

# Matrix Product State for Feature Extraction of Higher-Order Tensors

Johann A. Bengua<sup>1</sup>, Ho N. Phien<sup>1</sup>, Hoang D. Tuan<sup>1</sup> and Minh N. Do<sup>2</sup>

**Abstract**—This paper introduces matrix product state (MPS) decomposition as a computational tool for extracting features of multidimensional data represented by higher-order tensors. Regardless of tensor order, MPS extracts its relevant features to the so-called *core tensor* of maximum order three which can be used for classification. Mainly based on a successive sequence of singular value decompositions (SVD), MPS is quite simple to implement without any recursive procedure needed for optimizing local tensors. Thus, it leads to substantial computational savings compared to other tensor feature extraction methods such as higher-order orthogonal iteration (HOOI) underlying the Tucker decomposition (TD). Benchmark results show that MPS can reduce significantly the feature space of data while achieving better classification performance compared to HOOI.

**Index Terms**—Higher-order tensor, tensor feature extraction, supervised learning, tensor classification, matrix product state (MPS), core tensor, dimensionality reduction, Tucker decomposition (TD).

## I. INTRODUCTION

THERE is an increasing need to handle large multidimensional datasets that cannot efficiently be analyzed or processed using modern day computers. Due to the curse of dimensionality it is urgent to investigate mathematical tools which can evaluate information beyond the properties of large matrices [1]. The essential goal is to reduce the dimensionality of multidimensional data, represented by tensors, with a minimal information loss by mapping the original tensor space to a lower-dimensional tensor space through tensor-to-tensor or tensor-to-vector projection [1]. The most natural option for these kinds of projection is to utilize an appropriate tensor decomposition [2] to represent the original tensor in terms of a combination of possibly lower-order tensors.

A popular method for tensor decomposition is the Tucker decomposition (TD) [3], also known as higher-order singular value decomposition (HOSVD) when orthogonality constraints are imposed [4]. As a tensor-to-tensor projection, it is an important tool for solving problems related to feature extraction, feature selection and classification of large-scale multidimensional datasets in various research fields. Its well-known application in computer vision was introduced in [5] to analyze some ensembles of facial images represented by fifth-order tensors. In data mining, the HOSVD was also applied to identify handwritten digits [6]. In addition, the HOSVD has been applied in neuroscience, pattern analysis, image classification and signal processing [7]–[9]. The central concept of using the TD is to decompose a large multidimensional tensor into a set of *common factor* matrices and a single *core tensor* which is considered as reduced features of the original tensor in spite of its lower dimension [7]. In practice, the TD is often performed in conjunction with some constraints, e.g. nonnegativity, orthogonality, etc., imposed on the common factors in order to obtain a better feature core tensor [7]. However, constraints like orthogonality often leads to an NP-hard computational problem [10]. Practical application of the TD is normally limited to small-order tensors. This is due to the fact that the TD core tensor preserves the higher-order structure of the original tensor, with its dimensionality remaining fairly large in order to capture relevant interactions between components of the tensor [2].

The higher-order orthogonal iteration (HOOI) [11] is an alternating least squares (ALS) for finding TD approximation of a tensor. Its application to independent component analysis (ICA) and simultaneous matrix diagonalization was investigated in [12]. Another TD-based method is multilinear principal component analysis (MPCA) [10], an extension of classical principal component analysis (PCA), which is closely related to HOOI. The motivation behind MPCA is that generally PCA takes vectors as inputs for dimensionality reduction, hence tensor data would need to be vectorized and this can result in large computational and memory require-

<sup>1</sup>Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia; Email: johann.a.bengua@student.uts.edu.au, ngocphien.ho@uts.edu.au, tuan.hoang@uts.edu.au.

<sup>2</sup>Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA; Email: minhdo@illinois.edu

ments, even for low order data.

The matrix product state (MPS) decomposition [13]–[16] is a tensor-to-tensor projection that has been proposed and applied to study quantum many-body systems with great success, prior to its introduction to the mathematics community under the name tensor-train (TT) decomposition [17], however, to the best of our knowledge its application to machine learning and pattern analysis has not been proposed. The MPS decomposition is fundamentally different from the TD in terms of its geometric structure as it is made up of local component tensors with maximum order three. Consequently, applying the MPS decomposition to large higher-order tensors can potentially avoid the computational bottleneck of the TD and related algorithms.

Motivated by both the TD and MPS decompositions, we propose to use MPS as a dimensionality reduction technique that consists of low-order common factors and a low-order core tensor. Specifically, MPS decomposes a higher-order tensor in such a way that its MPS representation is expressed in a *mixed-canonical form* [18]. In this form, a unique core tensor can be extracted and is naturally described by an orthogonal space spanned by the common factors. This new approach provides a unique and efficient way of feature extraction applied to tensor classification problem. Specifically, in the tensor classification problem it is applied to firstly extract the core tensor and common factors for the training set. Then the core tensor of test set is extracted by means of common factors. Once the core tensors of both training and test sets are acquired, they can be used for classifiers such as K-nearest neighbors (KNN) and linear discriminant analysis (LDA).

When compared to HOOI, MPS is not only simpler to implement but also more effective in terms of computational savings, feature space and classification success rate (CSR). This is due to the fact that MPS can obtain orthogonal common factors based on successive SVDs without needing any recursive local optimization procedure. We use supervised learning (classification) problems to benchmark MPS and compare its performance with HOOI. The datasets include the Columbia object image libraries 100 (COIL-100) [19], [20], the brain-computer imagery (BCI) dataset from Shanghai Jiao Tong University [21], the extended Yale Face Database B (EYFB) from the Computer Vision Laboratory of the University of California San Diego [22]. Experimental results show that in most cases, MPS provides better CSR compared to HOOI.

The rest of the paper is structured as follows. Section II introduces mathematical notation and preliminaries used in the paper. It then formulates the tensor classification

problem and describes how to solve it utilizing the TD. Section III describes in detail how to apply the concept of MPS to tensor classification problem, and subsequently proposes the idea of the MPS core tensor and common factors. MPS is then described in detail, followed by computational complexity analysis. In Section IV, experimental results are shown to compare MPS to HOOI. The conclusions are given in Section V.

## II. TENSOR CLASSIFICATION

To make the paper self-contained we introduce some notations and preliminaries of multilinear algebra [2]. A *tensor* is a multidimensional array and its *order* (also known as way or mode) is the number of dimensions it contains. Zero-order tensors are scalars and denoted by lowercase letters, e.g.,  $x$ . A first-order tensor is a vector and denoted by boldface lowercase letters, e.g.,  $\mathbf{x}$ . A matrix is a second order tensor and denoted by boldface capital letters, e.g.,  $\mathbf{X}$ . A higher-order tensor (tensors of order three and above) are denoted by boldface calligraphic letters, e.g.,  $\mathcal{X}$ . Generally, an  $N$ th-order tensor is denoted as  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ , where each  $I_i$  is the dimension of the local subspace  $i$ . We also denote  $x_i$  as the  $i$ th entry of a vector  $\mathbf{x}$  and  $x_{ij}$  as an element of a matrix  $\mathbf{X}$ . Generally, an element of an  $N$ th-order tensor  $\mathcal{X}$  is denoted as  $x_{i_1 \dots i_N}$ .

