{R}^{S \times S}$, $\mathbf{Y}^{(0)} \in \mathbb{R}^{S \times S}$ with all zeros, $\mathcal{G} = \mathcal{X} \times_1 \mathbf{D}_1^T \times_2 \cdots \times_N \mathbf{D}_N^T \times_{N+1} \mathbf{D}_{N+1}^{-1}$, $e_0 = 1$ and $l = 0$;
repeat
  $l \leftarrow l + 1$;
  // Update $\mathbf{A}$ *with the low-rank constraint*
  $\hat{\mathbf{A}} \leftarrow \mathbf{D}_{N+1} - \mathbf{Y}/p$;
  $(\mathbf{U}, \mathbf{S}, \mathbf{V}) \leftarrow \text{svd}(\hat{\mathbf{A}})$;
  $\mathbf{s} \leftarrow P_\lambda(\text{diag}(\mathbf{S}))$;
  $\mathbf{A} \leftarrow \mathbf{U} \cdot \text{diag}(\mathbf{s}) \cdot \mathbf{V}^T$;
  // Update $\mathbf{D}_1, .., \mathbf{D}_N$ *with the orthogonality constraint*
  for n = 1,..., N **do**
    $\mathcal{C}_n \leftarrow \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_{n-1} \mathbf{D}_{n-1} \times_{n+1} \mathbf{D}_{n+1} \times_{n+2} \cdots \times_{N+1} \mathbf{D}_{N+1}$;
    $\hat{\mathbf{D}}_n \leftarrow (\mathbf{X}_{(n)} \cdot \mathbf{C}_{n_{(n)}}^T) \cdot (\mathbf{C}_{n_{(n)}} \cdot \mathbf{C}_{n_{(n)}}^T)^{-1}$;
    $(\mathbf{Q}, \mathbf{R}) \leftarrow QR(\hat{\mathbf{D}}_n)$;
    $\mathbf{D}_n \leftarrow \mathbf{Q}(:, 1 : R_n)$;
  end for
  // Update $\mathbf{D}_{N+1}$ *of the sample-variable*
  $\mathcal{C}_{N+1} \leftarrow \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_N \mathbf{D}_N$;
  $\mathbf{D}_{N+1} \leftarrow (\mathbf{X}_{(N+1)} \cdot \mathbf{C}_{N+1_{(N+1)}}^T + \mathbf{Y} + p \cdot \mathbf{A}) \cdot (\mathbf{C}_{N+1_{(N+1)}} \cdot \mathbf{C}_{N+1_{(N+1)}}^T + p \cdot \mathbf{I}_S)^{-1}$;
  // Update the core tensor of coefficients $\mathcal{G}$
  $\mathcal{G} \leftarrow \mathcal{X} \times_1 \mathbf{D}_1^T \times_2 \cdots \times_N \mathbf{D}_N^T \times_{N+1} \mathbf{D}_{N+1}^{-1}$;
  // Update the Lagrangian multiplier matrix $\mathbf{Y}$
  $\mathbf{Y}^{(l)} \leftarrow \mathbf{Y}^{(l-1)} + p \cdot (\mathbf{A} - \mathbf{D}_{N+1})$;
  // Stopping criterion
  $e_l \leftarrow \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_{N+1} \mathbf{D}_{N+1}\|_2 / \|\mathcal{X}\|_2$;
until $|e_{l-1} - e_l| \leq tol$ or $l = l_{max}$
$\mathcal{G} \leftarrow \mathcal{X} \times_1 \mathbf{D}_1^T \times_2 \cdots \times_N \mathbf{D}_N^T$;
return $(\mathcal{G}, \mathbf{D}_1, .., \mathbf{D}_N)$;

$c_k = \frac{q_k + q_{k+1}}{2}$, $k = 1, ..., 2^b - 1$ as the quantization boundaries. Then, the quantized coefficients $\hat{\mathcal{G}}_q$ are encoded in order to be transmitted, using Huffman coding and the learned coding dictionary. Since Huffman coding is a lossless compression algorithm that is used to further compress the data without losing any other information, the quantization process is the only additional loss of our learning method that depends on the quantization bits used.

Finally, to decompress the received data, we only need to synthesize the learned basis matrices $\mathbf{D}_1, .., \mathbf{D}_N$ with the decoded and quantized core tensor of coefficients $\hat{\mathcal{G}}_q$, i.e.,

$$\hat{\mathcal{X}} \approx \hat{\mathcal{G}}_q \times_1 \mathbf{D}_1 \times_2 \cdots \times_N \mathbf{D}_N, \quad (16)$$

giving an efficient approximation of the original data.

C. Computational Complexity

An important characteristic of an efficient compression algorithm is the low computational complexity of the compressor in order to satisfy the requirements of the compression applications that must be executed under tight resources and latency constraints. Therefore, we need to examine the complexity of the proposed compression process by analyzing the three stages that it is consisted.

The first stage is the mode-n multiplication of a sample $\hat{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ with the learned basis matrices $\mathbf{D}_n \in \mathbb{R}^{I_n \times R_n}$ for $n = 1, .., N$ and $R_n \leq I_n$ (often $R_n = I_n/2$), in order to obtain the reduced-size core tensor of coefficients $\hat{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ that need to be transmitted. Since the mode-n product can also be expressed in terms of unfolded tensors as in the equation (3), N matrix multiplications must be performed. Therefore, there are needed $O(R_1 I_1 I_2 \cdots I_N + R_1 R_2 I_2 \cdots I_N + ... + R_1 R_2 \cdots R_n I_n I_{n+1} \cdots I_N + ... + R_1 R_2 \cdots R_N I_N)$ operations using existing efficient implementations for the matrix products. In addition, the order of the multiplications can be varied as desired for optimizing the speed, since it is irrelevant in a series of distinct mode-n products. So, the final cost of this stage is $O(R_i I_1 I_2 \cdots I_N)$ with $R_i = \min(R_1, .., R_N)$.

