High Order Tensor Formulation for Convolutional Sparse Coding

Adel Bibi and Bernard Ghanem
King Abdullah University of Science and Technology (KAUST), Saudi Arabia
adel.bibi@kaust.edu.sa, bernard.ghanem@kaust.edu.sa

Abstract

Convolutional sparse coding (CSC) has gained attention for its successful role as a reconstruction and a classification tool in the computer vision and machine learning community. Current CSC methods can only *reconstruct single-feature 2D images independently. However, learning multi-dimensional dictionaries and sparse codes for the reconstruction of multi-dimensional data is very important, as it examines correlations among all the data jointly. This provides more capacity for the learned dictionaries to better reconstruct data. In this paper, we propose a generic and novel formulation for the CSC problem that can handle an arbitrary order tensor of data. Backed with experimental results, our proposed formulation can not only tackle applications that are not possible with standard CSC solvers, including colored video reconstruction (5D- tensors), but it also performs favorably in reconstruction with much fewer parameters as compared to naive extensions of standard CSC to multiple features/channels.*

1. Introduction

Convolutional Sparse Coding (CSC) is an important building block for a plethora of applications ranging from image and video compression to deep convolutional networks [17]. To name a few, it has been successfully applied in visual object tracking [28], image and video processing [7, 2, 6], computational imaging [9], low- and mid-level feature learning [5], high-level vision problems [26, 27] and even line drawings [23]. It can be argued that the success of CSC may be biologically inspired, since there exists evidence linking CSC to how images are formed in the human visual system [22]. In principle, images are formed due to the firing of a sparse number of receptive fields in the human brain. This can be represented mathematically as a sum of convolutional operators firing at sparse locations. Mathematically, the standard CSC problem is formulated as follows:

$$\min_{\mathbf{x}_k, \mathbf{d_k} \forall k} \ \frac{1}{2} \sum_{n}^{N} \|\mathbf{y}_n - \sum_{k}^{K} \mathbf{d}_k * \mathbf{x}_k^n\|_2^2 + \lambda \sum_{k}^{K} \|\mathbf{x}_k^n\|_1 \quad \text{s.t.} \ \ \|\mathbf{d}_k\|_2^2 \leq 1 \ \ \forall k = 1, ..., K \tag{1}$$

where * denotes 2D circular convolution, $\mathbf{y}_n$ is the image to be reconstructed, $\mathbf{x}_k^n$ is the k^{th} sparse code for the n^{th} image $\mathbf{y}_n$, and $\mathbf{d}_k$ is the k^{th} filter to be learned. The penalty parameter $\lambda > 0$ controls the trade off between good reconstruction and code sparsity. Problem (1) is similar in essence and form to the standard dictionary learning objective [1, 20]. However, the main difference lies in modeling how an image is reconstructed. While CSC assumes that an image is a sum of convolution responses between filters $\mathbf{d}_k$ and sparse maps $\mathbf{x}_k^n$, dictionary learning reconstructs an image as a sparse number of linear (unstructured) combinations of general dictionary elements [16, 20, 21].

Problem (1) is biconvex and non-smooth. Thus, finding efficient solvers remains a challenge. However, there have been several works in the literature that address this challenge. For instance, the work of [4, 15] uses Parseval's theorem to transform the objective completely into the frequency domain. By introducing auxiliary variables to separate the coding from the dictionary learning subproblem, the joint optimization is solved using the Alternating Direction Method of Multipliers (ADMM). Heide *et al*. [8] solves the CSC problem by using a classical fixed point method, where one alternates between optimizing for the sparse codes and the dictionaries independently. For each subproblem (sparse coding/dictionary learning), ADMM can be used on the underlying convex optimization. Bibi *et al*. [3] also proposed a way to solve one of the subproblems efficiently.

However, due to the nature of the problem and its difficulty, learning multi-dimensional dictionaries (*e.g*. collections of colored images, colored videos, hyperspectral images, or in general videos represented by features of multiple sources) and their sparse codes to reconstruct multi-dimensional data is fundamental but remains a challenge. Joint treatment of features acquired from multiple sources (*e.g*. spectral images, HOG features, and colors) often leads to better performance when compared to treating them separately due to the high order correlations between features. This is evident for instance in some previous work on face recognition [24], and robust PCA [19] to name a few. In this paper, we propose a generic new formulation to the CSC problem in the tensor domain. The proposed model relies on a relatively recent tensor factorization strategy (called t-SVD) tailored to third-order tensors [12, 11]. We extend this

factorization to an arbitrary order tensor which allows us to seamlessly handle multi-dimensional data (images/videos) more naturally and incorporate correlations among the various features/channels. Interestingly, the new data formation model reduces back to standard CSC, as well as, standard sparse dictionary learning when the tensor order is set to particular sizes.

Tensor factorization techniques and multi-linear analysis provide an essential tool for handling multi-dimensional data. Kolda *et al*. [13, 14] provides a good summary of tensor decompositions and their applications. For instance, the tensor Tucker decomposition has been widely used for tensor completion [18] and face recognition [24]. The t-SVD factorization [12, 11] for third order tensors has been used for sparse dictionary learning problems [29], robust PCA [19], clustering multi-way data [10], and image inpainting [32, 31]. However, the t-SVD on third order tensors is limited to 1D convolutions and cannot handle higher order tensor inputs. This prohibits its use in CSC problems, where at least 2D convolutions are required for reconstructing multi-dimensional data.

Contributions. (i) We propose a novel generic tensor formulation to the CSC problem, where standard CSC and standard sparse dictionary learning are merely special cases. **(ii)** To the best of our knowledge, we are proposing the first CSC method that explicitly handles multi-dimensional data. So, for baseline comparisons on multi-dimensional data, we test our method against standard CSC applied to the different dimensions (channels) independently. Extensive experiments demonstrate that our model can achieve similar results as standard CSC techniques with much fewer parameters and performs significantly better in high sparsity domains. **(iii)** We demonstrate the ease of extending our formulation to an arbitrary order tensor by learning a dictionary and sparse codes from colored video (5D tensor), a task that has not been systematically possible before.

