2D Sparse Dictionary Learning via Tensor Decomposition

Sung-Hsien Hsieh
Inst. of Info. Science, Academia Sinica &
Graduate Inst. Comm. Eng., NTU
Email: parvaty316@iis.sinica.edu.tw

Chun-Shien Lu
Inst. of Info. Science
Academia Sinica,
Email: lcs@iis.sinica.edu.tw

Soo-Chang Pei
Graduate Inst. Comm. Eng.
National Taiwan University,
Email: pei@cc.ee.ntu.edu.tw

***Abstract*—The existing dictionary learning methods mostly focus on 1D signals, leading to the disadvantage of incurring overload of memory and computation if the size of training samples is large enough. Recently, 2D dictionary learning paradigm has been validated to save massive memory usage, especially for large-scale problems.**

To address this issue, we propose novel 2D dictionary learning algorithms based on tensors in this paper. Our learning problem is efficiently solved by CANDECOMP/PARAFAC (CP) decomposition. In addition, our algorithms guarantee sparsity constraint, which makes that sparse representation of the learned dictionary is equivalent to the ground truth. Experimental results confirm the effectness of our methods.

***Keywords*—CANDECOMP/PARAFAC (CP) decomposition, Dictionary learning, Sparse representation, Tensor**

I. Introduction

Sparse representation (SR) has attracted much attention in the literature due to its broad applications. SR describes a signal of interest by few features in some analytic transformed domain or learned dictionary; the latter is more appropriate for applications with specific groups of signals, like classification, denoising, and inpainting. The traditional (1D) model of dictionary learning is expressed as $\boldsymbol{Y} = \boldsymbol{D}\boldsymbol{X}$, where $\boldsymbol{Y} \in \mathbb{R}^{m\times l}$ is the collected training data with $\boldsymbol{y}_i$ (i'th column of $\boldsymbol{Y}$) being a 1D training sample, $\boldsymbol{D} \in \mathbb{R}^{m\times n}$ is the dictionary we want to learn, and $\boldsymbol{X} \in \mathbb{R}^{n\times l}$ is the corresponding sparse coefficient matrix. In the 1D model, m is the length of training samples, l is the number of training data, and n is the number of atoms. Let $\boldsymbol{d}_i$ be the i'th atom of $\boldsymbol{D}$. If $n > m$, $\boldsymbol{D}$ is called an overcomplete dictionary. If training samples are not 1D, they are reshaped into 1D vectors.

In the literature, K-SVD [1] and FDDL [2] focus on how to learn overcomplete dictionary. However, their limitation is that m cannot be large because of limited memory storage and limited computational resources. This means that the dictionary only captures local/insufficient or low-frequency information that violate the need of many applications in requiring global structure or detailed information.

Recently, Wang *et al.* [3] and Hawe *et al.* [4] extend dictionary learning to 2D case and propose 2D_SSM and SeDiL, respectively. The advantage of 2D dictionary learning is to save massive memory usage (*e.g.*, saving 98% memory usage with $m = 64$ and $n = 256$) based on the separable property of the dictionary $\boldsymbol{D}$, expressed as $\boldsymbol{D} = \boldsymbol{D}_2 \otimes \boldsymbol{D}_1$, where $\otimes$ is Kronecker product [5] and $\boldsymbol{D}_1, \boldsymbol{D}_2 \in \mathbb{R}^{\sqrt{m}\times\sqrt{n}}$. Under Kronecker structure, the sizes of $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are far smaller than that of $\boldsymbol{D}$ and, thus, both can be learned in a separable manner by popular K-SVD with low memory usage.

Technically, 2D_SSM [3] cannot enforce sparsity constraint and, thus, cannot guarantee that the atoms of learned dictionary are the same with those of original dictionary. 2D_SSM also fails to learn both $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ simultaneously. In addition, our method and SeDiL [4] exhibit different styles of learning dictionary but SeDiL requires tuning several parameters to balance different constraints. We will further conduct performance comparison among the three methods in Sec. IV.

In this paper, we propose two novel 2D dictionary algorithms with and without the use of separable structure, respectively. Our approaches are based on CANDECOMP/ PARAFAC (CP) decomposition [6][7], which is a generalization of singular value decomposition (SVD) from 2D matrices to tensors. For non-separable learning, CP decomposition is directly applied for learning atoms, while for separable learning, the coherence between different atoms should be additionally considered by modifying CP decomposition via Regularized Alternating Least Square (RALS) [8].