Afterward, the compressed tensor $\hat{\mathcal{G}}$ is quantized to b bits using the symbols of the learned encoding dictionary. Specifically, the quantization process involves the search of the nearest symbol to each compressed value. Therefore, we need to check $O(\log(2^b)) = O(b)$ symbols for each of the $R_1 R_2 \cdots R_N$ entries of $\hat{\mathcal{G}}$, since the dictionary consists of 2^b ordered elements. This results in $O(R_1 R_2 \cdots R_N b)$ operations.

The final stage of the compression process is the encoding of the quantized core tensor $\hat{\mathcal{G}}_q$ in order to be transmitted. Since the coding dictionary has been obtained during the training process, we only need to search the quantized values in the dictionary and replace them with their binary representation. Similar with the previous step, we need $O(R_1 R_2 \cdots R_N b)$ operations to check 2^b ordered symbols for the $R_1 R_2 \cdots R_N$ entries of $\hat{\mathcal{G}}_q$.

As a conclusion, the proposed scheme needs $O(I_1 I_2 \cdots I_N \cdot \max(R_i, b))$ operations to efficiently compress and transmit a new sample. This is a low computational cost that depends on the dimensions of the original and the compressed sample and the number of quantization bits.

V. Experimental Results

The performance of the proposed compression algorithm is quantified on both synthetic and real remote sensing data. Specifically, we have experimented with 4D random generated synthetic data, as well as with 3D and 4D real satellite multispectral image sequences.

To assess the performance of our algorithm, we use the *Normalized Root Mean Square Error* (NRMSE) which is defined as

$$\text{NRMSE} = \frac{\|\mathcal{Y} - \hat{\mathcal{Y}}\|_2}{\|\mathcal{Y}\|_2}, \tag{17}$$

where $\mathcal{Y}$ and $\hat{\mathcal{Y}}$ are the original and the reconstructed signal, respectively. Additionally, we evaluate the performance of the decompressed multispectral images in terms of the *Peak Signal to Noise Ratio* (PSNR) given by

$$\text{PSNR} = 10 \cdot \log_{10}(\frac{R^2}{\text{MSE}}), \tag{18}$$

where R is the maximum value of the input image and MSE stands for the Mean Square Error defined as the average of the squares of the differences between the original and the estimated images. Note that higher PSNR values correspond to a better quality decompressed image.

A. Synthetic Data Scenario

In the synthetic data case, we generated 100 fourth-order tensors of size $20 \times 20 \times 20 \times 20$ with multilinear rank $(10, 10, 10, 10)$. The samples are synthesized by four basis matrices of size 20×10 and a coefficient core tensor of size $10 \times 10 \times 10 \times 10$ corresponding to each sample. The basis matrix of each dimension, as well as the coefficient tensor of each sample, have uniformly distributed random values in the interval (0,1).

Concerning the tensor learning method of the proposed compression algorithm, the choice of the dimensions of the core tensor is critical for the quality of the representation of the samples as a multilinear combination of basis matrices along each mode, as well as for the compression ratio achieved in the compression process. Note that in this work, the *compression ratio* is defined as the reduction in size relative to the uncompressed size, i.e.,

$$\text{Compression Ratio} = 1 - \frac{\text{Compressed Size}}{\text{Uncompressed Size}} \tag{19}$$

where the compressed size indicates the product of the dimensions of the core tensor of the learned decomposition and the uncompressed size is the product of the original dimensions.

The trade-off between compression and reconstruction is reported in Table I for several dimensions of the core tensor as a percentage of the original dimensions for 50 testing samples, obtained without quantization and coding. For instance, by taking 50% of the original dimensions $I_1 \times \cdots \times I_N$, as the size $R_1 \times \cdots \times R_N$ of the core tensor, then $R_n = I_n/2$ for $n = 1, .., N$, with rounding to the nearest integer. As we can see from the results, the compression quality is improved rapidly from 50% and above of the original dimensions as the size of the core tensor, which is the multilinear rank of the data. The reason is that the dimensions of the core tensor actually indicate the multilinear rank of the data, which gives a better representation as a multilinear combination of the learned basis matrices. However, the reconstruction error is small even for very high compression ratios and is decreasing as the compression is decreasing.

An other parameter of the proposed tensor decomposition learning method is the number of training samples used. The results on synthetic data were almost the same for different number of training samples. Therefore, we used 50 samples for the training process in this scenario.

TABLE I: Mean reconstruction error (and standard deviation (SD)) of 50 synthetic samples, obtained without quantization and coding, for different dimensions of the core tensor as a percentage of the original dimensions and the corresponding compression ratio.

Dimensions of the Core tensor (% of the original dimensions)	**Mean Testing NRMSE (SD)**	**Compression Ratio**
10	0.0069 ($7 \cdot 10^{-4}$)	0.9999
20	0.0056 ($5 \cdot 10^{-4}$)	0.9984
30	0.0038 ($3 \cdot 10^{-4}$)	0.9919
40	0.0024 ($3 \cdot 10^{-4}$)	0.9744
50	10^{-9} (10^{-12})	0.9375
60	$8 \cdot 10^{-11}$ ($2 \cdot 10^{-13}$)	0.8704
70	$3 \cdot 10^{-11}$ ($6 \cdot 10^{-14}$)	0.7599
80	$8 \cdot 10^{-12}$ (10^{-13})	0.5904
90	$7 \cdot 10^{-12}$ (10^{-13})	0.3439
100	$6 \cdot 10^{-16}$ ($2 \cdot 10^{-17}$)	0

Nevertheless, the proposed learning method is followed by quantization and encoding of the reduced-size core tensor. The number of quantization bits controls the additional loss of our learning method, as well as the compression achieved in terms of the number of *bits per pixel per band* (bpppb), which is defined as the number of the compressed bits over the number of the original elements (i.e. the product of the size of the original tensor).