2. Notations and Preliminaries

In this section, we introduce the notations that shall be used throughout the paper. The operator $\otimes$ indicates a tensor product, while the $\mathbf{H}$ superscript over matrices indicates hermitian conjugate. The matrices $\mathbf{F}_n$ and $\mathbf{I}_n$ indicate the $(n \times n)$ normalized discrete Fourier transform matrix and the identity matrix, respectively. Following the notations and definitions of [14], the order of a tensor is the number of its dimensions. For instance, scalars are tensors of order zero, vectors are tensors of order one, and matrices are tensors of order two. They will be denoted by lowercase, bold lowercase, and bold capital letters (*e.g.* a, $\mathbf{a}$, $\mathbf{A}$), respectively. Higher order tensors refer to tensors of order three and more, and they will be denoted by boldface Euler script letters (*e.g.* $\mathcal{A}$). Higher order tensors having a second dimension of one are traditionally called vector tensors and are denoted as $\vec{\mathcal{A}} \in \mathbb{R}^{J_1 \times 1 \times J_3 \cdots \times J_n}$ [11]. The Frobenius squared norm of an N^{th}-order tensor $\mathcal{A} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ is $\|\mathcal{A}\|_F^2 = \sum_{i_1,i_2,\ldots,i_N} \mathcal{A}(i_1, i_2, \ldots, i_N)^2$, while its $\ell_{1,\ldots 1}$ norm is $\|\mathcal{A}\|_{1,\ldots 1} = \sum_{i_1,\ldots,i_N} |\mathcal{A}(i_1, \ldots, i_N)|$. The inner product between two tensors of the same size is $\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1,\ldots,i_N} \mathcal{A}(i_1, \ldots, i_N)\mathcal{B}(i_1, \ldots, i_N)$. It is often convenient to unfold a tensor into a matrix. The unfold operation can be done along any of its dimensions. Thus, a k^{th}-mode fold/unfold of a tensor is defined as $\text{unfold}_k(\mathcal{A}) := \mathbf{A}_{(k)} \in \mathbb{R}^{J_k \times (J_1 \ldots J_{k-1} J_{k+1} \ldots J_N)}$ and $\text{fold}_k(\mathbf{A}_{(k)}) := \mathcal{A}$. A tensor-matrix product depends on the dimension along which the product is conducted. It is called an n-mode product, if the product is along the n^{th} dimension. For $\mathbf{U} \in \mathbb{R}^{I \times J_n}$, the n-mode product is defined as $(\mathcal{A} \times_n \mathbf{U})_{(i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N)} = \sum_{i_n=1}^{J_n} \mathcal{A}(i_1, \ldots, i_N)\mathbf{U}(j, i_n)$. We also review some notations necessary for introducing the 3^{rd}-order tensor factorization using t-SVD. All the notations and preliminaries that follow assume $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. We use the MATLAB notation $\mathcal{A}(i, :, :)$, $\mathcal{A}(:, j, :)$, and $\mathcal{A}(:, :, k)$ to denote the i^{th} horizontal, lateral, and frontal slice of a 3^{rd}-order tensor. For simplicity, we denote $\mathbf{A}^{(k)} = \mathcal{A}(:, :, k)$ as the k^{th} frontal slice. For a 3^{rd}-order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we have the following definitions [12, 11]:

Definition 2.1 (Circulant Tensor Unfolding) [12]:

$$circ(\mathcal{A}) = \begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(n_3)} & \cdots & \mathbf{A}^{(2)} \\ \mathbf{A}^{(2)} & \mathbf{A}^{(1)} & \cdots & \mathbf{A}^{(3)} \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{A}^{(n_3)} & \mathbf{A}^{(n_3-1)} & \cdots & \mathbf{A}^{(1)} \end{bmatrix} \tag{2}$$

where $circ(\mathcal{A})$ is a block circulant matrix of size $n_1 n_3 \times n_2 n_3$, which essentially generates circular shifts out of the blocks of the frontal slices of $\mathcal{A}$.

Definition 2.2 (Tensor Vectorization) [12]:

$$MatVec(\mathcal{A}) = \begin{bmatrix} \mathbf{A}^{(1)} \\ \mathbf{A}^{(2)} \\ \vdots \\ \mathbf{A}^{(n_3)} \end{bmatrix}, \quad fold(MatVec(\mathcal{A})) = \mathcal{A} \tag{3}$$

where $fold(.)$, not to be confused with the n^{th}-mode fold, is subscript-free and simply re-concatenates the frontal slices into a 3^{rd}-order tensor. Note that the operator $circ(\mathcal{A})$ can be in fact written as a composite of two operators as follows $circ(\mathcal{A}) = CI(MatVec(\mathcal{A}))$, where $CI(.)$ simply generates all circular shifts out of the fundamental blocks of $MatVec(\mathcal{A})$. The reason behind the introduction of $CI(.)$ will become apparent in later sections.

Definition 2.3 (t-product) [12, 11]: The t-product between $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$ is an $n_1 \times$

$n_4 \times n_3$ tensor $\mathcal{Z}$ defined as follows:

$$\mathcal{Z} = \mathcal{A} \circledast \mathcal{B} = fold(circ(\mathcal{A})MatVec(\mathcal{B})) \quad (4)$$

$$\text{such that: } \mathcal{Z}(i,j,:) = \sum_{k=1}^{n_2} \mathcal{A}(i,k,:) \circledast \mathcal{B}(k,j,:) \ \forall i,j \quad (5)$$

Notice that the 3rd-order tensor $\mathcal{A}$ can be seen as a concatenation of vector tensors in the second dimension. So, the t-product becomes analogous to standard matrix vector multiplication with the inner operation replaced with t-products (refer to Figure 1). It is important to note that $circ(.)$ introduces structure to the tensor resulting in an efficient Fourier diagonalization, as follows:

$$circ(\mathcal{A}) = (\mathbf{F}_{n_3} \otimes \mathbf{I}_{n_1}) bdiag(MatVec(\hat{\mathcal{A}}))(\mathbf{F}_{n_3} \otimes \mathbf{I}_{n_2})^{\mathbf{H}}$$

$$\text{such that: } bdiag(MatVec(\hat{\mathcal{A}})) = \begin{bmatrix} \hat{\mathbf{A}}^{(1)} & & \\ & \ddots & \\ & & \hat{\mathbf{A}}^{(n_3)} \end{bmatrix}$$

The matrix $bdiag(MatVec(\hat{\mathcal{A}}))$ is a concatenation of the frontal slices of $\hat{\mathcal{A}}$ along a diagonal matrix, where $\hat{\mathcal{A}}$ denotes the result of the discrete Fourier transform of $\mathcal{A}$ along the 3rd dimension, *e.g.* $\hat{\mathcal{A}} = fft(\mathcal{A}, [\,], 3)$ in MATLAB.

3. Mathematical Tensor Formulation

In this section, we give a detailed discussion of our tensor CSC formulation using the new proposed operator for higher order t-products that has the potential to uncover high dimensional correlation among features/channels in the data. Throughout this section and for the sake of generality, both images and filters will be treated as arbitrary d-dimensional tensors. For example, a colored image/filter $\mathcal{I}_1$ can be represented by a 3rd-order tensor $\mathcal{I}_1 \in \mathbb{R}^{3 \times n_2 \times n_3}$, where n_2 and n_3 are the spatial dimensions. However, throughout the derivation of our formulation, we always allocate the second dimension for the concatenation of different images/filters and that means all d-dimensional images/filters are represented by a $(d+1)$-dimensional tensor. Thus, our example image/filter is now $\vec{\mathcal{I}}_1 \in \mathbb{R}^{3 \times 1 \times n_2 \times n_3}$, and the set of images/filters $\{\vec{\mathcal{I}}_1, \ldots, \vec{\mathcal{I}}_N\}$ can now be concatenated along the second dimension with $\mathcal{I} \in \mathbb{R}^{n_1 \times N \times n_2 \times n_3}$.

CSC can only be written as a linear sum of t-products, if the images/filters are 1D signals (*i.e.* vectorized patches):

$$\begin{aligned} \min_{\mathcal{D}, \vec{\mathcal{X}}} \quad & \frac{1}{2} \sum_n^N \|\vec{\mathcal{Y}}_n - \mathcal{D} \circledast \vec{\mathcal{X}}_n\|_F^2 + \lambda \|\vec{\mathcal{X}}_n\|_{1,1,1} \\ \text{s.t.} \quad & \|\vec{\mathcal{D}}_k\|_F^2 \leq 1 \ \forall k = 1, \ldots, K \end{aligned} \quad (6)$$

where $\circledast$ is appropriately defined as in Equation (4), $\vec{\mathcal{Y}}_n \in \mathbb{R}^{n_1 \times 1 \times n_2}$, $\mathcal{D} \in \mathbb{R}^{n_1 \times K \times n_2}$, and $\vec{\mathcal{X}}_n \in \mathbb{R}^{K \times 1 \times n_2}$. A similar formulation has been used for standard sparse dictionary learning [32, 30], but with a tube norm $\|.\|_{1,1,2}$ replacing the $\|.\|_{1,1,1}$ norm. Unfortunately, such formulations ignore the 2D spatial structure in images and cannot handle multi-dimensional features at each pixel.