A mode- $n$  fiber of a tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$  is defined by fixing all indices but  $i_n$  and denoted by  $\mathbf{x}_{i_1 \dots i_{n-1} i_{n+1} \dots i_N}$ .

Mode- $n$  matricization (also known as mode- $n$  unfolding or flattening) of a tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$  is the process of unfolding or reshaping the tensor into a matrix  $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \cdots I_{n-1} I_{n+1} \cdots I_N)}$  by rearranging the mode- $n$  fibers to be the columns of the resulting matrix. Tensor element  $(i_1, \dots, i_{n-1}, i_n, i_{n+1}, \dots, i_N)$  maps to matrix element  $(i_n, j)$  such that

$$j = 1 + \sum_{k=1, k \neq n}^N (i_k - 1) J_k \quad \text{with} \quad J_k = \prod_{m=1, m \neq n}^{k-1} I_m. \quad (1)$$

The mode- $n$  product of a tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$  with a matrix  $\mathbf{A} \in \mathbb{R}^{J_n \times I_n}$  results into a new tensor of size  $I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N$  which is denoted as  $\mathcal{X} \times_n \mathbf{A}$ . Elementwise, it is described by

$$(\mathcal{X} \times_n \mathbf{A})_{i_1 \dots i_{n-1} j_n i_{n+1} \dots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \dots i_n} a_{j_n i_n}. \quad (2)$$

The inner product of two tensors  $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$  is defined as

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \dots i_N} y_{i_1 i_2 \dots i_N}. \quad (3)$$

Accordingly, the Frobenius norm of  $\mathcal{X}$  is  $\|\mathcal{X}\|_F = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}$ .

Having sufficient notations and preliminaries of multilinear algebra, we are considering the following tensor classification problem:

**Tensor classification problem.** *Given a set of  $K$  training samples represented by  $N$ th-order tensors  $\mathcal{X}^{(k)} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  ( $k = 1, 2, \dots, K$ ) corresponding to  $D$  categories, and a set of  $L$  test data  $\mathcal{Y}^{(\ell)} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  ( $\ell = 1, 2, \dots, L$ ), classify the test data into the categories  $D$  with high accuracy.*

The problem is usually addressed by the following steps:

- *Step 1:* Apply tensor decomposition method to the training set find a set of common factors and corresponding reduced features of each training sample  $\mathcal{X}^{(k)}$ .
- *Step 2:* Extract the reduced features of each test sample  $\mathcal{Y}^{(\ell)}$  in the test set using the common factors in *Step 1*.
- *Step 3:* Perform classification based on the reduced features of training and test sets using conventional methods [23] such as K-nearest neighbors (KNN) and linear discriminant analysis (LDA).

For *Step 1*, the authors in [7] proposed methods based on the TD to obtain the common factors and the core tensor from the training set. More specifically, the  $K$  training sample tensors are firstly concatenated along the mode  $(N+1)$  so that the training set is represented by an  $(N+1)$ th-order tensor  $\mathcal{X}$  defined as

$$\mathcal{X} = [\mathcal{X}^{(1)} \mathcal{X}^{(2)} \dots \mathcal{X}^{(K)}] \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N \times K}. \quad (4)$$

TD-based method such as HOOI [11] is then applied to have the approximation

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \dots \times_N \mathbf{A}^{(N)}, \quad (5)$$

where each matrix  $\mathbf{A}^{(j)} = [\mathbf{a}_1^{(j)}, \mathbf{a}_2^{(j)}, \dots, \mathbf{a}_{\Delta_j}^{(j)}] \in \mathbb{R}^{I_j \times \Delta_j}$  ( $j = 1, 2, \dots, N$ ) is orthogonal, i.e.  $\mathbf{A}^{(j)T} \mathbf{A}^{(j)} = \mathbf{I}$  ( $\mathbf{I} \in \mathbb{R}^{\Delta_j \times \Delta_j}$  denotes the identity matrix). It is called by a *common factor* matrix and can be thought of as the principal components in each mode  $j$ . The parameters  $\Delta_j$  satisfying

$$\Delta_j \leq \text{rank}(\mathbf{X}_{(j)}) \quad (6)$$

are referred to as the bond dimensions or compression ranks of the TD. The  $(N+1)$ th-order tensor

$$\mathcal{G} \in \mathbb{R}^{\Delta_1 \times \Delta_2 \times \dots \times \Delta_N \times K}$$

is called the *core tensor*, which contains reduced features of the training samples, is represented in the subspace

spanned by the common factors  $\mathbf{A}^{(j)}$ . More specifically, if  $\mathcal{G}$  is matricized such that  $\mathbf{G}_{(N+1)} \in \mathbb{R}^{K \times (\Delta_1 \Delta_2 \dots \Delta_N)}$ , each row  $k$  of  $\mathbf{G}$  represents

$$N_f = \prod_{j=1}^N \Delta_j \quad (7)$$

number of reduced features of the corresponding sample  $\mathcal{X}^{(k)}$  in the training set.

The core tensor  $\mathcal{G}$  and common factors  $\mathbf{A}^{(j)}$  are found as the solution of the following nonlinear least square

$$\min_{\mathcal{R}, \mathbf{U}^{(j)}} \|\mathcal{X} - \mathcal{R} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \dots \times_N \mathbf{U}^{(N)}\|_F^2 \quad (8)$$

subject to  $(\mathbf{U}^{(j)})^T \mathbf{U}^{(j)} = \mathbf{I}, j = 1, \dots, N,$

which is addressed by alternating least square (ALS) in each  $\mathbf{U}^{(j)}$  (with other  $\mathbf{U}^{(\ell)}$ ,  $\ell \neq j$  held fixed). The computation complexity per one iteration consisting of  $N$  ALS in  $\mathbf{U}^{(j)}$ ,  $j = 1, \dots, N$  is [24]

$$\mathcal{O}(K \Delta I^N + N K I \Delta^{2(N-1)} + N K \Delta^{3(N-1)}) \quad (9)$$

for

$$I_j \equiv I \quad \text{and} \quad \Delta_j \equiv \Delta, j = 1, 2, \dots, N. \quad (10)$$

After computing the core tensor and the common factors for the training set, we proceed to *Step 2* to extract the core tensor containing reduced features for the test set. Specifically, the core tensor for the test set is given by

$$\mathcal{Q} = \mathcal{Y} \times_1 (\mathbf{A}^{(1)})^T \dots \times_N (\mathbf{A}^{(N)})^T, \quad (11)$$

where the test set is defined as

$$\mathcal{Y} = [\mathcal{Y}^{(1)} \mathcal{Y}^{(2)} \dots \mathcal{Y}^{(L)}] \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N \times L}, \quad (12)$$

and  $\mathcal{Q} \in \mathbb{R}^{\Delta_1 \times \Delta_2 \times \dots \times \Delta_N \times L}$ . Again, the test core tensor  $\mathcal{Q}$  can be matricized such that  $\mathbf{Q}_{(N+1)} \in \mathbb{R}^{L \times (\Delta_1 \Delta_2 \dots \Delta_N)}$ , each row  $l$  of  $\mathbf{Q}$  represents  $N_f = \prod_{j=1}^N \Delta_j$  number of reduced features of the corresponding sample  $\mathcal{Y}^{(l)}$  in the test set. Finally,  $\mathbf{G}_{(N+1)}$  and  $\mathbf{Q}_{(N+1)}$  can be used for performing classification according to *Step 3*.