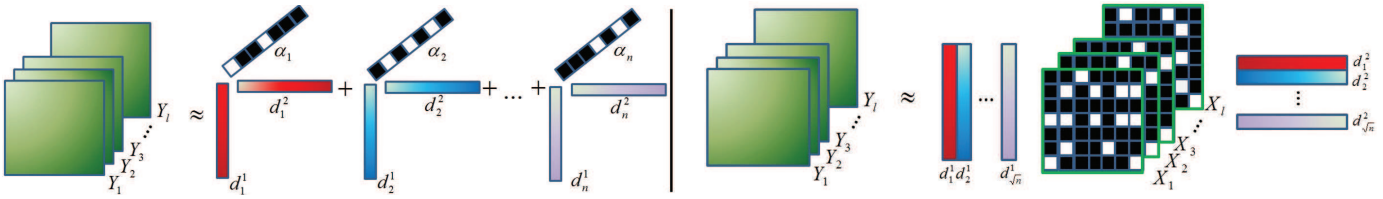
II. Preliminary

Notations and tensor decomposition will be introduced here to ease descriptions later. $\|\cdot\|_0$ denotes ℓ_0-norm. Bold font is used for representing matrix or vector. $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{\sqrt{m}\times\sqrt{m}\times l}$ and $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\sqrt{n}\times\sqrt{n}\times l}$ denote 3D matrices. These multi-dimensional matrices are often called tensors. $\boldsymbol{Y}_i^T$ is the transpose of $\boldsymbol{Y}_i$, where $\boldsymbol{Y}_i \in \mathbb{R}^{\sqrt{m}\times\sqrt{m}}$ is the i'th training sample in the training set $\boldsymbol{\mathcal{Y}}$ for $1 \le i \le l$. $\boldsymbol{d}_i^j$ is the i'th column of $\boldsymbol{D}_j$. $d_{i,k}^j$ is the k'th entry of $\boldsymbol{d}_i^j$ (these rules are also applied to $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$). "$\circ$," "$\odot$," and "$\otimes$", respectively, denote outer product, Khatri-Rao product, and Kronecker product.

$\boldsymbol{\mathcal{Y}}_{(k)}$ is called mode-k (dimension-k) unfolding matrix, where $\boldsymbol{\mathcal{Y}}_{(1)} \in \mathbb{R}^{\sqrt{m}\times\sqrt{ml}}$ is defined by arranging all the mode-1 (dimension-1) vectors as columns of a matrix. Similarly, $\boldsymbol{\mathcal{Y}}_{(2)} \in \mathbb{R}^{\sqrt{m}\times\sqrt{ml}}$ and $\boldsymbol{\mathcal{Y}}_{(3)} \in \mathbb{R}^{l\times m}$. In addition, $\otimes_k$ is called k-mode product of tensor. For instance, $\boldsymbol{\mathcal{Y}} \otimes_k \boldsymbol{D}_1 = \boldsymbol{D}_1\boldsymbol{\mathcal{Y}}_{(k)}$.

CANDECOMP/PARAFAC (CP) decomposition [6][7][9] is a tool for decomposing a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1\times n_2\times n_3}$ into the sum of R rank-one tensors as: $\boldsymbol{\mathcal{A}} \approx \sum_{i=1}^{R} \boldsymbol{u}_i \circ \boldsymbol{v}_i \circ \boldsymbol{w}_i$, where R is the rank of $\boldsymbol{\mathcal{A}}$, $\boldsymbol{u}_i \in \mathbb{R}^{n_1}$,$\boldsymbol{v}_i \in \mathbb{R}^{n_2}$, and $\boldsymbol{w}_i \in \mathbb{R}^{n_3}$.

978-1-4799-7088-9/14/$31.00 ©2014 IEEE



(a) (b)

Fig. 1. 2D sparse dictionary learning. (a) Non-separable model: $\boldsymbol{Y}_i$ is a linear combination of 2D signals, $\alpha_{j,i}\boldsymbol{B}_j$ $(1 \le j \le n)$, where $\boldsymbol{B}_j$ is further decomposed into $\alpha_{j,i}\boldsymbol{d}_j^1 \circ \boldsymbol{d}_j^2$. Black and white blocks indicate zero and non-zero entries, respectively. (b) Separable model: $\boldsymbol{Y}_i$ is decomposed into $\boldsymbol{D}_1\boldsymbol{X}_i\boldsymbol{D}_2^T$.

III. Proposed Algorithms

We describe and compare the proposed separable 2D dictionary learning model and its non-separable version.

A. Non-Separable 2D Dictionary Learning

A dictionary $\boldsymbol{D}$ is said to be non-separable if it cannot be represented as $\boldsymbol{D} = (\boldsymbol{D}_2 \otimes \boldsymbol{D}_1)$. On the contrary, similar to 1D case, we consider that $\boldsymbol{Y}_i$ is a linear combination of 2D bases as $\sum_j \alpha_{j,i}\boldsymbol{B}_j$ $(1 \le j \le n)$, where $\boldsymbol{B}_j \in \mathbb{R}^{\sqrt{m}\times\sqrt{m}}$ can be approximated by rank-one matrix, namely $\boldsymbol{B}_j \simeq \boldsymbol{d}_j^1 \circ \boldsymbol{d}_j^2$ (see Fig. 1(a)), and $\alpha_{j,i}$ is the corresponding sparse coefficient. Note that $\boldsymbol{B}_j \simeq \boldsymbol{d}_j^1 \circ \boldsymbol{d}_j^2$ is the necessary assumption for non-separable 2D dictionary learning. If the assumption doesn't hold, it means that $\boldsymbol{B}_j$ can be decomposed into the sum of more than one rank-one matrices and the number of non-zero coefficient increases, leading to the result that $\boldsymbol{\mathcal{Y}}$ is not sparse in non-separable 2D dictionary.