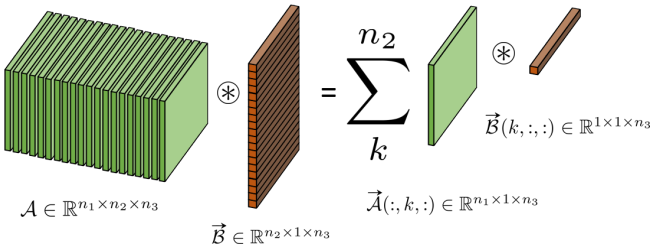


Figure 1. Shows how the t-product operation is analogous to the matrix-vector multiplication with the inner product replaced with t-products. It is an illustration of equation (5) where $n_4 = 1$.

To overcome the aforementioned issues of previous tensor formulations, we consider an arbitrary order tensor, where the n^{th} multi-dimensional training image is $\vec{\mathcal{Y}}_n \in \mathbb{R}^{n_1 \times 1 \times n_2 \times \cdots \times n_d}$, the K filters $\{\vec{\mathcal{D}}_i\}_{i=1}^K$ are concatenated along the second dimension forming the dictionary $\mathcal{D} \in \mathbb{R}^{n_1 \times K \times n_2 \times \cdots \times n_d}$, and the n^{th} sparse code is $\vec{\mathcal{X}}_n \in \mathbb{R}^{K \times 1 \times n_2 \times \cdots \times n_d}$. In this generalization, each filter has the same dimensionality as each image. However, in Section 4, we discuss how filters with smaller spatial extent can be learned. To this end, new tensor theory has to be developed for high dimensional t-product decompositions.

We start by defining the operator $\circledast_{HO}$ for tensors with order higher than three. To do so, we redefine the $circ(.)$, $MatVec(.)$, and $fold(.)$ operators for generic order tensors as $circ_{HO}(.)$, $MatVec_{HO}(.)$, and $fold_{HO}(.)$ where:

Definition 3.1 (High Order t-products):

$$\mathcal{D} \circledast_{HO} \vec{\mathcal{X}} = fold_{HO}(circ_{HO}(\mathcal{D})MatVec_{HO}(\vec{\mathcal{X}})) \quad (7)$$

The operators $circ_{HO}(.)$ and $MatVec_{HO}(.)$ apply $circ(.)$ and $MatVec(.)$ recursively on all the dimensions in the order $(3, 4, \ldots, d)$ as follows:

Definition 3.2 (High Order Recursive $MatVec(.)$):

$$MatVec_{HO}(.) = MatVec_{(d)}(\ldots(MatVec_{(3)}(.)))$$

Here, $MatVec_{(i)}$ is the standard $MatVec(.)$ operator, but it unfolds along the i^{th} dimension such that $MatVec_{HO}(.) : \mathbb{R}^{n_1 \times K \times n_2 \times \cdots \times n_d} \rightarrow \mathbb{R}^{(n_1 n_2 \ldots n_d) \times K}$ and $MatVec_{(i)}(.) : \mathbb{R}^{n_1 \times K \times n_2 \times \cdots \times n_d} \rightarrow \mathbb{R}^{(n_1 n_i) \times K \times n_2 \times \cdots \times n_{i-1} \times 1 \times n_{i+1} \times \cdots \times n_d}$.

Definition 3.3 (High Order Recursive $circ(.)$):

$$circ_{HO}(.) = CI_{(d)}(MatVec_{(d)}(\ldots(CI_{(3)}(MatVec_3(.)))))$$

Similarly, the operator $CI_{(i)}(.)$ circularly shifts the tensor blocks of size n_i along the second dimension to form a block circulant tensor such that $CI_i(.) : \mathbb{R}^{(n_1 n_i) \times K \times n_2 \times \cdots \times n_{i-1} \times 1 \times n_{i+1} \times \cdots \times n_d} \rightarrow$

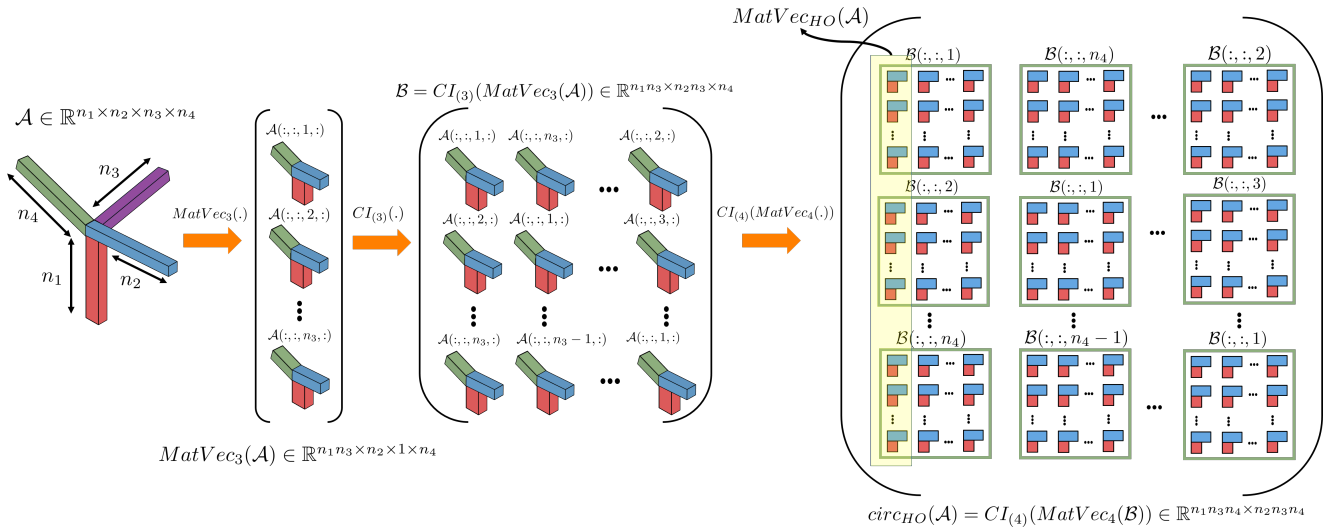


Figure 2. Demonstrates the operation of the proposed operators ($MatVec_i(.)$, $CI_{(i)}(.)$, and $circ_{HO}(.)$) on a 4^{th}-order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$ from left to right. In practice, there is no need to construct the matrix $circ_{HO}(\mathcal{A})$ as it can be efficiently diagonalized in the Fourier domain. Refer to text for more details. It is also clear that if $n_1 = n_2 = 1$, the resultant matrix $circ_{HO}(\mathcal{A})$ is best known as a Block Circulant with Circulant Blocks matrix (BCCB) which represents a 2D convolutional operator.

$\mathbb{R}^{(n_1 n_i) \times (K n_i) \times n_2 \times \cdots \times n_{i-1} \times n_{i+1} \times \cdots \times n_d}$. Thus, $circ_{HO}(.)$ recursively applies $circ(.)$ from the third dimension to the last to produce an $\mathbb{R}^{(n_1 n_2 \ldots n_d) \times (K n_2 \ldots n_d)}$ matrix.

Simply, $fold_{HO}(.) : \mathbb{R}^{(n_1 n_2 \ldots n_d) \times (K n_2 \ldots n_d)} \rightarrow \mathbb{R}^{n_1 \times K \times n_2 \times \cdots \times n_d}$ refolds the matrix back into a tensor in the same order. Figure 2 demonstrates all the previously defined higher order operators on a 4^{th} order tensor.