Although the core tensors  $\mathcal{G}$  and  $\mathcal{Q}$  can be used for direct training and classification in *Step 3*, their dimensionality often remain large. This is due to the fact that they retain the same orders of their own original tensors. Thus, it may require a further dimensionality reduction of the core tensors using techniques such as Fisher score ranking before inputting them into classifiers to improve the classification accuracy [7]. Besides, the computational complexity of HOOI to obtain (5) is high as (9) for the computational complexity per one iteration shows, which may become prohibitive when the order of the training tensor is large. In addition, due to the unbalanced

single-mode matricization (one mode versus the rest) of the tensor when using HOOI, it may not be capable of capturing the mutual correlation between modes of the tensor which usually needs multimode matricization (a few modes versus the rest). Hence, it might cause loss of important information for classification while decomposing the tensor. To circumvent these issues, we propose to use the MPS decomposition in the next section.

### III. MATRIX PRODUCT STATE DECOMPOSITION FOR TENSOR FEATURE EXTRACTION

In this section we develop a tensor feature extraction method based on MPS decomposition as an alternative solution to the above stated tensor classification problem. Subsection III-A presents the concept of common factors and a core tensor for the MPS decomposition of tensor feature extraction and classification problems. Then the MPS is proposed in Subsection III-B. We finally analyse the computational complexity of MPS in comparison with HOOI in Subsection III-C.

#### A. Common factors and core tensor of matrix product state decomposition

We now introduce the concept of core tensor and common factors of an MPS representing the training tensor  $\mathcal{X}$  in (4). Without loss of generality, let us opt to *permute* the mode  $K$  of the tensor  $\mathcal{X}$  such that

$$\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times K \times I_n \times \dots \times I_N}$$

(the mode  $K$  is now located at  $n$  which can be chosen arbitrarily but conventionally we choose it at the middle of the chain, say  $n = \text{round}(N/2)$ ). Accordingly, positions of modes  $I_n, \dots, I_N$  are shifted by one to the right. Then, we present the elements of  $\mathcal{X}$  in the following *mixed-canonical form* [18] of the matrix product state (MPS) or tensor train (TT) decomposition [14]–[17]:

$$x_{i_1 \dots k \dots i_N} = \mathbf{B}_{i_1}^{(1)} \dots \mathbf{B}_{i_{n-1}}^{(n-1)} \mathbf{G}_k^{(n)} \mathbf{C}_{i_n}^{(n+1)} \dots \mathbf{C}_{i_N}^{(N+1)}, \quad (13)$$

where  $\mathbf{B}_{i_j}^{(j)}$  and  $\mathbf{C}_{i_{(j-1)}}^{(j)}$  (the upper index “ $(j)$ ” denotes the position  $j$  of the matrix in the chain) of dimension  $\Delta_{(j-1)} \times \Delta_j$  ( $\Delta_0 = \Delta_{N+1} = 1$ ), are called “left” and “right” *common factors* which satisfy the following orthogonality conditions:

$$\sum_{i_j} (\mathbf{B}_{i_j}^{(j)})^T \mathbf{B}_{i_j}^{(j)} = \mathbf{I}, \quad (j = 1, \dots, n-1) \quad (14)$$

and

$$\sum_{i_{(j-1)}} \mathbf{C}_{i_{(j-1)}}^{(j)} (\mathbf{C}_{i_{(j-1)}}^{(j)})^T = \mathbf{I}, \quad (j = n+1, \dots, N+1) \quad (15)$$

respectively, where  $\mathbf{I}$  denotes the identity matrix. Each  $\mathbf{G}_k^{(n)}$  for  $k = 1, 2, \dots, K$  is a matrix of dimension  $\Delta_{n-1} \times \Delta_n$  and the MPS *core tensor* is defined by

$$\mathcal{G}^{(n)} = [\mathbf{G}_1^{(n)} \mathbf{G}_2^{(n)} \dots \mathbf{G}_K^{(n)}] \in \mathbb{R}^{\Delta_{n-1} \times \Delta_n \times K}$$

which describes the reduced features of the training set. The parameters  $\Delta_j$  are called the bond dimensions or compression ranks of the MPS.

Using the common factors  $\mathbf{B}_{i_j}^{(j)}$  and  $\mathbf{C}_{i_{(j-1)}}^{(j)}$ , we can extract the core tensor for the test tensor  $\mathcal{Y}$ . Specifically, we first need to permute  $\mathcal{Y}$  defined in Eq. (12) in such a way that the index  $\ell$  is at the same position as  $k$  in the training tensor to ensure the compatibility between the training and test tensors, i.e., the permuted

$$\mathcal{Y} \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times L \times I_n \times \dots \times I_N}.$$

Then the core tensor  $\mathcal{Q}^{(n)}$  of the test tensor  $\mathcal{Y}$  is given by

$$\mathcal{Q}^{(n)} = [\mathbf{Q}_1^{(n)} \mathbf{Q}_2^{(n)} \dots \mathbf{Q}_L^{(n)}] \in \mathbb{R}^{\Delta_{n-1} \times \Delta_n \times L},$$

where

$$\mathbf{Q}_{(\ell)}^{(n)} = \sum_{i_1, \dots, i_N} (\mathbf{B}_{i_1}^{(1)})^T \dots (\mathbf{B}_{i_{n-1}}^{(n-1)})^T y_{i_1 \dots \ell \dots i_N} (\mathbf{C}_{i_n}^{(n+1)})^T \dots (\mathbf{C}_{i_N}^{(N+1)})^T. \quad (16)$$

Note that as the core tensors  $\mathcal{G}^{(n)}$  and  $\mathcal{Q}^{(n)}$  of both training and test tensors are extracted by using the same common factors, they are represented by the same base. Thus, they can be used for the classification directly for *Step 3*. More precisely, we can matricize  $\mathcal{G}^{(n)}$  and  $\mathcal{Q}^{(n)}$  to

$$\mathbf{G}^{(n)} \in \mathbb{R}^{K \times (\Delta_{n-1} \Delta_n)}$$

and

$$\mathbf{Q}^{(n)} \in \mathbb{R}^{L \times (\Delta_{n-1} \Delta_n)}$$

such that each of their rows, either  $\mathbf{G}^{(n)}$  or  $\mathbf{Q}^{(n)}$  is a sample containing

$$N_f = \Delta_{n-1} \Delta_n \quad (17)$$

number of reduced features of the original sample in either training or test set, respectively.