Now, the objective function for 2D dictionary learning can be formulated as:

$$\delta = \min_{\hat{\boldsymbol{\alpha}}_j, \hat{\boldsymbol{d}}_j^1, \hat{\boldsymbol{d}}_j^2} \|\boldsymbol{\mathcal{Y}} - \sum_j \hat{\boldsymbol{d}}_j^1 \circ \hat{\boldsymbol{d}}_j^2 \circ \hat{\boldsymbol{\alpha}}_j\|, \tag{1}$$

where $\hat{\boldsymbol{\alpha}}_j = [\hat{\alpha}_{j,1}, \hat{\alpha}_{j,2}, ..., \hat{\alpha}_{j,l}]$, as shown in Fig. 1(a). $\|\cdot\|$ is analogous to Frobenius norm, which calculates square root of sum of square of all entries. If sparsity constraint is not enforced ($\hat{\boldsymbol{\alpha}}_j$'s are not sparse), Eq. (1) can simply be solved by CP decomposition. To ensure $\hat{\boldsymbol{\alpha}}_j$'s are sparse, we present a two-step strategy like K-SVD [1].

Step 1: Compute sparse representation $\hat{\boldsymbol{\alpha}}_j$'s with the given $\hat{\boldsymbol{d}}_j^1$'s and $\hat{\boldsymbol{d}}_j^2$'s by 2D OMP [10] that originally requires separable dictionaries. However, it can be modified to fit non-separable structure since it does not require $\hat{\boldsymbol{D}}_1$ and $\hat{\boldsymbol{D}}_2$ but only $\hat{\boldsymbol{B}}_{i,j}$ generated from $\hat{\boldsymbol{d}}_i^1$ and $\hat{\boldsymbol{d}}_j^2$ is needed (see [10]).

Step 2: Given $\hat{\boldsymbol{\alpha}}_j$, update $\hat{\boldsymbol{d}}_j^1$ and $\hat{\boldsymbol{d}}_j^2$. First, Eq. (1) can be rewritten as:

$$\begin{aligned}\|\boldsymbol{\mathcal{Y}} - \sum_j \hat{\boldsymbol{d}}_j^1 \circ \hat{\boldsymbol{d}}_j^2 \circ \hat{\boldsymbol{\alpha}}_j\| &= \|\boldsymbol{\mathcal{Y}} - \sum_{t\neq j} \hat{\boldsymbol{d}}_t^1 \circ \hat{\boldsymbol{d}}_t^2 \circ \hat{\boldsymbol{\alpha}}_t - \hat{\boldsymbol{d}}_j^1 \circ \hat{\boldsymbol{d}}_j^2 \circ \hat{\boldsymbol{\alpha}}_j\| \\ &= \|\boldsymbol{\mathcal{E}}_j - \hat{\boldsymbol{d}}_j^1 \circ \hat{\boldsymbol{d}}_j^2 \circ \hat{\boldsymbol{\alpha}}_j\|,\end{aligned} \tag{2}$$

where $\boldsymbol{\mathcal{E}}_j = \boldsymbol{\mathcal{Y}} - \sum_{t\neq j} \hat{\boldsymbol{d}}_t^1 \circ \hat{\boldsymbol{d}}_t^2 \circ \hat{\boldsymbol{\alpha}}_t$. To ensure the sparsity of $\hat{\boldsymbol{\alpha}}_j$'s, $\boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}_j)$ like the definition in K-SVD is a function used to delete the i'th training sample of $\boldsymbol{\mathcal{E}}_j$ when $\alpha_{j,i} = 0$. Let $\hat{\boldsymbol{\mathcal{E}}}_j = \boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}_j)$, where $\hat{\boldsymbol{\mathcal{E}}}_j \in \mathbb{R}^{\sqrt{m}\times\sqrt{m}\times\|\hat{\boldsymbol{\alpha}}_j\|_0}$ only contains training samples with non-zero coefficients. By employing CP decomposition on $\hat{\boldsymbol{\mathcal{E}}}_j$ with $R = 1$, we can find a solution, $\hat{\boldsymbol{d}}_j^1$, $\hat{\boldsymbol{d}}_j^2$, and update $\hat{\boldsymbol{\alpha}}_j$, that can minimize Eq. (2).

Algorithm 1 : Non-Separable 2D Dictionary Learning.

Input: $\boldsymbol{\mathcal{Y}}$; **Output:** $\hat{\boldsymbol{d}}_j^1$'s, $\hat{\boldsymbol{d}}_j^2$'s;

01. Initialize $\hat{\boldsymbol{d}}_j^1$, $\hat{\boldsymbol{d}}_j^2 \sim \mathbb{N}(0, \frac{1}{m})$, $k = 0$, $\delta_k = 0$.
02. **Do**
03. Solve $\hat{\boldsymbol{\alpha}}_j$'s given $\boldsymbol{d}_j^1$'s, $\boldsymbol{d}_j^2$'s by 2D OMP;
04. **for** $j = 1$ to n
05. Compute $\boldsymbol{\mathcal{E}}_j$;
06. Do CP decomposition of $\boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}_j)$ with $R = 1$;
07. and Update $\hat{\boldsymbol{d}}_j^1$, $\hat{\boldsymbol{d}}_j^2$, $\hat{\boldsymbol{\alpha}}_j$;
08. **end for**
09. $\delta_{k+1} = \|\boldsymbol{\mathcal{Y}} - \sum_{j=1}^{n} \hat{\boldsymbol{d}}_j^1 \circ \hat{\boldsymbol{d}}_j^2 \circ \hat{\boldsymbol{\alpha}}_j\|$, $k = k + 1$;
10. **While** $(\delta_{k+1}/\delta_k > 1)$;

$\mathbb{N}(0, \frac{1}{m})$ denotes a normal distribution.