By induction, one can show the following diagonalization property for any $\mathcal{D} \in \mathbb{R}^{n_1 \times K \times n_2 \times \cdots \times n_d}$,

$$circ_{HO}(\mathcal{D}) = (\mathbf{F}_{n_d} \otimes \mathbf{F}_{n_{d-1}} \otimes \cdots \otimes \mathbf{F}_{n_2} \otimes \mathbf{I}_{n_1}) \quad (8)$$
$$bdiag(MatVec_{HO}(\hat{\mathcal{D}}))(\mathbf{F}_{n_d} \otimes \mathbf{F}_{n_{d-1}} \otimes \cdots \otimes \mathbf{F}_{n_2} \otimes \mathbf{I}_K)^{\mathbf{H}}$$

where $\hat{\mathcal{D}}$ is the Fourier transform of $\mathcal{D}$ along the dimensions $(3, 4 \ldots, d)$. The operator $bdiag(.)$ diagonalizes the blocks of $MatVec_{HO}(\hat{\mathcal{D}})$ along the first dimension n_1 such that $bdiag(.) : \mathbb{C}^{(n_1 n_2 \ldots n_d) \times K} \rightarrow \mathbb{C}^{(n_1 n_2 \ldots n_d) \times (K n_2 \ldots n_d)}$. Then, $bdiag(MatVec_{HO}(\hat{\mathcal{D}}))$ is a block diagonal matrix with each sub block of size $\mathbb{R}^{n_1 \times K}$ (see Figure 3).

Therefore, the overall tensor CSC problem for arbitrary d^{th}-order tensors (TCSC) can be formulated as:

$$\min_{\mathcal{D}, \vec{\mathcal{X}}} \quad \frac{1}{2} \sum_{n}^{N} \|\vec{\mathcal{Y}}_n - \mathcal{D} \circledast_{HO} \vec{\mathcal{X}}_n\|_F^2 + \lambda \|\vec{\mathcal{X}}_n\|_{\underbrace{1, \ldots, 1}_{d+1}}$$
$$\text{s.t.} \quad \|\vec{\mathcal{D}}_k\|_F^2 \leq 1 \; \forall k = 1, \ldots, K \quad (9)$$

Relation to standard CSC. Our proposed CSC formulation reduces back to standard CSC in Problem (1), when we set $n_1 = 1$, $n_i = 1 \; \forall i \geq 4$, and the second dimension $K = 1$ (for simplicity, we show the case of 1 filter in the dictionary here). That is $\mathcal{D} = \vec{\mathcal{D}} \in \mathbb{R}^{n_1 \times 1 \times n_2 \times n_3}$,

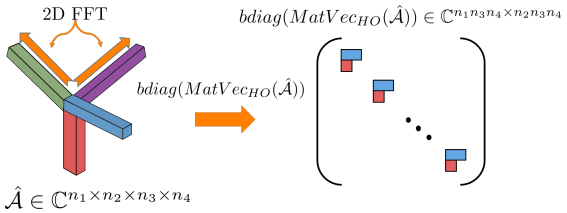


Figure 3. Shows how the operator $bdiag(MatVec_{HO}(.))$ diagonalizes an input tensor $\hat{\mathcal{A}}$.

$\vec{\mathcal{Y}} \in \mathbb{R}^{n_1 \times 1 \times n_2 \times n_3}$, and $\vec{\mathcal{X}} \in \mathbb{R}^{1 \times 1 \times n_2 \times n_3}$. Now, the matrix resulting from $circ_{HO}(\vec{\mathcal{D}})$ can be diagonalized as:

$$circ_{HO}(\vec{\mathcal{D}}) = (\mathbf{F}_{n_3} \otimes \mathbf{F}_{n_2}) bdiag(MatVec_{HO}(\hat{\vec{\mathcal{D}}}))$$
$$(\mathbf{F}_{n_3} \otimes \mathbf{F}_{n_2})^{\mathbf{H}} \quad (10)$$

This is because, from (8), $\mathbf{I}_{n_1} = 1$, $\mathbf{F}_{n_i} = 1 \; \forall i \geq 4$. Here, $\hat{\vec{\mathcal{D}}}$ is the 2D Fourier transform of $\vec{\mathcal{D}}$ along the dimensions (3,4). The diagonalization in (10) indicates that the matrix $circ_{HO}(\vec{\mathcal{D}})$ is a 2D convolutional (also called Block Circulant with Circulant Blocks or BCCB) matrix of the filter $\vec{\mathcal{D}}$, since it is diagonalized using the 2D DFT matrix $\mathbf{F}_{n_3} \otimes \mathbf{F}_{n_2}$. As such, it is easy to show now that $\vec{\mathcal{D}} \circledast_{HO} \vec{\mathcal{X}} = \mathbf{d} * \mathbf{x}$, where $\mathbf{d}$ and $\mathbf{x}$ are the 2D image patches of size $n_2 \times n_3$. As given by the property in (5), we have $\mathcal{D} \circledast_{HO} \vec{\mathcal{X}} = \sum_{k=1}^{K} \mathbf{d}_k * \mathbf{x}_k$, when $\mathcal{D} \in \mathbb{R}^{n_1 \times K \times n_2 \times n_3}$ and $\vec{\mathcal{X}} \in \mathbb{R}^{K \times 1 \times n_2 \times n_3}$ are the concatenations of the K filters along the second dimension and sparse codes along the first, respectively. This is exactly the standard CSC Problem (1).

Relation to standard sparse dictionary learning. Our tensor formulation also reduces back to this important and

popular problem, when we set $n_i = 1\ \forall i \geq 2$. In this case, the tensor $\mathcal{D} \in \mathbb{R}^{n_1 \times K}$ is now a matrix and $\vec{\mathcal{X}} \in \mathbb{R}^{K \times 1}$ is only a vector, thus, leading to $circ_{HO}(\mathcal{D}) = \mathcal{D}$, $MatVec_{HO}(\vec{\mathcal{X}}) = \vec{\mathcal{X}}$, and $\mathbf{F}_{n_i} = 1\ \forall i \geq 2$. Therefore, $\mathcal{D} \circledast \vec{\mathcal{X}} = \mathbf{Dx}$ and the objective is the same as in standard sparse dictionary learning.

4. Proposed Solver

To solve our tensor CSC formulation in Problem (9), we use the traditional fixed point strategy, where we alternate between the optimization of the sparse codes $\vec{\mathcal{X}}_n$ and the dictionary $\mathcal{D}$. Each subproblem is convex, so we use ADMM to solve it. For the sake of notation simplicity and without loss of generality, our ADMM derivation sets the number of training examples $N = 1$. The detailed derivation with $N \geq 1$ is left for the **supplementary material**.

Subproblem (1): Sparse Coding. Fixing $\mathcal{D}$, we solve:

$$\begin{aligned} \arg\min_{\vec{\mathcal{X}}, \vec{\mathcal{Z}}} \ & \frac{1}{2}\|\vec{\mathcal{Y}} - \mathcal{D} \circledast_{HO} \vec{\mathcal{X}}\|_F^2 + \lambda \|\vec{\mathcal{Z}}\|_{\underbrace{1, \ldots, 1}_{d+1}} \\ \text{s.t.} \quad & \vec{\mathcal{X}} = \vec{\mathcal{Z}} \end{aligned} \tag{11}$$

By introducing variable $\vec{\mathcal{Z}}$, we split the smooth and non-smooth parts of the objective. We apply ADMM to iteratively minimize Problem (11) via the following update steps of the primal variables $(\vec{\mathcal{X}}, \vec{\mathcal{Z}})$ and the dual variable $\vec{\mathcal{U}}$.