In the next section we will show that Eq. (13) can be implemented straightforwardly without any recursive local optimization procedure like ALS in HOOI required. This results into substantial computational savings. Thus, the MPS can overcome the aforementioned issues of HOOI.

### B. Matrix product state for feature extraction

We describe the MPS method for computing the core tensor and common factors of the training set. More specifically, we show how to decompose the training tensor  $\mathcal{X}$  into the MPS according to Eq. (13). To this end, we apply two successive sequences of SVDs to the tensor which include left-to-right sweep for computing the left common factors  $\mathbf{B}_{i_1}^{(1)}, \dots, \mathbf{B}_{i_{n-1}}^{(n-1)}$  and right-to-left sweep for computing the right common factors  $\mathbf{C}_{i_n}^{(n+1)}, \dots, \mathbf{C}_{i_N}^{(N+1)}$  and core tensor  $\mathcal{G}^{(n)}$  in Eq. (13) explained in the following [18]:

- *Left-to-right sweep for left factor computation:*

The left-to-right sweep involves acquiring matrices  $\mathbf{B}_{i_j}^{(j)}$  ( $i_j = 1, \dots, I_j$ , where  $j = 1, \dots, n-1$ ) fulfilling orthogonality condition in Eq. (14). Let us start by performing the mode-1 matricization of  $\mathcal{X}$  to obtain the matrix

$$\mathbf{W} \in \mathbb{R}^{I_1 \times (I_2 \cdots K \cdots I_N)}.$$

Then applying the SVD to  $\mathbf{W}$  such that

$$\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T.$$

We then define the first common factors  $\mathbf{B}_{i_1}^{(1)} = \mathbf{U}_{i_1} \in \mathbb{R}^{1 \times \Delta_1}$ , where  $\Delta_1 \leq \text{rank}(\mathbf{X}_{(1)})$ , satisfying the left-canonical constraint in Eq. (14) due to the SVD. In order to find the next common factors  $\mathbf{B}_{i_2}^{(2)}$  we firstly form the matrix

$$\mathbf{W} = \mathbf{S}\mathbf{V}^T \in \mathbb{R}^{\Delta_1 \times (I_2 \cdots K \cdots I_N)}.$$

The matrix  $\mathbf{W}$  is then reshaped to

$$\mathbf{W} \in \mathbb{R}^{(\Delta_1 I_2) \times (I_3 \cdots K \cdots I_N)}$$

and its SVD is given by

$$\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T.$$

Reshape the matrix

$$\mathbf{U} \in \mathbb{R}^{(\Delta_1 I_2) \times \Delta_2} \quad (\Delta_2 \leq \text{rank}(\mathbf{W}))$$

into a third-order tensor

$$\mathcal{U} \in \mathbb{R}^{\Delta_1 \times I_2 \times \Delta_2}$$

and we define  $\mathbf{B}_{i_2}^{(2)} = \mathbf{U}_{i_2}$  satisfying the left-canonical constraint due to the SVD. Applying a same procedure for determining  $\mathbf{B}_{i_3}^{(3)}$  by forming a new matrix

$$\mathbf{W} = \mathbf{S}\mathbf{V}^T \in \mathbb{R}^{\Delta_2 \times (I_3 \cdots K \cdots I_N)},$$

reshaping it to

$$\mathbf{W} \in \mathbb{R}^{(\Delta_2 I_3) \times (I_4 \cdots K \cdots I_N)},$$

performing SVD and so on. This procedure is iterated until we obtain all the matrices  $\mathbf{B}_{i_j}^{(j)}$  ( $i_j =$

$1, \dots, I_j$ , where  $j = 1, \dots, n-1$ ) fulfilling the left-canonical constraint in Eq. (14).

In a nutshell, after completing the left-to-right sweep elements of tensor  $\mathcal{X}$  are written in the following form:

$$x_{i_1 \cdots i_{n-1} k i_n \cdots i_{N+1}} = \mathbf{B}_{i_1}^{(1)} \cdots \mathbf{B}_{i_{n-1}}^{(n-1)} \mathbf{W}_{(k i_n \cdots i_N)}, \quad (18)$$

where the matrix  $\mathbf{W}$  is reshaped to the matrix form  $\mathbf{W} \in \mathbb{R}^{(\Delta_{n-1} K \cdots I_{N-1}) \times I_N}$  for the next right-to-left sweep process.

- *Right-to-left sweep for right factor computation:*

Similar to left-to-right sweep, we perform a sequence of SVDs starting from the right to the left of the MPS to get the matrices  $\mathbf{C}_{i_{(j-1)}}^{(j)}$  ( $i_{(j-1)} = 1, \dots, I_{(j-1)}$ , where  $j = n+1, \dots, N+1$ ) fulfilling the right-canonical condition in Eq. (15). To start, we apply the SVD to the matrix  $\mathbf{W}$  obtained previously in the left-to-right sweep such that

$$\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T.$$

Let us then define

$$\mathbf{C}_{i_N}^{(N+1)} = \mathbf{V}_{i_N}^T \in \mathbb{R}^{\Delta_N \times 1},$$

where  $\Delta_N \leq \text{rank}(\mathbf{W})$ , which satisfies the right-canonical constraint (Eq. (15)) due to the SVD. Next, multiply  $\mathbf{U}$  and  $\mathbf{S}$  together and reshape the resulting matrix into

$$\mathbf{W} \in \mathbb{R}^{(\Delta_{n-1} K \cdots I_{N-2}) \times (I_{N-1} \Delta_N)}.$$

Again, applying the SVD to the matrix  $\mathbf{W}$ , we have

$$\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T.$$

Reshape the matrix

$$\mathbf{V}^T \in \mathbb{R}^{\Delta_{N-1} \times (I_{N-1} \Delta_N)},$$

where  $\Delta_{N-1} \leq \text{rank}(\mathbf{W})$ , into a third-order tensor

$$\mathcal{V} \in \mathbb{R}^{\Delta_{N-1} \times I_{N-1} \times \Delta_N}$$

and we define the next common factor  $\mathbf{C}_{i_{(N-1)}}^{(N)} = \mathbf{V}_{i_{(N-1)}}$  satisfying Eq. (15). We need to obtain the matrix  $\mathbf{W}$  by multiplying  $\mathbf{U}$  and  $\mathbf{S}$  together for determining the next common factor, i.e.  $\mathbf{C}_{i_{N-2}}^{(N-1)}$ . This procedure is iterated until all the common factors  $\mathbf{C}_{i_{(j-1)}}^{(j)}$  ( $i_{(j-1)} = 1, \dots, I_{(j-1)}$ , where  $j = n+1, \dots, N+1$ ) are acquired. In the end, we obtain Eq. (13) for MPS decomposition of the tensor  $\mathcal{X}$  where the core tensor

$$\mathcal{G}^{(n)} \in \mathbb{R}^{\Delta_{n-1} \times \Delta_n \times K}$$

is determined by reshaping the matrix

$$\mathbf{G}^{(n)} = \mathbf{U}\mathbf{S} \in \mathbb{R}^{\Delta_{n-1} \times (K \Delta_n)}.$$

Having done this, we can substitute the common factors into Eq. (16) to extract the core tensor for the test tensor.