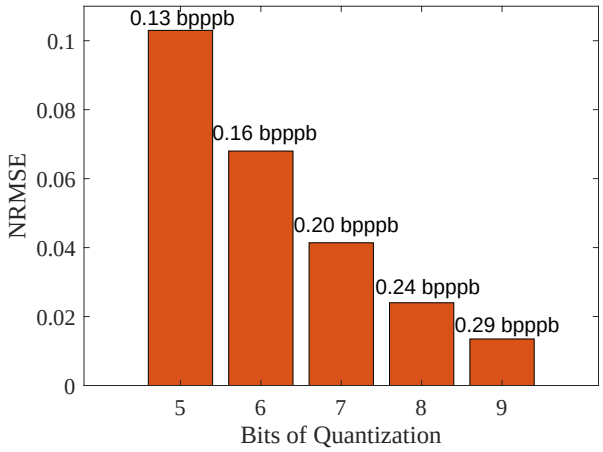


Fig. 3: Mean reconstruction error for different bits of quantization on 50 synthetic samples with the corresponding bpppb.

The impact of quantization in the proposed compression algorithm is evaluated in Fig. 3, in which the mean reconstruction error over 50 testing samples is presented for different bits of quantization, with the corresponding bpppb, by taking the 50% of the original dimensions as the size of the core tensor. Note that the original number of bits in this case is 16, since the core tensor has values in the range $[-2 \cdot 10^4, 2 \cdot 10^4]$. As it was expected, the more the bits of quantization, the better the reconstruction quality, but also the more the bpppb used. Therefore, we need to choose the number of quantization bits that gives an efficient reconstruction using the less bpppb.

B. Real 3D Data Scenario

In the real data case, we firstly experimented on publicly available remote sensing multispectral images daily acquired by the MODIS satellite[1] over the region of Chania in Crete. Specifically, we used the 365 images of 2017 in the training process and the corresponding 365 images of 2018 for testing, each of them modeled as a third-order tensor of size $227 \times 348 \times 5$ with two spatial and one spectral dimension.

However, instead of processing the multispectral images by spectral channel or by pixel, we represented each local patch of the image as a third-order tensor, which can preserve both the neighborhood relationship across the spatial dimensions and the global correlation among the spectral dimension. In addition, a patch of the image has usually a smaller rank due to stronger correlations, and as a consequence it needs smaller base matrices, achieving higher compression rates. Therefore, we took 8×8 patches in the spatial dimensions, with all their spectral bands, and we applied our tensor decomposition learning method separately for each $8 \times 8 \times 5$ patch of the image. Nevertheless, our method can be employed on the whole multispectral image as well.

TABLE II: Mean reconstruction error (and standard deviation (SD)) of 50 multispectral images, obtained without quantization and coding.

Dimensions of the Core tensor (% of the original dimensions)	**Mean Testing NRMSE (SD)**	**Compression Ratio**
20	1.1070 (0.0088)	0.9874
30	0.1268 (0.0597)	0.9749
40	0.1049 (0.0529)	0.9436
50	0.0371 (0.0191)	0.8493
60	0.0280 (0.0141)	0.7660
70	0.0141 (0.0080)	0.5507
80	0.0141 (0.0080)	0.5507
90	0.0067 (0.0044)	0.2317
100	$7 \cdot 10^{-16}$ ($6 \cdot 10^{-18}$)	0

As in the synthetic data case, first we evaluated our learning method on various dimensions of the core tensor as a percentage of the original dimensions. Specifically, using 50 multispectral images for training and 50 images for testing without quantization and coding, the results are reported in Table II. However, the core tensor may have the same dimensions for some percentage values when using small patches of

[1]NASA Worldview: https://worldview.earthdata.nasa.gov/

the image. This is the reason why 70% and 80% of the original dimensions give the same recovery error and compression ratio. To clearly show the trade-off between compression and reconstruction, we also report the compression ratio along with the reconstruction error for each percentage. One can observe from Table II that the bigger the size of the core tensor, the better the reconstruction quality, but the less the compression achieved. In addition, we can deduce from the results that our data has full multilinear rank. Nevertheless, we can efficiently represent them as a multilinear combination of reduced-size learned factor matrices using our method. Specifically, we have an efficient reconstruction with a high compression ratio by taking 50% of the original dimensions as the size of the core tensor. Hence, in the following experiments, we use these dimensions for the core tensor.

In addition, we evaluated our learning method with respect to the number of multispectral images used in the training process. As we can see from the results presented in Fig. 4, the mean reconstruction error over 50 testing images is almost the same for different number of training samples, as in the synthetic data scenario. However, we can efficiently learn a basis along each mode even for a small number of training images. Therefore, we used 50 samples for the training process in our experiments.

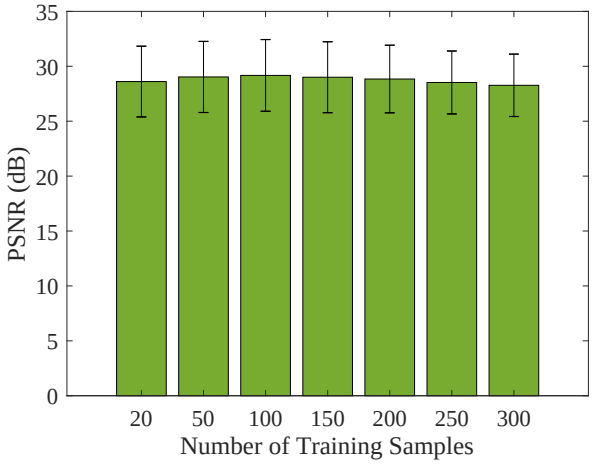


Fig. 4: Mean reconstruction error for different number of training samples on 50 multispectral images.