- **Update** $\vec{\mathcal{X}}$: For computational efficiency, we convert the objective into the Fourier domain by using the definition of the operator $\circledast_{HO}$. Since $\mathbf{F}_{n_d} \otimes \cdots \otimes \mathbf{F}_{n_2} \otimes \mathbf{I}_{n_1}$ and $\mathbf{F}_{n_d} \otimes \cdots \otimes \mathbf{F}_{n_2} \otimes \mathbf{I}_K$ are unitary matrices (or equivalently, by Parseval's theorem), the following can be easily shown true.

$$\begin{aligned} \frac{1}{2}\|\vec{\mathcal{Y}} - \mathcal{D} \circledast_{HO} \vec{\mathcal{X}}\|_F^2 = \frac{1}{2}\|MatVec_{HO}(\hat{\vec{\mathcal{Y}}}) \\ - bdiag\Big(MatVec_{HO}(\hat{\mathcal{D}})\Big) MatVec(\hat{\vec{\mathcal{X}}}))\|_F^2 \end{aligned} \tag{12}$$

where $bdiag\Big(MatVec_{HO}(\hat{\mathcal{D}})\Big)$concatenates the blocks of size n_1 of the matrix $\hat{\mathcal{D}}$ along the diagonal across all dimensions. Therefore, the objective is now separable in the dimensions $(3, \ldots, d)$ and the update rule in the Fourier domain for the i^{th} dimension is:

$$\begin{aligned} \hat{\mathcal{X}}^{(i)} &\leftarrow \arg\min_{\hat{\mathcal{X}}^{(i)}} \frac{1}{2}\|\hat{\mathcal{D}}^{(i)}\hat{\mathcal{X}}^{(i)} - \hat{\mathcal{Y}}^{(i)}\|_2^2 + \frac{\rho_1}{2}\|\hat{\mathcal{X}}^{(i)} - \hat{\mathcal{Z}}^{(i)}\|_2^2 \\ &\quad + < \hat{\mathcal{U}}^{(i)}, \hat{\mathcal{X}}^{(i)} > \\ \hat{\mathcal{X}}^{(i)} &\leftarrow (\hat{\mathcal{D}}^{(i)\mathbf{H}}\hat{\mathcal{D}}^{(i)} + \rho_1 \mathbf{I}_K)^{-1}\Big(\hat{\mathcal{D}}^{(i)\mathbf{H}}\hat{\mathcal{Y}}^{(i)} + \rho_1 \hat{\mathcal{Z}}^{(i)} - \hat{\mathcal{U}}^{(i)}\Big) \end{aligned} \tag{13}$$

Here, $\hat{\mathcal{D}}^{(i)} \in \mathbb{C}^{n_1 \times K}$ is the i^{th} frontal slice of $\hat{\mathcal{D}}$, *i.e.* $\hat{\mathcal{D}}^{(i)} = \hat{\mathcal{D}}(:, :, i)$ where i is a joint linear running index from 1 to $n_3 n_4 \ldots n_d$ since all indices from n_3 to n_d are combined into one for simplicity (refer to Figure (3). The same holds for $\hat{\mathcal{Y}}^{(i)}$ and $\hat{\mathcal{X}}^{(i)}$. As evident from Equation (13), $\vec{\mathcal{X}}$ is computed in the Fourier domain, constructed one frontal slice at a time $\hat{\mathcal{X}}^{(i)} \in \mathbb{R}^{K \times 1}$, as a sparse linear combination of the Fourier elements of the filters in $\hat{\mathcal{D}}^{(i)}$ of the features in the first dimension (*i.e.* n_1). Of course, $\mathcal{X}$ can be reconstructed back from $\hat{\mathcal{X}}^{(i)}\ \forall i$ by taking the inverse Fourier transform of $\hat{\vec{\mathcal{X}}} \in \mathbb{R}^{n_1 \times K \times n_2 \times \cdots \times n_d}$ along dimensions $(3, \ldots, d)$.

- **Update** $\vec{\mathcal{Z}}$: We need to solve:

$$\vec{\mathcal{Z}} \leftarrow \arg\min_{\vec{\mathcal{Z}}} \frac{\lambda}{\rho_1}\|\vec{\mathcal{Z}}\|_{\underbrace{1, \ldots, 1}_{d+1}} + \frac{\rho_1}{2}\|\vec{\mathcal{Z}} - \overbrace{\Big(\vec{\mathcal{X}} + \frac{\vec{\mathcal{U}}}{\rho_1}\Big)}^{\vec{\mathcal{A}}}\|_F^2 \tag{14}$$

This is the proximal operator to the ℓ_1 norm, popularly known as the soft thresholding operator, $S_{\frac{\lambda}{\rho_1}}(a) = sign(a)\max(0, |a| - \frac{\lambda}{\rho_1})$. It is applied in an element-wise fashion to tensor $\vec{\mathcal{A}}$.

- **Update** $\vec{\mathcal{U}}$: $\vec{\mathcal{U}} \leftarrow \vec{\mathcal{U}} + \rho_1(\vec{\mathcal{X}} - \vec{\mathcal{Z}})$

Subproblem (2): Dictionary Learning. Fixing $\vec{\mathcal{X}}$ from subproblem (1), we solve for $\mathcal{D}$ using ADMM, where the augmented Lagrangian is:

$$\begin{aligned} \mathcal{L}(\mathcal{D}, \mathcal{T}, \mathcal{G}) := \frac{1}{2}\|\vec{\mathcal{Y}} - (\mathcal{D} \circledast_{HO} \vec{\mathcal{X}})\|_F^2 + \frac{\rho_2}{2}\|\mathcal{D} - \mathcal{T}\|_F^2 \\ + \langle \mathcal{G}, \mathcal{D} - \mathcal{T} \rangle + \sum_{k=1}^{K} \mathbb{1}_{\{\|\vec{\mathcal{T}}_k\|_F^2 \leq 1\}} \end{aligned} \tag{15}$$

Note that all operations in Equation (15) are preserved under unitary matrix multiplication. Unlike the sparse coding step, the problem can be entirely solved in the Fourier domain. By using the diagonalization property of $\circledast_{HO}$, the augmented Lagrangian is rewritten as:

$$\begin{aligned} \mathcal{L}(\hat{\mathcal{D}}, \hat{\mathcal{T}}, \hat{\mathcal{G}}) := \frac{1}{2}\|\hat{\vec{\mathcal{Y}}} - (\hat{\mathcal{D}} \circledast_{HO} \hat{\vec{\mathcal{X}}})\|_F^2 + \frac{\rho_2}{2}\|\hat{\mathcal{D}} - \hat{\mathcal{T}}\|_F^2 \\ + \langle \hat{\mathcal{G}}, \hat{\mathcal{D}} - \hat{\mathcal{T}} \rangle + \sum_{k}^{K} \mathbb{1}_{\|\hat{\vec{\mathcal{T}}}_k\|_F^2 \leq 1} \end{aligned} \tag{16}$$