The proposed algorithm depicted in **Algorithm 1** repeats these two steps until δ in Eq. (1) cannot be further reduced.

B. Separable 2D Dictionary Learning

In separable dictionary learning model, $\boldsymbol{Y}_i$ is decomposed into $\boldsymbol{D}_1\boldsymbol{X}_i\boldsymbol{D}_2^T$, where $\boldsymbol{X}_i$ is sparse, as illustrated in Fig. 1(b).

Instead of alternatively fixing $\boldsymbol{D}_1$ or $\boldsymbol{D}_2$ to simplify this problem, as done in 2D_SSM [3], our proposed algorithm aims to optimize $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ simultaneously. To this end, the objective function is formulated as:

$$\delta = \min_{\hat{\boldsymbol{\mathcal{X}}}, \hat{\boldsymbol{D}}_1, \hat{\boldsymbol{D}}_2} \|\boldsymbol{\mathcal{Y}} - \hat{\boldsymbol{\mathcal{X}}} \otimes_1 \hat{\boldsymbol{D}}_1 \otimes_2 \hat{\boldsymbol{D}}_2\|, \tag{3}$$

where $\hat{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^{\sqrt{n}\times\sqrt{n}\times l}$, $\hat{\boldsymbol{D}}_1 \in \mathbb{R}^{\sqrt{m}\times\sqrt{n}}$, and $\hat{\boldsymbol{D}}_2 \in \mathbb{R}^{\sqrt{m}\times\sqrt{n}}$. $\boldsymbol{\mathcal{Y}}$ can be formulated and summarized as:

$$\begin{aligned}\boldsymbol{\mathcal{Y}} = &(\boldsymbol{d}_j^1 \circ \boldsymbol{d}_k^2 \circ \boldsymbol{x}_{j,k}) + \sum_{a\neq j}^{n} (\boldsymbol{d}_j^1 \circ \boldsymbol{d}_a^2 \circ \boldsymbol{x}_{j,a}) \\ &+ \sum_{b\neq k}^{n} (\boldsymbol{d}_b^1 \circ \boldsymbol{d}_k^2 \circ \boldsymbol{x}_{b,k}) + \sum_{a\neq j}^{n}\sum_{b\neq k}^{n} (\boldsymbol{d}_a^1 \circ \boldsymbol{d}_b^2 \circ \boldsymbol{x}_{a,b}),\end{aligned} \tag{4}$$

where $\boldsymbol{x}_{j,k} = [x_{j,k}^1, x_{j,k}^2, ..., x_{j,k}^l]^T$, to solve Eq. (3). Moreover, let $\boldsymbol{\mathcal{E}}^{j,k} = \boldsymbol{\mathcal{Y}} - \sum_{a\neq j}^{n}\sum_{b\neq k}^{n}(\boldsymbol{d}_a^1 \circ \boldsymbol{d}_b^2 \circ \boldsymbol{x}_{a,b}) = \boldsymbol{d}_j^1 \circ \boldsymbol{d}_k^2 \circ \boldsymbol{x}_{j,k} + \sum_{a\neq j}^{n}(\boldsymbol{d}_j^1 \circ \boldsymbol{d}_a^2 \circ \boldsymbol{x}_{j,a}) + \sum_{b\neq k}^{n}(\boldsymbol{d}_b^1 \circ \boldsymbol{d}_k^2 \circ \boldsymbol{x}_{b,k})$.

Compared with non-separable learning model in Eq. (2), $\boldsymbol{d}_j^1$, $\boldsymbol{d}_k^2$, and $\boldsymbol{x}_{j,k}$ cannot be solved by directly performing CP decomposition on $\boldsymbol{\mathcal{E}}^{j,k}$ since the right-hand side does not

Algorithm 2 : Separable 2D Dictionary Learning

Input: $\mathcal{Y}$; **Output:** $\hat{\boldsymbol{D}}_1$, $\hat{\boldsymbol{D}}_2$;