Note that the MPS decomposition described by Eq. (13) can be performed exactly or approximately depending on the bond dimensions  $\Delta_j$  ( $j = 1, \dots, N$ ) which have the following bound [17]:

$$\Delta_j \leq \text{rank}(\mathbf{W}) \leq \text{rank}(\mathbf{X}_{[j]}), \quad (19)$$

versus their counterpart (6) in HOOI, where  $\text{rank}(\mathbf{X}_{[j]})$  denotes the rank of the matrix  $\mathbf{X}_{[j]}$  of size  $(I_1 I_2 \cdots I_j) \times (I_{j+1} \cdots K \cdots I_N)$  which is the *mode*-(1, 2, ...,  $j$ ) *matricization* of the tensor  $\mathcal{X}$ . In practice, each bond dimension  $\Delta_j$  is usually truncated to be smaller than  $\text{rank}(\mathbf{W})$  on every SVD of  $\mathbf{W}$  leading to an efficient MPS decomposition. To this end, we rely on thresholding the singular values of  $\mathbf{W}$ . For instance, applying SVD to the matrix  $\mathbf{W} \in \mathbb{R}^{I \times J}$  (let us assume  $I \leq J$ ), we have  $\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ , where  $\mathbf{S} = \text{diag}(s_1, s_2, \dots, s_I)$  are the nonvanishing singular values. With a threshold  $\epsilon$  being defined in advance, we truncate the bond dimension by keeping only  $\Delta$  singular values such that

$$\frac{\sum_{j=1}^{\Delta} s_j}{\sum_{j=1}^I s_j} \geq \epsilon. \quad (20)$$

Having done this, we have  $\mathbf{W} \approx \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T$ , where  $\tilde{\mathbf{U}} \in \mathbb{R}^{I \times \Delta}$ ,  $\tilde{\mathbf{S}} \in \mathbb{R}^{\Delta \times \Delta}$  and  $\tilde{\mathbf{V}}^T \in \mathbb{R}^{\Delta \times J}$ . Note that the larger the  $\epsilon$  the more accurate MPS decomposition of the tensor  $\mathcal{X}$  but less efficient in reducing the dimensionality of the tensor. Therefore, one needs to choose an appropriate value for  $\epsilon$  via empirical simulations. A summary of applying MPS decomposition for tensor feature extraction can be found in Table I.

### C. Complexity analysis

For a given training tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times K}$  under the assumption (10), the TD and MPS representation of  $\mathcal{X}$  consists of

$$NI\Delta + K\Delta^N$$

and

$$(N-2)I\Delta^2 + K\Delta^2 + 2I\Delta$$

parameters, respectively. The dominant computational complexity of MPS is  $\mathcal{O}(KI^{(N+1)})$  due to the first SVD of the matrix obtained from mode-1 matricization of  $\mathcal{X}$ . On the other hand, the computational complexity of HOOI is (9) per iterations with unknown iteration number to attained the convergence of ALS rounds. In addition, it usually employs the HOSVD to initialize the tensors which involves the cost of order  $\mathcal{O}(NKI^{N+1})$ , and thus very expensive with large  $N$  compared to MPS.

TABLE I: Matrix product state for tensor feature extraction

<b>Input:</b>	$\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times K \times \cdots \times I_N}$ , $\epsilon$ : SVD threshold
<b>Output:</b>	$\mathcal{G}^{(n)} \in \mathbb{R}^{\Delta_{n-1} \times \Delta_n \times K}$ , $\mathbf{B}_{i_j}^{(j)}$ ( $i_j = 1, \dots, I_j, j = 1, \dots, n-1$ ) $\mathbf{C}_{i_{(j-1)}}^{(j)}$ ( $i_{(j-1)} = 1, \dots, I_{(j-1)}, j = n+1, \dots, N+1$ )
1:	Set $\mathbf{W} = \mathbf{X}_{(1)}$ % Mode-1 matricization of $\mathcal{X}$
2:	<b>for</b> $j = 1$ <b>to</b> $n-1$ % Left-to-right sweep
3:	$\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ % SVD of $\mathbf{W}$
4:	$\mathbf{W} \approx \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T$ % Thresholding $\mathbf{S}$ using Eq. (20)
5:	$\mathbf{B}_{i_j}^{(j)} = \tilde{\mathbf{U}}_{i_j}$ % Set common factors
6:	$\mathbf{W} = \tilde{\mathbf{S}}\tilde{\mathbf{V}}^T$ % Construct new matrix $\mathbf{W}$
7:	<b>end</b>
8:	Reshape $\mathbf{W} \in \mathbb{R}^{(\Delta_{n-1} K \cdots I_N) \times I_N}$
9:	<b>for</b> $j = N+1$ <b>down to</b> $n+1$ % right-to-left sweep
10:	$\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ % SVD of $\mathbf{W}$
11:	$\mathbf{W} \approx \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T$ % Thresholding $\mathbf{S}$ using Eq. (20)
12:	$\mathbf{C}_{i_{(j-1)}}^{(j)} = \tilde{\mathbf{V}}_{i_{(j-1)}}^T$ % Set common factors
13:	$\mathbf{W} = \tilde{\mathbf{U}}\tilde{\mathbf{S}}$ % Construct new matrix $\mathbf{W}$
14:	<b>end</b>
15:	Set $\mathcal{G}^{(n)} = \mathcal{W}$ % Training core tensor

Texts after symbol “%” are comments.

## IV. EXPERIMENTAL RESULTS

In this section, we apply MPS to perform feature extraction for classification problem. The method is applied to a few datasets and compared with one of the most popular feature extraction methods, i.e. HOOI. Specifically, three datasets, namely Columbia Object Image Libraries (COIL)-100 [19], [20], Extended Yale Face Database B (EYFB) [22] and BCI Jiaotong dataset (BCIJ) [21], are used to benchmark the simulation. In all simulations, we rely on the threshold  $\epsilon$  defined in Eq. (20) to adjust the bond dimensions of MPS in Eq. (13) as well as that of HOOI in Eq. (5).