- **Update** $\hat{\mathcal{D}}$**:** Similar to updating $\vec{\mathcal{X}}$, updating $\hat{\mathcal{D}}$ is done completely in the Fourier domain for efficiency. The problem is again separable in dimensions $(3, \ldots, d)$ and the update rule for the i^{th} dimension is:

$$\begin{aligned} \hat{\mathcal{D}}^{(i)} &\leftarrow \arg\min_{\hat{\mathcal{D}}^{(i)}} \frac{1}{2}\|\hat{\mathcal{D}}^{(i)}\hat{\mathcal{X}}^{(i)} - \hat{\mathcal{Y}}^{(i)}\|_2^2 + \frac{\rho}{2}\|\hat{\mathcal{D}}^{(i)} - \hat{\mathcal{T}}^{(i)}\|_2^2 \\ &\quad + < \hat{\mathcal{G}}^{(i)}, \hat{\mathcal{D}}^{(i)} > \\ \hat{\mathcal{D}}^{(i)} &\leftarrow \Big(\hat{\mathcal{Y}}^{(i)}\hat{\mathcal{X}}^{(i)\top} + \rho_2 \hat{\mathcal{T}}^{(i)} - \hat{\mathcal{G}}^{(i)}\Big)(\hat{\mathcal{X}}^{(i)}\hat{\mathcal{X}}^{(i)\top} + \rho_2 \mathbf{I}_K)^{-1} \end{aligned} \tag{17}$$

• **Update $\hat{\mathcal{T}}$:** $\hat{\mathcal{T}}$ is updated using the proximal operator for the ℓ_2 unit ball as follows:

$$\hat{\mathcal{T}} \leftarrow \underset{\|\hat{\vec{\mathcal{T}}}_i\|_F^2 \leq 1}{\arg\min} \ \frac{\rho_2}{2}\|\hat{\mathcal{T}} - \Big(\hat{\mathcal{D}} + \frac{1}{\rho_2}\hat{\mathcal{G}}\Big)\|_F^2 \tag{18}$$

However, in CSC problems, the filters $\vec{\mathcal{D}}_k$ have smaller spatial support than the training images $\vec{\mathcal{Y}}$. To allow for a smaller spatial support in the Fourier domain, the constraints $\|\hat{\vec{\mathcal{T}}}_i\|_F^2$ can be replaced with $\|\hat{\vec{\mathcal{T}}}_i \times_p \Psi \times_q \Gamma\|_F^2 \leq 1$. Essentially, we assume that the spatial dimensions are along the p^{th} and q^{th} dimensions of tensor $\mathcal{D}$. The matrices Ψ and Γ simply apply inverse (d-2) inverse Fourier transform on $\hat{\vec{\mathcal{T}}}$ and crop the spatial dimensions according to the required filter sizes. This technique has been used in standard CSC before [4, 8]. As such, $\hat{\vec{\mathcal{T}}}$ is updated by solving:

$$\begin{aligned} \hat{\mathcal{T}} \leftarrow \underset{\hat{\mathcal{T}}}{\arg\min} \ & \frac{\rho_2}{2}\|\hat{\mathcal{T}} - \Big(\hat{\mathcal{D}} + \frac{1}{\rho_2}\hat{\mathcal{G}}\Big)\|_F^2 \\ \text{s.t.} \quad & \|\hat{\vec{\mathcal{T}}}_i \times_p \Psi \times_q \Gamma\|_F^2 \leq 1 \ \ \forall i = 1, \ldots, K \end{aligned} \tag{19}$$

This problem emits the following closed form solution for each filter, where $\mathcal{E}_i = \big(\hat{\mathcal{D}}_i + \frac{1}{\rho_2}\hat{\mathcal{G}}_i\big) \times_p \Psi \times_q \Gamma$:

$$\hat{\mathcal{T}}_i = \begin{cases} \frac{\mathcal{E}_i}{\|\mathcal{E}_i\|_F} & : \text{ if } \|\mathcal{E}_i\|_F^2 \geq 1 \\ \mathcal{E}_i & : \text{ else} \end{cases} \tag{20}$$

• **Update $\hat{\mathcal{G}}$:** $\hat{\mathcal{G}} \leftarrow \hat{\mathcal{G}} + \rho_2(\hat{\mathcal{D}} - \hat{\mathcal{T}})$

The CSC framework is divided into training and testing. During training, we solve Problem (9) to compute $\mathcal{D}$ and $\vec{\mathcal{X}}$ by alternating between subproblems (**1**) and (**2**). For testing, $\mathcal{D}$ is fixed and only subproblem (**1**) is solved to reconstruct the test samples.

5. Experiments

In this section, we conduct three different experiments to demonstrate the effectiveness of using our TCSC formulation. **(1)** Since, to the best of our knowledge, there are no existing methods that perform CSC on multi-dimensional data (images/filters/videos), we compare the reconstruction of TCSC with standard CSC over each feature dimension independently. We call it single channel CSC or SCSC for short. For a fair comparison in terms of reconstruction error, we compare both methods over several levels of sparsity in both training and testing to showcase the effectiveness of TCSC in high sparsity domains with much fewer parameters. Colored images (4^{th} order tensors) are used in this case. **(2)** We demonstrate how TCSC performance is affected by varying K (number of filters). **(3)** Lastly, we run TCSC on (5^{th}-order) tensors for colored video reconstruction.

Figure 4. Shows 6 out of 10 samples from the Fruit and City datasets that have been used for training. The left column shows a total of 12 out of 29 test images.

Datasets. Following common convention in the literature [4, 8] and to vary the datasets across experiments, we use the *city* dataset [26] for training in experiment **(1)** and the *fruit* dataset [26] in experiment **(2)**. Each dataset comprises 10 training images ($N = 10$). For testing in both experiments **(1)** and **(2)**, we randomly select 25 images from the *house* dataset [8] along with four commonly used images in the literature [26, 8]. Figure 4 shows some images in these datasets. As for colored video reconstruction in experiment **(3)** and following common practice in the dictionary learning community [32], we use the basketball video from the OTB50 dataset [25] for training (*i.e.* $N = 1$). From this video, we select 10 frames ($t \in \{1, 10, 20, \ldots, 90\}$) for training and pack them in the last tensor dimension ($n_4 = 10$). Then, we reconstruct 10 intermediate testing frames at $t \in \{5, 15, 25, \ldots, 95\}$.

Complexity. For this analysis, the dictionary is $\mathcal{D} \in \mathbb{R}^{n_1 \times K \times n_2 \times n_3}$. TCSC is computationally more expensive than SCSC (unless $n_1 \gg K$); however, it is more attractive memory-wise. The time complexity of TCSC is $\mathcal{O}(n_2 n_3 K^3) + \mathcal{O}(n_1 n_2 n_3 \log(n_2 n_3))$, as compared to $\mathcal{O}(n_1 n_2 n_3 K^2) + \mathcal{O}(n_1 n_2 n_3 \log(n_2 n_3))$ for SCSC. More importantly, TCSC has n_1 times fewer parameters as compared to SCSC, thus, making it much more memory efficient as it encodes higher order correlations. More details of this analysis are left to the **supplemntary material**.

Parameters and Implementation Details. Since SCSC reconstructs each feature channel separately, we have $n_1 = 1$. All the images from the *fruit*, *city*, and *house* datasets are of spatial size $n_2 = n_3 = 100$. Also, $N = 10$ images are used for training and $K = 100$ filters in all experiments unless stated otherwise. During training, we set $\rho_1 = \rho_2 = 1$ increasing in every iteration as follows $\rho_{1,2} = \min(\rho_{\text{max}}, \gamma\rho_{1,2})$, where $\gamma = 10^{-2}$ and $\rho_{\text{max}} = 600$. As for testing, we set $\rho_1 = 10^{-3}$, $\rho_{\text{max}} = 100$ and $\gamma = 10^{-1}$ for faster convergence. The filter spatial size is 11×11.