01. Initialize $\hat{\boldsymbol{D}}_1$, $\hat{\boldsymbol{D}}_2 \sim \mathbb{N}(0, \frac{1}{m})$, $k = 0$.
02. **Do**
03. Solve $\hat{\boldsymbol{\mathcal{X}}}$'s given $\hat{\boldsymbol{D}}_1$, $\hat{\boldsymbol{D}}_2$ by 2D OMP;
04. **for** $j = 1$ to n
05. Compute $\boldsymbol{\mathcal{E}}^{j,j}$;
06. Do Modified CP decomposition of $\boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}_{j,j})$
07. by RALS in Eq.(6)-Eq.(10) and Update $\hat{\boldsymbol{D}}_1$, $\hat{\boldsymbol{D}}_2$, $\hat{\boldsymbol{X}}$;
08. **end for**
09. $\delta_{k+1} = \min_{\hat{\boldsymbol{\mathcal{X}}}, \hat{\boldsymbol{D}}_1, \hat{\boldsymbol{D}}_2} \|\mathcal{Y} - \hat{\boldsymbol{\mathcal{X}}} \otimes_1 \hat{\boldsymbol{D}}_1 \otimes_2 \hat{\boldsymbol{D}}_2\|$, $k = k + 1$;
10. **While** $(\delta_{k+1}/\delta_k > 1)$;

contain $\boldsymbol{d}_j^1$ and $\boldsymbol{d}_k^2$ only. On the contrary, we have to consider not only $\boldsymbol{d}_j^1$, $\boldsymbol{d}_k^2$, and $\boldsymbol{x}_{j,k}$ but also $\boldsymbol{x}_{j,a}$'s and $\boldsymbol{x}_{b,k}$'s.

To deal with this problem, CP decomposition with $R = 1$ is modified and solved by Regularized Alternating Least Square (RALS) [8]. To simplify notations, we let $\boldsymbol{f} = \boldsymbol{d}_j^1$, $\boldsymbol{g} = \boldsymbol{d}_k^2$, and $\boldsymbol{h} = \boldsymbol{x}_{j,k}$. Also let $\boldsymbol{f}^t$ be the solution at the t'th iteration. RALS will run until $\boldsymbol{f}^t$, $\boldsymbol{g}^t$, and $\boldsymbol{h}^t$ no longer change. Please refer to **Appendix** for our detailed derivations. The proposed separable algorithm is depicted in **Algorithm 2**, where all atoms in $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are updated at each iteration.

C. Memory Usage and Computational Overhead Among Separable and Non-Separable Models, and K-SVD

Let 1D dictionary be denoted as $\boldsymbol{D} \in \mathbb{R}^{m \times n}$ and let its corresponding 2D separable dictionaries be $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ ($\in \mathbb{R}^{\sqrt{m} \times \sqrt{n}}$). In addition, in the non-separable model, the learnt 2D dictionary is composed of $\boldsymbol{d}_j^1$'s and $\boldsymbol{d}_j^2$'s ($1 \leq j \leq n$). Apparently, non-separable learning model consumes more memory usage than separable learning model in training because each $\boldsymbol{d}_j^1 \circ \boldsymbol{d}_j^2$ is independent of others, requiring more storage. However, the performance of non-separable dictionary is better due to less limitations.

More specifically, in the testing phase, memory usage of separable and non-separable dictionaries only costs $\frac{2}{\sqrt{mn}}$ and $\frac{2}{\sqrt{m}}$, respectively, of those required in K-SVD. In the training phase, our approaches are a two-step framework similar to K-SVD, as mentioned in Sec. III-A. In Step 1 (sparse signal recovery), 2D OMP, a necessary operation in our approaches, requires $\frac{2}{\sqrt{mn}}$ percent and $\frac{2}{\sqrt{m}}$ percent, respectively, of those required in 1D OMP [3] for separable and non-separable cases. In Step 2 (dictionary updating), according to Eq. (11), the 2D matrix, $\boldsymbol{\mathcal{E}}^{j,k}$, dominates the memory usage since other operations only involve 1D operations. However, we can simply compute each row of $\boldsymbol{\mathcal{E}}^{j,k}$ multiplied by $\left((\boldsymbol{h}^t \odot \boldsymbol{g}^t) - \sum_{a \neq j}^{n} (\boldsymbol{x}_{j,a}^t \odot \boldsymbol{d}_a^2)\right) + \lambda \boldsymbol{f}^t$ instead of storing the whole $\boldsymbol{\mathcal{E}}^{j,k}$. In other words, CP decomposition based on RALS only requires few extra memory.

Since dictionary learning is an off-line task, the time complexity for learning dictionary is secondary and is not a main issue in our proposed algorithm. Since CP decomposition is well-known in tensor decomposition, however, some researches focus on analyzing the complexity of CP decomposition and speeding it up [11][12].

IV. Experimental Results

Our approaches with separable and non-separable learning strategies are denoted as S and NS, respectively. They were compared with K-SVD [1], 2D_SSM [3], and SeDiL [4].

We validate the performance of proposed algorithms for denoising images corrupted by additive Gaussian noise with standard deviation σ_{noise}. Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) [13] were used as the performance criteria. We first cropped noisy images into 10000 8×8 patches as training data with $m = 64$. Then, these patches were used for learning dictionaries with $n = 256$, where initial dictionaries were Gaussian random matrix. For denoising purpose, we used 2D OMP [10] with stopping criteria, $|\boldsymbol{Y}_i - \sum_j \hat{\alpha}_{j,i} \hat{\boldsymbol{d}}_j^1 \circ \hat{\boldsymbol{d}}_j^2| < \sigma_{noise}^2$ and $|\boldsymbol{Y}_i - \hat{\boldsymbol{D}}_1 \hat{\boldsymbol{X}}_i \hat{\boldsymbol{D}}_2^T| < \sigma_{noise}^2$, for our methods NS and S, respectively. We can observe from Table I that both our methods outperform 2D_SSM and SeDiL. Our separable algorithm is slightly inferior to K-SVD but it saves much memory usage in training. Our non-separable learning algorithm is comparable to K-SVD and better than its separable counterpart.