The COIL-100 dataset has 7200 color images of 100 objects (72 images per object) with different reflectance and complex geometric characteristics. Each image is initially a 3rd-order tensor of dimension  $128 \times 128 \times 3$  and then is downsampled to the one of dimension  $32 \times 32 \times 3$ . The dataset is divided into training and test sets randomly consisting of  $K$  and  $L$  images, respectively according to a certain holdout (H/O) ratio  $r$ , i.e.  $r = \frac{L}{K}$ . Hence, the training and test sets are represented by four-order tensors of dimensions  $32 \times 32 \times 3 \times K$  and  $32 \times 32 \times 3 \times L$ , respectively. In Fig. 1 and Fig. 2, we show how a few objects of the training and test sets ( $r = 0.5$  is chosen), respectively, change after applying MPS and HOOI to reduce the number of features with two different values of threshold,  $\epsilon = 0.9, 0.65$ . In both training and test sets, we can see that with  $\epsilon = 0.9$ , the images are not

modified significantly due to the fact that many features are preserved. However, in the case that  $\epsilon = 0.65$ , the images are blurred. That is because less features are kept. However, we can observe that the shapes of objects are still preserved. Especially, in most cases MPS seems to preserve the color of the images better than HOOI. This is because the bond dimension corresponding to the color mode  $I_3 = 3$  has a small value, e.g.  $\Delta_3 = 1$  for  $\epsilon = 0.65$  in HOOI. This problem arises due to the the unbalanced matricization of the tensor corresponding to the color mode. Specifically, if we take a mode-3 matricization of tensor  $\mathcal{X} \in \mathbb{R}^{32 \times 32 \times 3 \times K}$ , the resulting matrix of size  $3 \times (1024K)$  is extremely unbalanced. Therefore, when taking SVD with some small threshold  $\epsilon$ , the information corresponding to this color mode may be lost due to dimension reduction. On the contrary, we can efficiently avoid this problem in MPS by permuting the tensor such that  $\mathcal{X} \in \mathbb{R}^{32 \times K \times 3 \times 32}$  before applying the tensor decomposition.

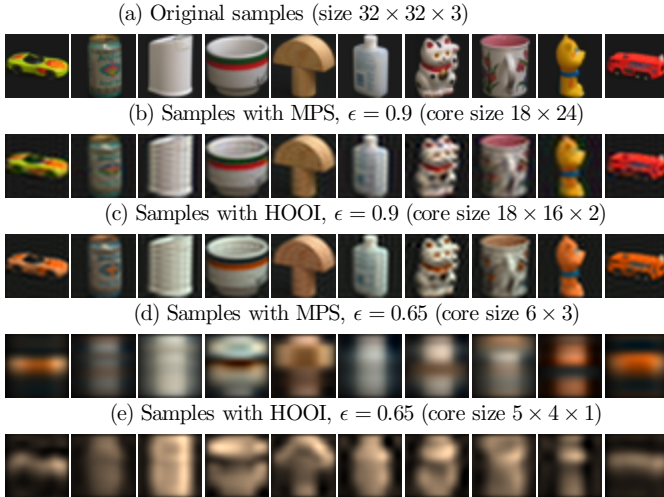


Fig. 1: Modification of a 10 objects in the **training set** of COIL-100 are shown after applying MPS and HOOI corresponding to  $\epsilon = 0.9$  and  $0.65$  to reduce the number of features of each object.

To validate MPS for classification, the core tensors with full sizes obtained from both methods are input directly to a classifier which is chosen as the K nearest neighbor with  $K=1$  (KNN-1) in our case. For each H/O ratio, the classification success rate (CSR) is averaged over 10 iterations of randomly splitting the dataset into training and test sets. Comparison of performance between MPS and HOOI is shown in Fig. 3 for four different H/O ratios, i.e.  $r = (50\%, 80\%, 90\%, 95\%)$ . In each plot, we show the CSR with respect to threshold  $\epsilon$ . We can see that MPS performs quite well when compared to HOOI. Especially, with small  $\epsilon$ , MPS performs much

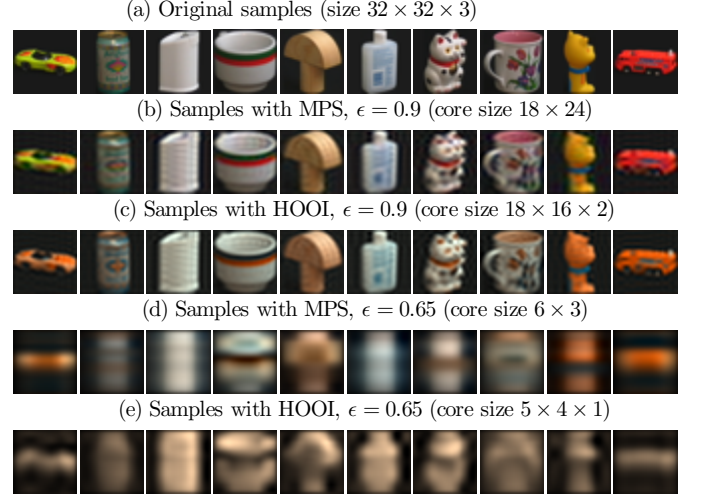


Fig. 2: Modification of 10 objects in the **test set** of COIL-100 are shown after applying MPS and HOOI corresponding to  $\epsilon = 0.9$  and  $0.65$  to reduce the number of features of each object.

better than HOOI. Besides, we also show the best CSR corresponding to each H/O ratio obtained by different methods in Table. II. It can be seen that MPS always gives better results than HOOI even in the case of small value of  $\epsilon$  and number of features  $N_f$  defined by (7) and (17) for HOOI and MPS, respectively.

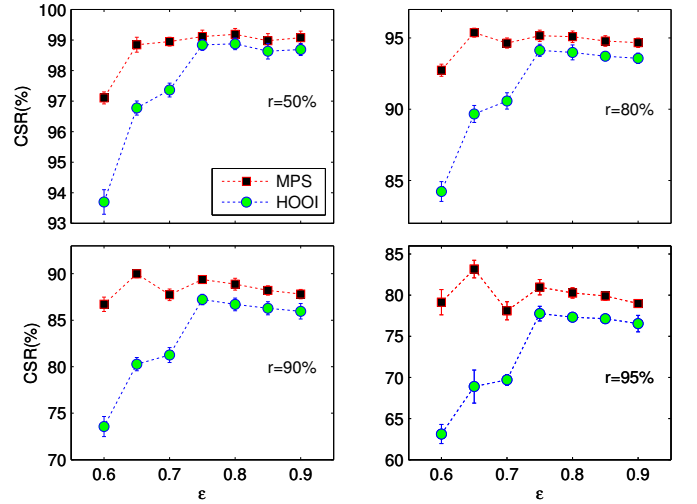


Fig. 3: Error bar plots of CSR versus thresholding rate  $\epsilon$  for different H/O ratios.

We also perform experiment on the EYFB dataset which contains 16128 grayscale images with 28 human subjects, under 9 poses, where for each pose there is 64 illumination conditions. Similar to [25], to improve computational time each image was cropped to keep only the center area containing the face, then resized

TABLE II: COIL-100 benchmark: The best CSR corresponding to different H/O ratios obtained by MPS and HOOI.