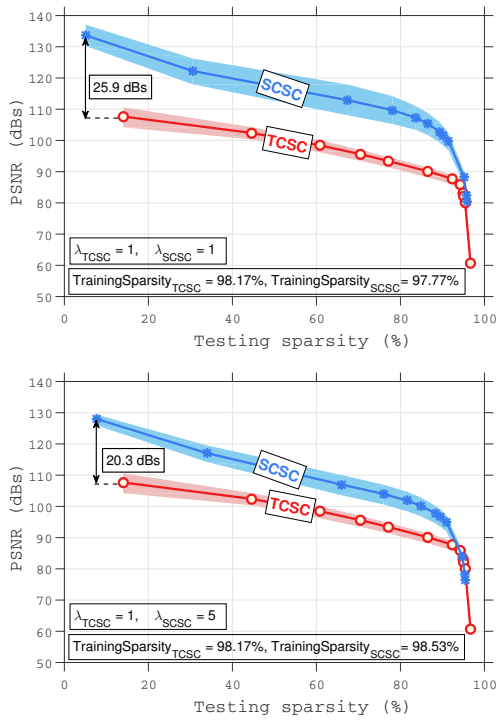


Figure 5. Shows TCSC and SCSC performance on testing set, when the SCSC training sparsity varies from 97.77% to 98.53%.

For TCSC, the images are colored, so $n_1 = 3$. We use the same training setup as SCSC, *i.e.* they share the same n_2, n_3, N, K, spatial filter size, and optimization parameters $(\rho_1, \rho_2, \gamma, \rho_{\max})$. Since the 3rd and 4th dimensions are spatial, then $p = 3$ and $q = 4$. For colored video reconstruction, the TCSC dictionary is $\mathcal{D} \in \mathbb{R}^{3\times100\times100\times100\times10}$. All parameters are detailed in the **supplementary material**.

As for the evaluation, we propose a new method to compare reconstruction quality between two dictionaries for the same test sample. Since the TCSC and SCSC objectives are different, using the same trade off parameter λ for both methods does not guarantee the same sparsity level in the sparse codes. The sparsity level for $\mathcal{X} \in \mathbb{R}^{n_1\times n_2\times n_3}$ is defined as $g(\mathcal{X})/n_1n_2n_3$, where $g(\mathcal{X})$ is the total number of elements in $\mathcal{X}$ such that $|\mathcal{X}(i,j,k)| \leq 10^{-6}$. Therefore, if SCSC has the same sparsity level as TCSC, then this implies that SCSC has n_1 times more non-zero codes than TCSC. This is an important point to make because SCSC has n_1 times more codes than TCSC. To ensure a fair comparison, each method has to reconstruct each test sample with varying levels of sparsity, at which we report the reconstruction PSNR. Both dictionaries are learned using the same amount of sparsity in training.

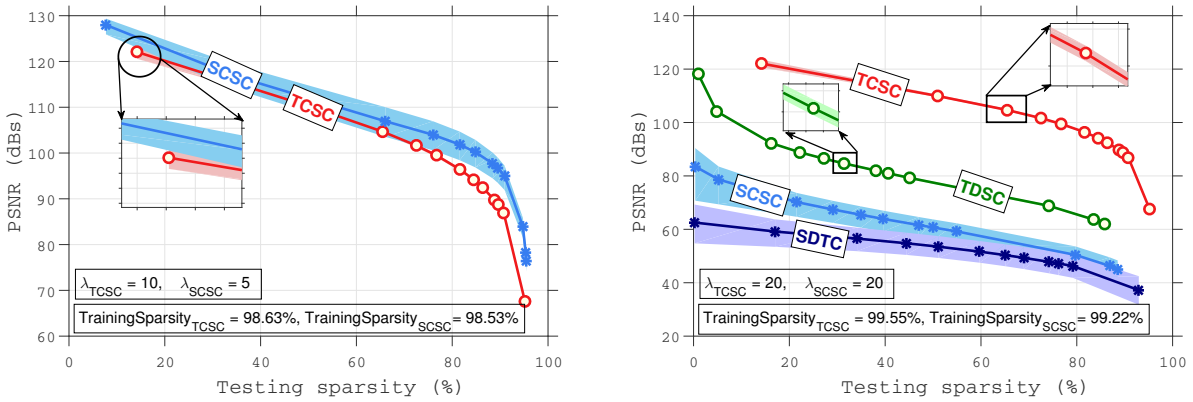
Experiment (1): Colored Image Reconstruction. Here, we compare the performance of TCSC and SCSC dictionaries in reconstructing 29 test samples described earlier. Unlike previous work, we want to ensure that both dictionaries lead to the same amount of sparsity during training and testing. Otherwise, this may lead to an unfair comparison, since dictionaries can be trained for a lower sparsity, which in turn leads to better reconstruction. To demonstrate this, we compare both TCSC and SCSC trained with the same $\lambda = 1$. This leads to a training sparsity level of 98.17% and 97.77% for TCSC and SCSC, respectively. Then, we retrain SCSC with $\lambda = 5$, leading to a higher training sparsity (98.53%). Once these different dictionaries have been trained, we reconstruct the test samples with both methods at varying levels of sparsity by varying λ during testing to generate the results in Figure 5. SCSC outperforms TCSC at all sparsity levels for both training scenarios, but the performance gap does decrease with an increase in training sparsity. In the first case, SCSC training sparsity is much less than that of TCSC. In the second case and since SCSC has $n_1 = 3$ times more parameters (it codes each feature independently), the 0.36% difference in training sparsity, which might not seem significant at first glance, corresponds to SCSC having 2.6×10^5 more non-zero elements in its codes than TCSC. As such, the SCSC dictionary can obviously lead to better reconstruction when the sparsity is allowed to be lower. Thus, it is essential to ensure that both TCSC and SCSC have the same high sparsity level during training. To do this, we train both methods for a large set of λ values and choose the pair that leads to the same sparsity. We detail how the training sparsity for both methods varies with different λ values in the **supplementary material**.

Table 1. Reconstruction error using TCSC trained with $\lambda = 20$ and an average sparsity of 64% across all 29 test samples for different values of K (number of filters in the dictionary).

K	20	40	60	80	100
dBs	70.63	94.45	98.35	104.43	104.70
Avg Sparsity(%)	63.93	64.00	66.66	60.73	65.43

Now, we compare TCSC and SCSC performance, when they are trained using the same sparsity level, first low then high sparsity. These results are summarized in Figure 6. TCSC consistently improves performance over SCSC as the training sparsity level increases for both. At high sparsity, TCSC can significantly outperform SCSC, while using $n_1 = 3$ times *less* parameters than SCSC. The performance gap in favor of TCSC results from the use of high order correlations among all features in the dictionary, instead of treating them independently.

To validate the generalizability of TCSC's dictionary, we use it to reconstruct each of the n_1 features/channels separately using the SCSC coding scheme, denoted by Tensor-Dictionary-Single-Coding (TDSC). This measures the capacity for TCSC's dictionary in reconstructing the independent n_1 features/channels overlooking any correlations among them. Similarly, we use the SCSC dictionary to reconstruct all n_1 channels jointly using TCSC coding, denoted by Single-Dictionary-Tensor-Coding (SDTC). This measures the capacity for SCSC's dictionary to uncover correlations among the n_1 features. Results of this study are shown in Figure 6(b), where the TDSC dictionary outperforms S-



(a) Lower training sparsity. (b) Higher training sparsity.

Figure 6. Shows that TCSC's performance consistently improves over SCSC as the training sparsity level increases, while maintaining n_1 times less parameters. TCSC's performance seems more stable, as it has lower variance across 29 testing samples. Also, Figure 6(b) shows an extra comparison between both TCSC and SCSC dictionaries, when they are used to reconstruct single channels independently and jointly, respectively. These are denoted as TDSC and SDTC, respectively. The TCSC dictionary has the potential to generalize in reconstructing single channels independently (TDSC) much more effectively than both SCSC and SDTC.