In addition, we also validate the proposed algorithms in terms of synthesis data. First, 1000 testing data, $\boldsymbol{X}_i$ for $1 \leq i \leq 1000$, were produced, where $\|\boldsymbol{X}_i\|_0 = k$ and each non-zero value is assigned to 1 or -1 randomly. Second, we generated $\boldsymbol{D}_1$, $\boldsymbol{D}_2$ as Gaussian random matrices, where ℓ_2-norm of each column was 1 and obtained true dictionary by $\boldsymbol{D}_{truth} = \boldsymbol{D}_2 \otimes \boldsymbol{D}_1$. Third, $\boldsymbol{Y}_i$ was obtained from $\boldsymbol{D}_1 \boldsymbol{X}_i \boldsymbol{D}_2^T$. In K-SVD, all 2D signals were reshaped into 1D signals. Consequently, dictionary learning was conducted from $\mathcal{Y}$. Then, learned dictionary was used to reconstruct $\hat{\boldsymbol{\mathcal{X}}}_i$ from $\boldsymbol{Y}_i$.

We present a new criterion for estimating the similarity between the learned dictionary and the ground truth, which is equivalent to the correctness of sparse representation, as:

$$SIM(\boldsymbol{D}_{truth}, \hat{\boldsymbol{D}}) = \max_P tr\left(abs(\boldsymbol{D}_{truth}^T * (\hat{\boldsymbol{D}} P))\right)/n, \quad (5)$$

where ℓ_2-norms of all atoms of the true dictionary $\boldsymbol{D}_{truth}$ and estimated one $\hat{\boldsymbol{D}}$ are normalized to 1, $tr(\cdot)$ and $abs(\cdot)$ denote the trace and absolute operators, respectively, and P is the permutation matrix that rearrange the columns of $\hat{\boldsymbol{D}}$.

Table II illustrates SIM versus variations of $\frac{m}{n}$ and $\frac{k}{n}$ under $l = 1000$ and $n = 256$. Thus, m and k are set according to the ratios of $\frac{m}{n}$ and $\frac{k}{n}$. We ignore the cases of $m < k$ (labeled as "none") since they are meaningless. It can be observed that our methods attain higher SIM values than both K-SVD and 2D_SSM and are comparable to SeDiL. Because 2D_SSM does not enforce sparsity constraint, it cannot ensure the correctness of sparse representation.

On the other hand, we validated that, compared to 1D dictionary learning, the memory usage of 2D dictionary learning is remarkably reduced, as shown in Fig. 2. Three algorithms, including OMP with 1D dictionary, 2D OMP with non-separable dictionary, and 2D OMP with separable dictionary, were compared in terms of the peak of memory cost. The y-axis in Fig. 2 denotes the memory cost rate when

TABLE I. PERFORMANCE OF DENOISING VS. σ_{noise}'S. THE LEFTMOST COLUMN SHOWS STD. DEVIATION\PSNR OF NOISY IMAGES. THE BEST PERFORMANCE IS LABELED IN BOLD FONT.

σ_{noise}\ PSNR	Boat Image		Barbara Image	
	PSNR (dB)	SSIM	PSNR (dB)	SSIM
	[1] \ [3] \ [4] \ S \ NS	[1] \ [3] \ [4] \ S \ NS	[1] \ [3] \ [4] \ S \ NS	[1] \ [3] \ [4] \ S \ NS
5 \ 34.16	**37.24**\37.01\37.00\37.08\37.17	0.940\0.938\0.937\0.939**0.943**	37.94\37.02\37.82\37.60**38.13**	0.963\0.938\0.962\0.962**0.971**
10\28.11	**33.64**\32.13\32.85\33.47\33.61	0.883\0.875\0.870\0.874**0.886**	34.10\33.01\33.62\34.03**34.19**	0.931\0.907\0.926\0.930**0.932**
20\22.12	30.37\29.83\29.22\30.19**30.43**	0.804\0.784\0.772\0.786**0.805**	30.32\28.89\29.45\30.27**30.35**	0.870\0.827\0.853\0.869**0.881**
30\18.11	28.40\27.63\26.86\28.12**28.51**	0.743\0.720\0.697\0.722**0.749**	27.92\27.02\26.16**27.94**\27.93	0.806\0.781\0.757\0.804**0.815**
60\12.56	25.03\24.25\24.07\24.92**25.14**	0.623\0.596\0.591\0.618**0.627**	23.59\23.16\23.06\23.64**23.84**	0.638\0.613\0.612\0.638**0.650**

TABLE II. SIM VS. VARIATIONS OF $\frac{m}{n}$ AND $\frac{k}{n}$ WITH $n = 256$.