Algorithm	CSR	$N_f$	$\epsilon$	CSR	$N_f$	$\epsilon$
$r = 50\%$				$r = 85\%$		
HOOI	$98.87 \pm 0.19$	198	0.80	$94.13 \pm 0.42$	112	0.75
MPS	<b><math>99.19 \pm 0.19</math></b>	120	0.80	<b><math>95.37 \pm 0.31</math></b>	18	0.65
$r = 90\%$				$r = 95\%$		
HOOI	$87.22 \pm 0.56$	112	0.75	$77.76 \pm 0.90$	112	0.75
MPS	<b><math>89.38 \pm 0.40</math></b>	$59 \pm 5$	0.75	<b><math>83.17 \pm 1.07</math></b>	18	0.65

to  $73 \times 55$ . The training and test datasets are not selected randomly but partitioned according to poses. More precisely, the training and test datasets are selected to contain poses 0, 2, 4, 6 and 8 and 1, 3, 5, and 7, respectively. For a single subject the training tensor has size  $5 \times 73 \times 55 \times 64$  and  $4 \times 73 \times 55 \times 64$  is the size of the test tensor. Hence for all 28 subjects we have fourth-order tensors of sizes  $140 \times 73 \times 55 \times 64$  and  $112 \times 73 \times 55 \times 64$  for the training and test datasets, respectively.

We apply the MPS and HOOI to extract the core tensors before inputting them into the classifiers. However, in this experiments we realize that the size of each core tensor remains very large even with small threshold used, e.g., for  $\epsilon = 0.75$ , the core size of each sample obtained by MPS and HOOI are  $18 \times 201 = 3618$  and  $14 \times 15 \times 13 = 2730$ , respectively which is not useful for directly classifying. Therefore, we need to further reduce the sizes of core tensors before feeding them to classifiers for a better performance. In our experiment, we simply apply a further truncation to each core tensor by keeping the first few dimensions of each mode of the tensor. Intuitively, this can be done as we have already known that the space of each mode is orthogonal and ordered in such a way that the first dimension corresponds to the largest singular value, the second one corresponds to the second largest singular value and so on. Therefore, we can independently truncate the dimension of each mode to a reasonably small value (which can be determined empirically) without changing significantly the meaning of the core tensor. It then gives rise to a core tensor of smaller size that can be used directly for classification. More specifically, suppose that the core tensors obtained by MPS and HOOI have sizes  $Q \times \Delta_1 \times \Delta_2$  and  $Q \times \Delta_1 \times \Delta_2 \times \Delta_3$ , where  $Q$  is the number  $K$  ( $L$ ) of training (test) samples, respectively. The core tensors are then truncated to be  $Q \times \tilde{\Delta}_1 \times \tilde{\Delta}_2$  and  $Q \times \tilde{\Delta}_1 \times \tilde{\Delta}_2 \times \tilde{\Delta}_3$ , respectively such that  $\tilde{\Delta}_l < \Delta_l$  ( $l = 1, 2, 3$ ). Note that each  $\tilde{\Delta}_l$  is chosen to be the same for both training and test core tensors. We show the classification results for different threshold values  $\epsilon$  in Table. III using two different classifiers, i.e. KNN-1 and LDA. In this result, the core tensors

obtained by MPS and HOOI are reduced to have sizes of  $Q \times \tilde{\Delta}_1 \times \tilde{\Delta}_2$  and  $Q \times \tilde{\Delta}_1 \times \tilde{\Delta}_2 \times \tilde{\Delta}_3$ , respectively such that  $\tilde{\Delta}_1 = \tilde{\Delta}_2 = \Delta \in (10, 11, 12, 13, 14)$  and  $\tilde{\Delta}_3 = 1$ . As a result, the reduced core tensors obtained by both methods have the same size for classification. Note that each value of CSR in Table. III is computed by taking the average of the ones obtained from classifying different reduced core tensors due to different  $\Delta$ . We can see that the MPS gives rise to better results for all threshold values using different classifiers. More importantly, MPS with smallest  $\epsilon$  can also produce CSR as well as largest  $\epsilon$ . The LDA classifier gives rise to the best result, i.e.  **$97.32 \pm 0.89$** .

Lastly, we study the BCI dataset which consists of single trial recognition for BCI electroencephalogram (EEG) data involving left/right motor imagery (MI) movements. The original experiment had five subjects and the paradigm required subjects to control a cursor by imagining the movements of their right or left hand for 2 seconds with a 4 second break between trials. Subjects were required to sit and relax on a chair, looking at a computer monitor approximately 1m from the subject at eye level. For each subject, data was collected over two sessions with a 15 minute break in between. The first session contained 60 trials (30 trials for left, 30 trials for right) and were used for training. The second session consisted of 140 trials (70 trials for left, 70 trials for right). The EEG signals were sampled at 500Hz and preprocessed with a filter at 8-30Hz, hence for each subject the data consisted of a multidimensional tensor  $channel \times time \times trial$ . Prior to simulation we preprocessed the data by transforming the tensor into the time-frequency domain using complex Morlet wavelets with bandwidth parameter  $f_b = 6\text{Hz}$  (CMOR6-1) to make classification easier [26], [27]. The wavelet center frequency  $f_c = 1\text{Hz}$  is chosen. Hence, the size of the concatenated tensors are  $62 \text{ channels} \times 23 \text{ frequency bins} \times 50 \text{ time frames} \times Q$ .

We perform the experiment for subject 1 and 2 of the dataset. Similar to the case of the EYFB dataset, after applying the feature extraction methods, the core tensors still have high dimension, so we need to further reduce their sizes before using them for classification.



TABLE III: EYFB benchmark with reduced core tensors being used for classification, core sizes of MPS and HOOI are  $Q \times \Delta \times \Delta$  and  $Q \times \Delta \times \Delta \times 1$ , where  $\Delta \in (10, \dots, 14)$ , respectively.

Algorithm	CSR ( $\epsilon = 0.9$ )	CSR ( $\epsilon = 0.85$ )	CSR ( $\epsilon = 0.80$ )	CSR ( $\epsilon = 0.75$ )
<b>KNN-1</b>				
HOOI	90.71 $\pm$ 1.49	90.89 $\pm$ 1.60	91.61 $\pm$ 1.26	88.57 $\pm$ 0.80
MPS	<b>94.29 <math>\pm</math> 0.49</b>	<b>94.29 <math>\pm</math> 0.49</b>	<b>94.29 <math>\pm</math> 0.49</b>	<b>94.29 <math>\pm</math> 0.49</b>
<b>LDA</b>				
HOOI	96.07 $\pm$ 0.80	95.89 $\pm$ 0.49	96.07 $\pm$ 0.49	96.07 $\pm$ 0.49
MPS	<b>97.32 <math>\pm</math> 0.89</b>	<b>97.32 <math>\pm</math> 0.89</b>	<b>97.32 <math>\pm</math> 0.89</b>	<b>97.32 <math>\pm</math> 0.89</b>

TABLE IV: BCI Jiaotong benchmark with reduced core tensors being used for classification, core sizes of MPS and HOOI are  $Q \times 12 \times \Delta$  and  $Q \times 12 \times \Delta \times 1$ , where  $\Delta \in (8, \dots, 14)$ , respectively.