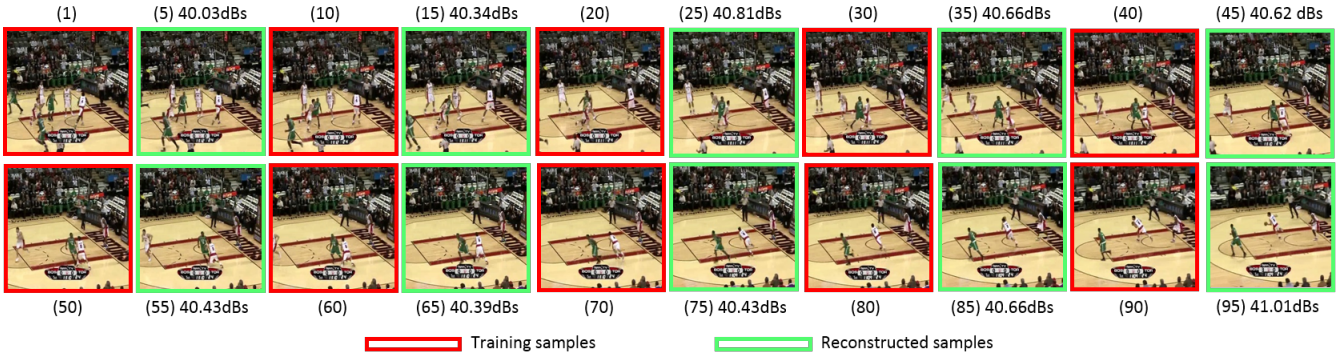


Figure 7. Shows color-coded frames of the basketball video [25] used in training, along with the reconstructed frames. The number in brackets indicates the frame ID. We also report the reconstruction PSNR for each of the reconstructed frames in *green*.

DTC as well as SCSC in PSNR.

Experiment (2): Effect of K. Here, we demonstrate that TCSC performance consistently improves as the number of filters K increases. We reconstruct the 29 test samples at a testing sparsity level of 64% with a varying number of filters K (refer to **supplementary material** for other sparsity levels). Table (1) summarizes these results and shows that TCSC reconstruction performance increases consistently as the number of filters increases from $K = 20$ to $K = 100$.

Experiment (3): Colored Video Reconstruction. To showcase how TCSC can handle an arbitrary order tensor, we conduct an experiment on colored video reconstruction. The dictionary $\mathcal{D} \in \mathbb{R}^{n_1 \times K \times n_2 \times n_3 \times n_4}$, where $n_1 = 3$ (RGB), $n_2 = n_3 = 100$ (spatial dimensions), $K = 100$ (filters), and $n_4 = 10$ (frames from the same video). This task aims to learn K filters (effectively colored video snippets) $\{\vec{\mathcal{D}}\}_{k=1}^{K}$ and their corresponding sparse codes $\{\vec{\mathcal{X}}_k\}_{k=1}^{K}$ to best reconstruct the training sample $\vec{\mathcal{Y}} \in \mathbb{R}^{n_1 \times 1 \times n_2 \times n_3 \times n_4}$. In fact, TCSC makes it possible (for the first time) to learn multi-dimensional filters from video by considering correlations across frames, as well as, the three color channels taken jointly across all pixel locations. Figure 7 shows example reconstructions of test frames after training on other frames in the same video.

6. Conclusions

In this paper, we propose the first tensor formulation for CSC problem (TCSC). TCSC allows the encoding of higher order correlations among features/channels that help in learning high capacity dictionaries. It reduces to standard 2D CSC or standard sparse dictionary learning for some special case of tensor dimensions. It also outperforms a naive extension of standard CSC to multi-dimensions, especially at high sparsity levels, while maintaining much fewer parameters. We show it is possible to extend TCSC to handle any arbitrary order tensor. We give an example of this in the context of colored video reconstruction.

Acknowledgments. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. rmk-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006. 1

[2] M. Aharon, M. Elad, and A. M. Bruckstein. On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them. *Linear algebra and its applications*, 2006. 1

[3] A. Bibi, H. Itani, and B. Ghanem. Fftlasso: Large-scale lasso in the fourier domain. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[4] H. Bristow, A. Eriksson, and S. Lucey. Fast convolutional sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 1, 6

[5] B. Chen, G. Polatkan, G. Sapiro, D. Blei, D. Dunson, and L. Carin. Deep learning with hierarchical convolutional factor analysis. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1887–1901, 2013. 1

[6] F. Couzinie-Devy, J. Mairal, F. Bach, and J. Ponce. Dictionary learning for deblurring and digital zoom. *arXiv preprint arXiv:1110.0957*, 2011. 1

[7] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 2006. 1

[8] F. Heide, W. Heidrich, and G. Wetzstein. Fast and flexible convolutional sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5135–5143, 2015. 1, 6

[9] F. Heide, L. Xiao, A. Kolb, M. B. Hullin, and W. Heidrich. Imaging in scattering media using correlation image sensors and sparse convolutional coding. *Optics express*, 2014. 1

[10] E. Kernfeld, S. Aeron, and M. Kilmer. Clustering multi-way data: a novel algebraic approach. *arXiv preprint arXiv:1412.7056*, 2014. 2

[11] M. E. Kilmer, K. Braman, N. Hao, and R. C. Hoover. Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging. *SIAM Journal on Matrix Analysis and Applications*, 34(1):148–172, 2013. 1, 2

[12] M. E. Kilmer and C. D. Martin. Factorization strategies for third-order tensors. *Linear Algebra and its Applications*, 435(3):641–658, 2011. 1, 2

[13] T. G. Kolda. *Multilinear operators for higher-order decompositions*. United States. Department of Energy, 2006. 2

[14] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009. 2

[15] B. Kong and C. C. Fowlkes. Fast convolutional sparse coding. Technical report, Department of Computer Science, University of California, Irvine, 2014. 1

[16] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural computation*, 2003. 1

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012. 1

[18] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):208–220, 2013. 2

[19] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan. Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5249–5257, 2016. 1, 2

[20] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009. 1

[21] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach. Supervised dictionary learning. In *Advances in neural information processing systems*, 2009. 1

[22] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, pages 3311–3325, 1997. 1

[23] S. Shaheen, L. Affara, and B. Ghanem. Constrained convolutional sparse coding for parametric based reconstruction of line drawings. In *International Conference on Computer Vision (ICCV)*, 2017. 1

[24] M. A. O. Vasilescu and D. Terzopoulos. Multilinear image analysis for facial recognition. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 511–514. IEEE, 2002. 1, 2

[25] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2013. 6, 8

[26] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2528–2535. IEEE, 2010. 1, 6

[27] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2018–2025. IEEE, 2011. 1

[28] T. Zhang, A. Bibi, and B. Ghanem. In defense of sparse tracking: Circulant sparse tracker. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[29] Z. Zhang and S. Aeron. Denoising and completion of 3d data via multidimensional dictionary learning. *arXiv preprint arXiv:1512.09227*, 2015. 2

[30] Z. Zhang and S. Aeron. Denoising and completion of 3d data via multidimensional dictionary learning. *CoRR*, abs/1512.09227, 2016. 3

[31] Z. Zhang and S. Aeron. Exact tensor completion using t-svd. *IEEE Transactions on Signal Processing*, 2016. 2

[32] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer. Novel methods for multilinear data completion and de-noising based on tensor-svd. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3842–3849, 2014. 2, 3, 6