For the subsequently quantization process, we examined the performance of our compression scheme for different number of quantization bits over 50 multispectral images. The quality of the decompressed images is increasing, as the bits of quantization are increasing, according to the results reported in Fig. 5. Nevertheless, the compression achieved is decreasing in this case. Note here that the original number of bits is 13, since the core tensor has values in the range $[-3.8 \cdot 10^3, 3.4 \cdot 10^3]$.

Finally, we compared our algorithm with other compression methods, namely JPEG2000 which is among the best-performing algorithms for 2D image compression, JPEG2000+DWT with the additional DWT for spectral decorrelation, and Tucker Thresholding [33] in which is used Tucker decomposition as in our approach, but it is based on coefficient

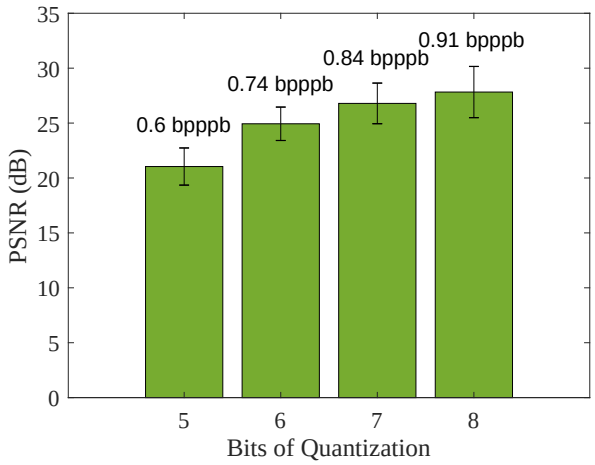


Fig. 5: Mean reconstruction error for different bits of quantization on 50 multispectral images with the corresponding bpppb.

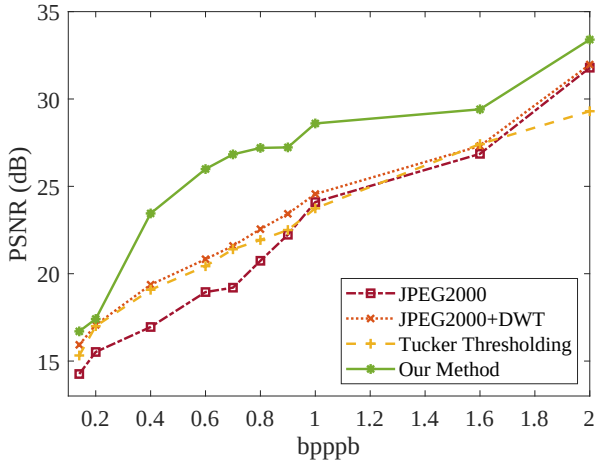
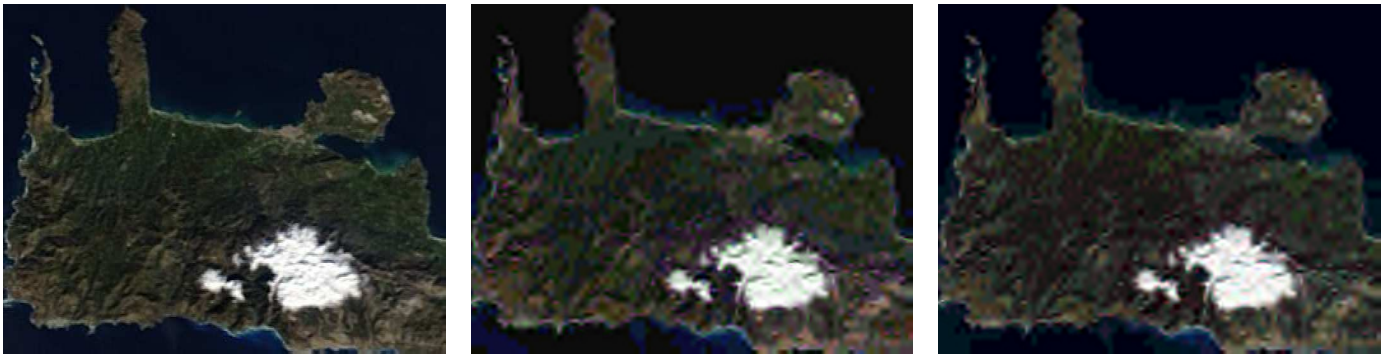


Fig. 6: Comparison of our scheme with other compression methods on a multispectral image, for different number of bpppb.

thresholding and zigzag traversal. The above methods have not been applied on fourth or higher-order tensor data. This is the reason that we compared our compression scheme on 3D multispectral images.

According to the results presented in Fig. 6 for a multispectral image, our approach outperforms the other compression methods for each number of bpppb, especially from 0.4 and above. JPEG2000 has the worst performance in most cases as it is applied only on the spatial dimensions, while JPEG2000+DWT and Tucker Thresholding have similar behavior. However, in each case, the quality of the decompressed images is increasing as the number of bpppb is increasing.

The potential of our method over the other compression methods is verified in Fig. 7 which illustrates the RGB representation of the original and the decompressed multispectral images for each method, using 0.8 bpppb. As we can see, JPEG2000 and JPEG2000+DWT can not preserve the colors in the image, while Tucker Thresholding gives a blurred version



(a) Original Image (b) JPEG2000, PSNR:20.37dB (c) JPEG2000+DWT, PSNR:21.27dB

(d) Tucker Thresholding, PSNR:21.93dB (e) Our Method, PSNR:27.20dB

Fig. 7: RGB representation of the original and the decompressed multispectral images for different compression methods, using 0.8 bpppb.