$\frac{m}{n}$ \ $\frac{k}{n}$	0.1	0.2	0.3	0.4
	[1] \ [3] \ [4] \ S \ NS	[1] \ [3] \ [4] \ S \ NS	[1] \ [3] \ [4] \ S \ NS	[1] \ [3] \ [4] \ S \ NS
0.1	0.92\0.89**0.94****0.94****0.94**	none	none	none
0.2	0.89\0.86**0.91**\0.90**0.91**	0.87\0.84\0.86\0.87**0.88**	none	none
0.3	0.86\0.86\0.88\0.88**0.90**	0.81\0.76**0.83****0.83****0.83**	0.76\0.75\0.78\0.79**0.80**	none
0.4	0.85\0.80\0.86\0.85**0.87**	0.79\0.73\0.81\0.81**0.83**	0.77\0.72\0.77\0.77**0.78**	0.73\0.70**0.76**\0.75\0.75
0.5	0.82\0.69\0.82\0.83**0.84**	0.77\0.69\0.79\0.79**0.80**	0.75\0.67\0.75**0.76****0.76**	0.71\0.66\0.74\0.73**0.75**

compared with that required in 1D dictionary learning. For example, 0.1 indicates that 1D dictionary learning needs 10 times memory cost than other methods. Note that the memory cost of 2D_SSM and SeDiL is the same with that of separable dictionary.

Our results show that memory reduction is more obvious when n is larger. In fact, for 1D Dictionary, the program will run out of memory under $n = 2^{13}$. Nevertheless, in non-separable and separable 2D dictionaries, they will fail until $n = 2^{20}$ and $n = 2^{26}$, respectively. For big data like video, a higher-dimensional dictionary is expected to save more memory cost. Thus, 2D dioctionary learning is especially helpful for large-scale problems.

Finally, we also conduct comparison in terms of computational time (results not shown here due to space limit). We can observe that (1) our separable learning algorithm is faster than other methods even though both m and n increase and (2) our non-separable algorithm is slower than SeDiL and its separable counterpart but is comparable to 2D_SSM.

V. CONCLUSION

We have proposed novel 2D dictionary learning algorithms based on the structure of tensors. We investigate how to learn non-separable and separable dictionaries by CP decomposition. Experimental results show that our methods generate accurate sparse representation and converge to good solutions.

ACKNOWLEDGMENT

This work was supported by National Science Council, Taiwan, ROC, under grant NSC 102-2221-E-001-022-MY2.

APPENDIX: ITERATIVE RALS-BASED CP DECOMPOSITION

$$\boldsymbol{f}^{t+1} = \arg\min_{\hat{\boldsymbol{f}}} \|\boldsymbol{\mathcal{E}}^{j,k}_{(1)} - \hat{\boldsymbol{f}}(\boldsymbol{h}^t \odot \boldsymbol{g}^t)^T - \sum_{a\neq j}^{n} \hat{\boldsymbol{f}}(\boldsymbol{x}^t_{j,a} \odot \boldsymbol{d}^2_a)^T\|_F + \lambda\|\hat{\boldsymbol{f}} - \boldsymbol{f}^t\|_F; \tag{6}$$

$$\boldsymbol{x}^{t+1}_{j,a} = \arg\min_{\hat{\boldsymbol{x}}_{j,a}} \|\boldsymbol{\mathcal{E}}^{j,k}_{(3)} - \boldsymbol{h}^t(\boldsymbol{g}^t \odot \boldsymbol{f}^{t+1})^T - \hat{\boldsymbol{x}}_{j,a}(\boldsymbol{d}^2_a \odot \boldsymbol{f}^{t+1})^T - \sum_{c\neq j,a}^{n} \boldsymbol{x}^t_{j,c}(\boldsymbol{d}^2_c \odot \boldsymbol{f}^{t+1})^T\|_F \text{ for all } a \neq j; \tag{7}$$

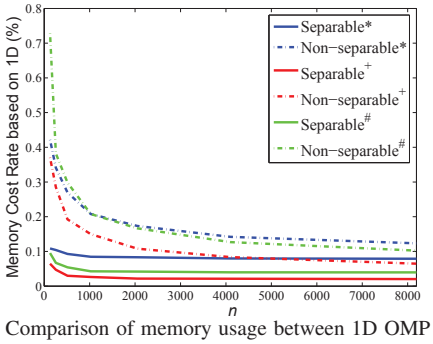


Fig. 2. Comparison of memory usage between 1D OMP with 1D dictionary and 2D OMP with non-separable/separable dictionar. The vertical axis denotes the memory rate calculated by dividing the memory cost of 2D case by that of 1D case. Different parameter settings were represented as * ($\frac{n}{m} = 0.25$ and $\frac{k}{m} = 0.25$), + ($\frac{n}{m} = 0.25$, $\frac{k}{m} = 0.125$), and # ($\frac{n}{m} = 0.125$, $\frac{k}{m} = 0.25$).