Algorithm	CSR ( $\epsilon = 0.9$ )	CSR ( $\epsilon = 0.85$ )	CSR ( $\epsilon = 0.80$ )	CSR ( $\epsilon = 0.75$ )
<b>Subject 1</b>				
HOOI	84.39 $\pm$ 1.12	83.37 $\pm$ 0.99	82.04 $\pm$ 1.05	84.80 $\pm$ 2.21
MPS	<b>87.24 <math>\pm</math> 1.20</b>	<b>87.55 <math>\pm</math> 1.48</b>	<b>87.24 <math>\pm</math> 1.39</b>	<b>87.65 <math>\pm</math> 1.58</b>
<b>Subject 2</b>				
HOOI	83.16 $\pm$ 1.74	82.35 $\pm$ 1.92	82.55 $\pm$ 1.93	79.39 $\pm$ 1.62
MPS	<b>90.10 <math>\pm</math> 1.12</b>	<b>90.10 <math>\pm</math> 1.12</b>	<b>90.00 <math>\pm</math> 1.09</b>	<b>91.02 <math>\pm</math> 0.70</b>

For instance, the reduced core sizes of MPS and HOOI are chosen to be  $Q \times 12 \times \Delta$  and  $Q \times 12 \times \Delta \times 1$ , where  $\Delta \in (8, \dots, 14)$ , respectively. For classification, we use LDA and the classification results are shown in Table. IV for different threshold values. We can see that MPS always performs better than HOOI.

## V. CONCLUSION

In this paper, we propose MPS as an alternative to TD-based algorithms for feature extraction applied to tensor classification problem. Compared to HOOI, MPS has been shown to have some advantages such as computational savings due to successive SVDs are employed and no recursive optimization needed for acquiring common factors and core tensors. In addition, using extracted features given by core tensors for classifiers is capable of leading to better classification success rate even though a same number of features is used from both MPS and HOOI. We have validated our method by applying it to classify a few multidimensional datasets, such as visual data (COIL-100 and EYFB) and EEG signals where training and test data represented by fourth-order tensors. Benchmark results show that MPS gives better classification rates than HOOI in most cases.

For the future outlook, we plan to further improve MPS for classifying very big multilinear datasets. We also plan to extend this promising tool for many other problems such as multilinear data compression and completion.

## REFERENCES

- [1] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "A survey of multilinear subspace learning for tensor data," *Pattern Recognition*, vol. 44, no. 7, pp. 1540 – 1551, 2011.
- [2] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.
- [3] L. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.
- [4] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, 2000.
- [5] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear analysis of image ensembles: Tensorfaces," *Proceedings of the 7th European Conference on Computer Vision, Lecture Notes in Comput. Sci.*, vol. 2350, pp. 447–460, 2002.
- [6] B. Savas and L. Eldén, "Handwritten digit classification using higher order singular value decomposition," *Pattern Recognition*, vol. 40, no. 3, pp. 993 – 1003, 2007.
- [7] A. H. Phan and A. Cichocki, "Tensor decompositions for feature extraction and classification of high dimensional datasets," *IEICE Nonlinear Theory and Its Applications*, vol. 1, no. 1, pp. 37–68, 2010.
- [8] L. Kuang, F. Hao, L. Yang, M. Lin, C. Luo, and G. Min, "A tensor-based approach for big data representation and dimensionality reduction," *IEEE Trans. Emerging Topics in Computing*, vol. 2, no. 3, pp. 280–291, Sept 2014.
- [9] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and A. H. 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, March 2015.
- [10] H. Lu, K. Plataniotis, and A. Venetsanopoulos, "MPCA: Multilinear principal component analysis of tensor objects," *IEEE Trans. Neural Networks*, vol. 19, no. 1, pp. 18–39, Jan 2008.
- [11] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "On the best rank-1 and rank-( $r_1, r_2, \dots, r_n$ ) approximation of higher-order tensors," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1324–1342, Mar. 2000.
- [12] L. D. Lathauwer and J. Vandewalle, "Dimensionality reduction in higher-order signal processing and rank-( $R_1, R_2, \dots, R_N$ ) reduction in multilinear algebra," *Linear Algebra and its Applications*, vol. 391, no. 0, pp. 31 – 55, 2004.
- [13] F. Verstraete, D. Porras, and J. I. Cirac, "Density matrix renormalization group and periodic boundary conditions: A quantum information perspective," *Phys. Rev. Lett.*, vol. 93, no. 22, p. 227205, Nov 2004.

- [14] G. Vidal, “Efficient classical simulation of slightly entangled quantum computation,” *Phys. Rev. Lett.*, vol. 91, no. 14, p. 147902, Oct 2003.
- [15] —, “Efficient simulation of one-dimensional quantum many-body systems,” *Phys. Rev. Lett.*, vol. 93, no. 4, p. 040502, Jul 2004.
- [16] D. Pérez-García, F. Verstraete, M. Wolf, and J. Cirac, “Matrix product state representations,” *Quantum Information and Computation*, vol. 7, no. 5, pp. 401–430, 2007.
- [17] I. V. Oseledets, “Tensor-train decomposition,” *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.
- [18] U. Schollwöck, “The density-matrix renormalization group in the age of matrix product states,” *Annals of Physics*, vol. 326, no. 1, pp. 96 – 192, 2011.
- [19] S. A. Nene, S. K. Nayar, and H. Murase, “Columbia object image library (coil-100),” *Technical Report CUCS-005-96*, Feb 1996.
- [20] M. Pontil and A. Verri, “Support vector machines for 3d object recognition,” *IEEE Trans. Patt. Anal. and Mach. Intell.*, vol. 20, no. 6, pp. 637–646, Jun 1998.
- [21] (2013) Data set for single trial 64-channels eeg classification in bci. [Online]. Available: <http://bcmi.sjtu.edu.cn/resource.html>
- [22] A. Georgiades, P. Belhumeur, and D. Kriegman, “From few to many: illumination cone models for face recognition under variable lighting and pose,” *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 23, no. 6, pp. 643–660, Jun 2001.
- [23] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.
- [24] M. Ishteva, P.-A. Absil, S. van Huffel, and L. de Lathauwer, “Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme,” *SIAM J. Matrix Anal. Appl.*, vol. 32, no. 1, pp. 115–135, 2011.
- [25] Q. Li and D. Schonfeld, “Multilinear discriminant analysis for higher-order tensor data classification,” *IEEE Trans. Patt. Anal. and Mach. Intell.*, vol. 36, no. 12, pp. 2524–2537, Dec 2014.
- [26] Q. Zhao and L. Zhang, “Temporal and spatial features of single-trial eeg for brain-computer interface,” *Computational Intelligence and Neuroscience*, vol. 2007, pp. 1–14, Jun 2007.
- [27] A. H. Phan, “NFEA: Tensor toolbox for feature extraction and application,” Lab for Advanced Brain Signal Processing, BSI, RIKEN, Tech. Rep., 2011.