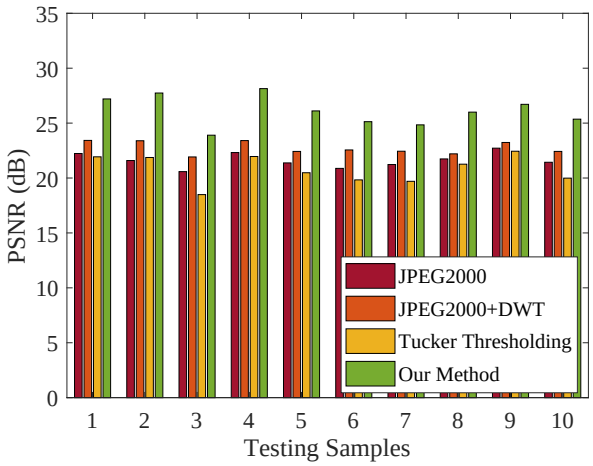


Fig. 8: Comparison of our scheme with other compression methods on different multispectral images, using 0.8 bpppb.

of it. On the contrary, our method gives a visually very similar image to the original one, even for high compression. These results can also be observed in Fig. 8 for different multispectral images, using the same number of bpppb.

Besides the rate-distortion performance, we also evaluated our algorithm in terms of its computational complexity. Specifically, Table III reports the compression time for different sizes

TABLE III: Compression time as a function of the size of the multispectral image (without quantization and coding).

Size	Compression Time (sec)	
	Tucker Thresholding	Our Method
$200 \times 300 \times 5$	0.4061	0.0038
$100 \times 150 \times 5$	0.1157	0.0018
$50 \times 75 \times 5$	0.0763	0.0018

of a multispectral image, considering only the compression stage, without quantization and coding, using a computer equipped with an Intel Core i5-4590 CPU 3.30GHz, 8GB RAM, and implemented in MATLAB R2015a. The results demonstrate the speedup achieved by our proposed compression method even when the size of the data increases since our approach only requires a multilinear projection into the learned basis matrices. On the other hand, the competitive compression method needs considerable more time for the compression stage and this time increases rapidly as the size of the data increases. The speed at compressing arbitrary high dimensional samples is an important advantage of our method since the compression applications often need to be executed under tight resources and latency constraints, such as in remote sensing environments where the multispectral images are collected on-board satellites and need to be transferred to the ground-based stations.

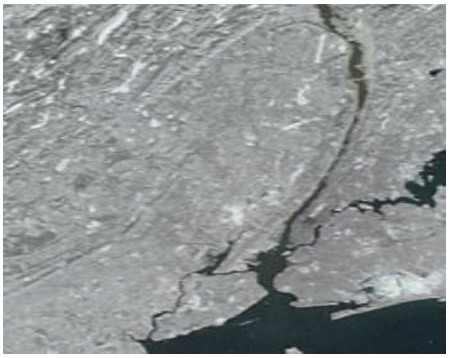
(a) Original Image - Chania

(b) Original Image - Barcelona

(c) Original Image - New York

(d) Decompressed Image - Chania

(e) Decompressed Image - Barcelona

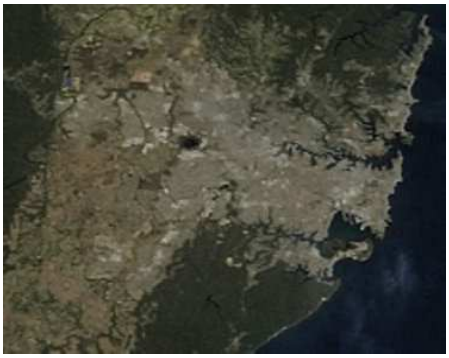


(f) Decompressed Image - New York

(g) Original Image - Sydney

(h) Original Image - Buenos Aires

(i) Original Image - Tokyo



(j) Decompressed Image - Sydney

(k) Decompressed Image - Buenos Aires

(l) Decompressed Image - Tokyo

Fig. 9: RGB representation of an original and the corresponding decompressed multispectral image from an image time series of each different datasets, using about 0.88 bpppb.

C. Real 4D Data Scenario

In the real data case, we also experimented on time series of remote sensing multispectral images daily acquired by the MODIS satellite over the regions of Chania ($227 \times 348 \times 5$), Barcelona ($234 \times 300 \times 5$), New York ($350 \times 350 \times 5$), Sydney ($300 \times 372 \times 5$), Buenos Aires ($288 \times 360 \times 5$) and Tokyo ($290 \times 332 \times 5$). Similarly with the 3D data scenario, we used 365 images of 2017 in the training process of each region and the corresponding 365 images of 2018 for testing. However, in this case, we used time series of the images per 7 days, without overlap, each of them modeled as a fourth-order tensor with an additional time-variable.

Therefore, we applied the proposed compression method for each dataset on 8×8 patches with all their spectral bands and time instances, using 50 training and 50 testing time series. In addition, we did not reduce the time-variable because of the abrupt changes in the images of each day with the appearance of clouds.

TABLE IV: Mean reconstruction error (and standard deviation) of 50 multispectral image time series for different datasets, using about 0.88 bpppb.

	PSNR (dB)	NRMSE
Chania	27.84 (1.44)	0.0418 (0.0084)
Barcelona	28.33 (1.15)	0.0367 (0.0071)
New York	29.18 (1.15)	0.0281 (0.0067)
Sydney	28.67 (1.24)	0.0353 (0.0081)
Buenos Aires	28.89 (1.23)	0.0328 (0.0085)
Tokyo	29.54 (1.21)	0.0285 (0.007)

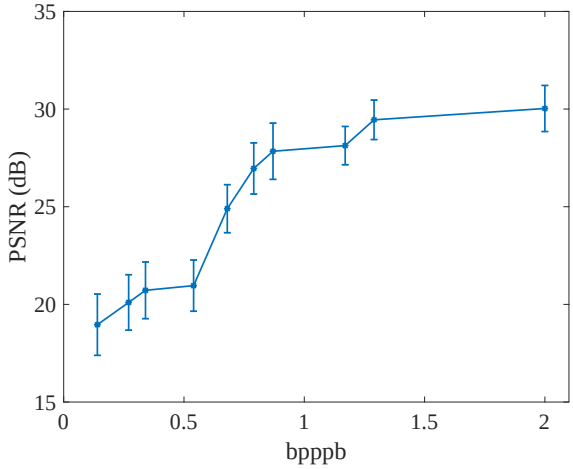


Fig. 10: Mean reconstruction error for different number of bpppb on 50 time series of multispectral images over the region of Chania.

Firstly, we applied our algorithm on different datasets using about 0.88 bpppb, with the results reported in Table IV. As we can see, the proposed compression method can efficiently decompress the times series in each case. This can also be observed from the RGB representation of an original and the corresponding decompressed multispectral image from an image time series of each region presented in Fig. 9, using the same number of bpppb. The similarity between the original and the decompressed images of each dataset demonstrates the efficacy of our method that can be applied to arbitrary high dimensional data.

Finally, we evaluated our method on different number of bpppb by using different dimensions for the core tensor and different number of quantization bits. Specifically, in Fig. 10 is presented the mean reconstruction error measured on the testing time series of multispectral images over the region of Chania. As it was expected, the more the bpppb used, the better the quality of the decompressed images. However, the reconstruction is efficient even for high compression.

D. Generalization of Tensor Decomposition Learning Method

Generalizing the proposed tensor learning method, we trained our model using 200 multispectral images acquired from the regions of Chania, Barcelona, New York, and Sydney.

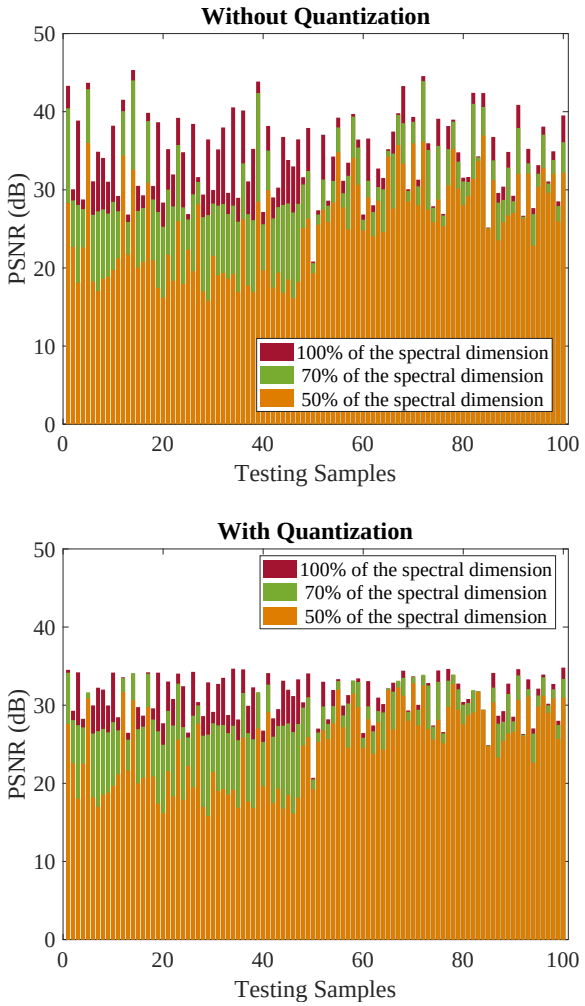


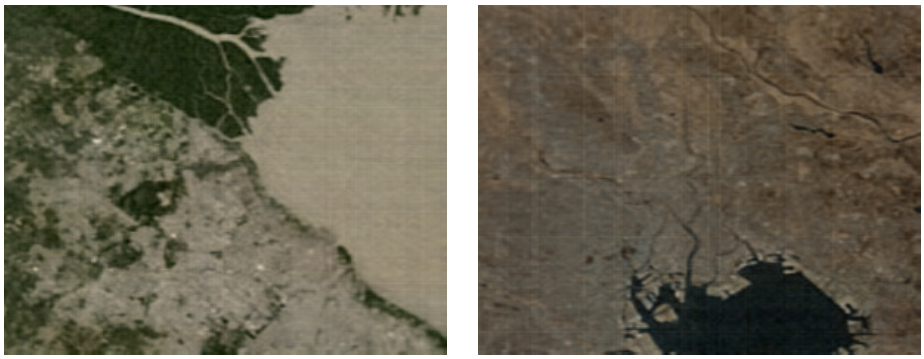
Fig. 11: The reconstruction error of the testing samples for different spectral dimensions of the core tensor as a percentage of the original dimension, before (top) and after (bottom) the quantization process.

(a) Original Image - Buenos Aires

(b) Original Image - Tokyo

(c) JPEG2000+DWT, PSNR:26.16dB

(d) JPEG2000+DWT, PSNR:23.85dB



(e) Tucker Thresholding, PSNR:27dB

(f) Tucker Thresholding, PSNR:24.39dB

(g) Our Method, PSNR:29.29dB

(h) Our Method, PSNR:27.02dB

Fig. 12: RGB representation of the original and the decompressed multispectral images from the unseen regions of Buenos Aires and Tokyo, using different compression methods and approximately 1.5 bpppb (best viewed online).

Then, we evaluated the learned basis matrices using 100 images from different regions, and specifically from Buenos Aires and Tokyo. In our experiments, we used 50 samples from each region, each of them cropped to have the same size of $227 \times 300 \times 5$.

The proposed compression method was applied on 8×8 patches with all their spectral bands, using 8 bits of quantization. Concerning the dimensions of the core tensor, we kept 50% of the original spatial dimensions, since the reconstruction is more sensitive to the spectral dimension in this scenario, as we can see from the results presented in Fig. 11. More specifically, we can observe that the quality of the decompressed images is better for a high spectral dimension of the core tensor. However, the loss from the quantization process is increasing in this case, because of the increase of the range of values of the core tensor. Despite that, the reconstruction error is small enough and the compression ratio is slightly decreasing from 0.85 to 0.75 using higher spectral dimensions for the core tensor.

Therefore, our learning method can be efficiently generalized to model satellite multispectral images from unseen regions, using the appropriate parameters. This can be verified in Fig. 12, by looking at the RGB representations of the original and the decompressed multispectral images from the testing regions of Buenos Aires and Tokyo obtained using our method as well as the state-of-the-art compression methods. As we observe, the decompressed images of the unseen regions are visually very similar to the original ones, using approximately 1.5 bpppb. At the same time, JPEG2000+DWT produces blurred outputs, while Tucker Thresholding produces artifacts in the images such as horizontal and vertical lines. Therefore, our method outperforms other competitive methods even in the case where the testing data looks significantly different from the training data.

E. Convergence

In this section, we investigate the empirical convergence of the proposed tensor decomposition learning algorithm when it is applied on real 3D and 4D remote sensing multispectral image sequences from the region of Chania.

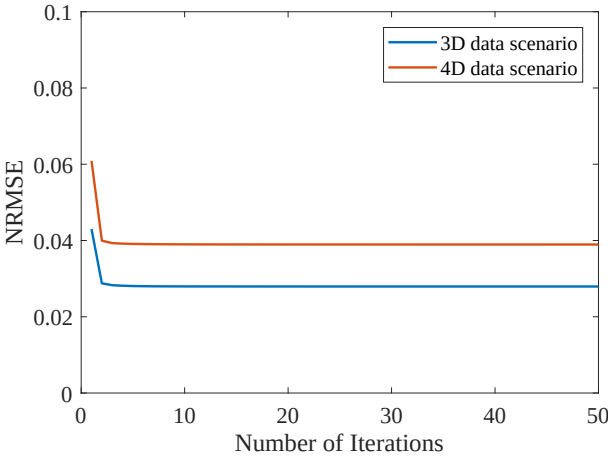


Fig. 13: Convergence behaviour of the proposed tensor decomposition learning algorithm in each real data scenario.

Specifically, in Fig. 13 is presented the reconstruction error of 50 training samples at each iteration and data scenario. As we can see, the Augmented Lagrangian function needs less than 5 iterations to converge at a stationary point in each case, while the reconstruction error is quite small from the first iteration. We can also observe that the training error is slightly increasing when the number of dimensions is increasing. Note here that the proposed algorithm was not applied on patches in this experiment.

VI. Conclusion

A novel tensor decomposition learning method is presented in this paper that learns a basis for each dimension from training samples, such that each new sample can be written as a multilinear combination of them. To achieve this, a constrained optimization problem is formulated and solved using the Alternating Direction Method of Multipliers. Based on the proposed learning method, an end-to-end compression algorithm is also presented that includes quantization and encoding, making it directly applicable in real-world applications. According to the results on synthetic data and real satellite multispectral image sequences acquired over different regions, the proposed scheme can efficiently compress arbitrary high dimensional data, achieving a better performance than other compression methods. Although we focus on the case of remote sensing observation compression, the proposed tensor decomposition learning paradigm can also be considered for other inverse imaging problems such as denoising or super-resolution, but we leave such investigations for future work.

Acknowledgment

This research work was partially supported by the Hellenic Foundation for Research and Innovation (HFRI) and the General Secretariat for Research and Technology (GSRT), under the HFRI PhD Fellowship grant no. 1509 and HFRI Faculty grant no. 1725 (V4-ICARUS), and by the CALCHAS project (contract no.842560) of the H2020 Framework Program of the European Commission.

References

[1] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[2] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[3] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE signal processing magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[4] G. Pavlidis, "Data coding and image compression," in *Mixed Raster Content*. Springer, 2017, pp. 49–211.

[5] B. Du, M. Zhang, L. Zhang, R. Hu, and D. Tao, "Pltd: Patch-based low-rank tensor decomposition for hyperspectral images," *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 67–79, 2016.

[6] M. Zhang, B. Du, L. Zhang, and X. Li, "A low-rank tensor decomposition based hyperspectral image compression algorithm," in *Pacific Rim Conference on Multimedia*. Springer, 2016, pp. 141–149.

[7] D. Wang, J. Zhou, K. He, C. Liu, and J. Xia, "Using tucker decomposition to compress color images," in *2009 2nd International Congress on Image and Signal Processing*. IEEE, 2009, pp. 1–5.

[8] L. Zhang, L. Zhang, D. Tao, X. Huang, and B. Du, "Compression of hyperspectral remote sensing images by tensor approach," *Neurocomputing*, vol. 147, pp. 358–363, 2015.

[9] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, "Tthresh: Tensor compression for multidimensional visual data," *IEEE transactions on visualization and computer graphics*, 2019.

[10] L. Fang, N. He, and H. Lin, "Cp tensor-based compression of hyperspectral images," *JOSA A*, vol. 34, no. 2, pp. 252–258, 2017.

[11] M. Cheng, L. Jing, and M. K. Ng, "Tensor-based low-dimensional representation learning for multi-view clustering," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2399–2414, 2018.

[12] Y. Wu, L. Fang, and S. Li, "Weighted tensor rank-1 decomposition for nonlocal image denoising," *IEEE Transactions on Image Processing*, vol. 28, no. 6, pp. 2719–2730, 2018.

[13] B. Penna, T. Tillo, E. Magli, and G. Olmo, "Transform coding techniques for lossy hyperspectral data compression," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, pp. 1408–1421, 2007.

[14] L. R. Tucker, "Implications of factor analysis of three-way matrices for measurement of change," *Problems in measuring change*, vol. 15, pp. 122–137, 1963.

[15] A. Kaarna and J. Parkkinen, "Transform based lossy compression of multispectral images," *Pattern Analysis & Applications*, vol. 4, no. 1, pp. 39–50, 2001.

[16] Y. Jiao, Q. Jin, X. Lu, and W. Wang, "Alternating direction method of multipliers for linear inverse problems," *SIAM Journal on Numerical Analysis*, vol. 54, no. 4, pp. 2114–2137, 2016.

[17] H. ZainEldin, M. A. Elhosseini, and H. A. Ali, "Image compression algorithms in wireless multimedia sensor networks: A survey," *Ain Shams engineering journal*, vol. 6, no. 2, pp. 481–490, 2015.

[18] M. Weeks and M. A. Bayoumi, "Three-dimensional discrete wavelet transform architectures," *IEEE Transactions on Signal Processing*, vol. 50, no. 8, pp. 2050–2063, 2002.

[19] E. Christophe, C. Mailhes, and P. Duhamel, "Hyperspectral image compression: adapting spiht and ezw to anisotropic 3-d wavelet coding," *IEEE Transactions on Image Processing*, vol. 17, no. 12, pp. 2334–2346, 2008.

[20] A. Aggoun, "Compression of 3d integral images using 3d wavelet transform," *Journal of Display Technology*, vol. 7, no. 11, pp. 586–592, 2011.

[21] M. M. H. Chowdhury and A. Khatun, "Image compression using discrete wavelet transform," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 4, p. 327, 2012.

[22] L. Chang, C.-M. Cheng, and T.-C. Chen, "An efficient adaptive klt for multispectral image compression," in *4th IEEE Southwest Symposium on Image Analysis and Interpretation*. IEEE, 2000, pp. 252–255.

[23] L. Liu, J. Yan, X. Zheng, H. Peng, D. Guo, and X. Qu, "Karhunen-loève transform for compressive sampling hyperspectral images," *Optical Engineering*, vol. 54, no. 1, p. 014106, 2015.

[24] Q. Du and J. E. Fowler, "Hyperspectral image compression using jpeg2000 and principal component analysis," *IEEE Geoscience and Remote sensing letters*, vol. 4, no. 2, pp. 201–205, 2007.

[25] B. Penna, T. Tillo, E. Magli, and G. Olmo, "Progressive 3-d coding of hyperspectral images based on jpeg 2000," *IEEE Geoscience and remote sensing letters*, vol. 3, no. 1, pp. 125–129, 2006.

[26] T.-J. Chen, S.-C. Lin, Y.-C. Lin, R.-G. Cheng, L.-H. Lin, and W. Wu, "Jpeg2000 still image coding quality," *Journal of digital imaging*, vol. 26, no. 5, pp. 866–874, 2013.

[27] A. Karami, M. Yazdi, and G. Mercier, "Hyperspectral image compression based on tucker decomposition and wavelet transform," in *2011 3rd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. IEEE, 2011, pp. 1–4.

[28] ——, "Compression of hyperspectral images using discerete wavelet transform and tucker decomposition," *IEEE journal of selected topics in applied earth observations and remote sensing*, vol. 5, no. 2, pp. 444–450, 2012.

[29] J. Li, F. Xing, and Z. You, "Compression of multispectral images with comparatively few bands using posttransform tucker decomposition," *Mathematical Problems in Engineering*, vol. 2014, Aug 2014.

[30] K. Rajan and V. Murugesan, "Hyperspectral image compression based on dwt and td with als method." *International Arab Journal of Information Technology (IAJIT)*, vol. 13, no. 4, 2016.

[31] J. Li and Z. Liu, "Compression of hyper-spectral images using an accelerated nonnegative tensor decomposition," *Open Physics*, vol. 15, no. 1, pp. 992–996, 2017.

[32] S. Hassanzadeh and A. Karami, "Compression and noise reduction of hyperspectral images using non-negative tensor decomposition and compressed sensing," *European Journal of Remote Sensing*, vol. 49, no. 1, pp. 587–598, 2016.

[33] R. Ballester-Ripoll and R. Pajarola, "Lossy volume compression using tucker truncation and thresholding," *The Visual Computer*, vol. 32, no. 11, pp. 1433–1446, 2016.

[34] M. A. Veganzones, J. E. Cohen, R. Cabral Farias, J. Chanussot, and P. Comon, "Compression-based nonnegative tensor CP decomposition of hyperspectral big data," Jan. 2015.

[35] A. Aidini, G. Tsagkatakis, and P. Tsakalides, "Compression of high-dimensional multispectral image time series using tensor decomposition learning," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[36] K. S. Gurumoorthy, A. Rajwade, A. Banerjee, and A. Rangarajan, "A method for compact image representation using sparse matrix and tensor projections onto exemplar orthonormal bases," *IEEE Transactions on Image Processing*, vol. 19, no. 2, pp. 322–334, 2009.

[37] A. Aidini, G. Tsagkatakis, and P. Tsakalides, "Tensor dictionary learning with representation quantization for remote sensing observation compression," in *2020 Data Compression Conference (DCC)*. IEEE, 2020.

[38] G. Tsagkatakis, K. Fotiadou, M. Giannopoulos, A. Aidini, A. Panousopoulou, and P. Tsakalides, "Matrix and tensor signal modelling in cyber physical systems," *Smart Water Grids: A Cyber-Physical Systems Approach*, p. 107, 2018.

[39] T. G. Kolda, "Multilinear operators for higher-order decompositions." Sandia National Laboratories, Tech. Rep., 2006.

[40] J. Håstad, "Tensor rank is np-complete," in *International Colloquium on Automata, Languages, and Programming*. Springer, 1989, pp. 451–460.