$$\boldsymbol{g}^{t+1} = \arg\min_{\hat{\boldsymbol{g}}} \|\boldsymbol{\mathcal{E}}^{j,k}_{(2)} - \hat{\boldsymbol{g}}(\boldsymbol{h}^t \odot \boldsymbol{f}^{t+1})^T - \sum_{b\neq k}^{n} \hat{\boldsymbol{g}}(\boldsymbol{x}^t_{b,k} \odot \boldsymbol{d}^1_b)^T\|_F + \lambda\|\hat{\boldsymbol{g}} - \boldsymbol{g}^t\|_F; \tag{8}$$

$$\boldsymbol{x}^{t+1}_{b,k} = \arg\min_{\hat{\boldsymbol{x}}_{b,k}} \|\boldsymbol{\mathcal{E}}^{j,k}_{(3)} - \boldsymbol{h}^t(\boldsymbol{g}^{t+1} \odot \boldsymbol{f}^{t+1})^T - \hat{\boldsymbol{x}}_{b,k}(\boldsymbol{g}^{t+1} \odot \boldsymbol{d}^1_b)^T - \sum_{c\neq k,b}^{n} \boldsymbol{x}^t_{c,k}(\boldsymbol{g}^{t+1} \odot \boldsymbol{d}^1_c)^T\|_F \text{ for all } b \neq k; \tag{9}$$

$$\boldsymbol{h}^{t+1} = \arg\min_{\hat{\boldsymbol{h}}} \|\boldsymbol{\mathcal{E}}^{j,k}_{(3)} - \hat{\boldsymbol{h}}(\boldsymbol{g}^{t+1} \odot \boldsymbol{f}^{t+1})^T - \sum_{a\neq j}^{n} \boldsymbol{x}^{t+1}_{j,a}(\boldsymbol{d}^2_a \odot \boldsymbol{f}^{t+1})^T - \sum_{b\neq k}^{n} \boldsymbol{x}^{t+1}_{b,k}(\boldsymbol{g}^{t+1} \odot \boldsymbol{d}^1_b)^T\|_F + \lambda\|\hat{\boldsymbol{h}} - \boldsymbol{h}^t\|_F. \tag{10}$$

How to choose a good λ can refer to [8]. Since $\|\cdot\|_F$ is differentiable, Eq. (6) is solvable by first derivative test. Due to space limitation, we show the solution of $\boldsymbol{f}^{t+1}$ below as:

$$\boldsymbol{f}^{t+1} = \frac{\left(\boldsymbol{\mathcal{E}}^{j,k}_{(1)}\left((\boldsymbol{h}^t \odot \boldsymbol{g}^t) - \sum_{a\neq j}^{n}(\boldsymbol{x}^t_{j,a} \odot \boldsymbol{d}^2_a)\right) + \lambda \boldsymbol{f}^t\right)}{S}, \tag{11}$$

where $S = (\boldsymbol{h}^t \odot \boldsymbol{g}^t)^T(\boldsymbol{h}^t \odot \boldsymbol{g}^t) + \sum_{a\neq j}^{n}(\boldsymbol{h}^t \odot \boldsymbol{g}^t)^T(\boldsymbol{x}^t_{j,a} \odot \boldsymbol{d}^2_a) + \sum_{a\neq j}^{n}(\boldsymbol{x}^t_{j,a} \odot \boldsymbol{d}^2_a)^T(\boldsymbol{h}^t \odot \boldsymbol{g}^t) + \sum_{a\neq j}^{n}\sum_{b\neq j}^{n}(\boldsymbol{x}^t_{j,a} \odot \boldsymbol{d}^2_b)^T(\boldsymbol{x}^t_{j,a} \odot \boldsymbol{d}^2_b + \lambda I)$.

REFERENCES

[1] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[2] L. Zhang, X. Feng, and D. Zhang, "Fisher discrimination dictionary learning for sparse representation," in *IEEE ICCV*, 2011, pp. 543–550.

[3] Y. Shi, X. Sun, J. Wang, and B. Yin, "Two dimensional synthesis sparse model," in *IEEE ICME*, 2013.

[4] S. Hawe, M. Seibert, and M. Kleinsteuber, "Separable dictionary learning," in *IEEE CVPR*, 2013, pp. 438–445.

[5] M. Duarte and R. Baraniuk, "Kronecker compressive sensing," *IEEE Trans. on Image Processing*, vol. 21, no. 2, pp. 494–504, 2012.

[6] J.D. Carroll and J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition," *Psychometrika*, vol. 35, pp. 283–319, 1970.

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

[8] N. Li., S. Kindermann, and C. Navasca, "Some convergence results on the regularized alternating least-squares method for tensor decomposition," *Linear Algebra Appl.*, vol. 438, pp. 796–812, 2013.

[9] M. E. Kilmer and C. D. M. Martin, "Decomposing a tensor," *SIAM News*, vol. 37, pp. 1–3, 2004.

[10] Y. Fang, J.J. Wu, and B.M. Huang, "2d sparse signal recovery via 2d orthogonal matching pursuit," *Science China Information sciences*, vol. 55, no. 4, pp. 889 – 897, 2012.

[11] P. Tichavsky A.-H. Phan and A. Cichocki, "Fast alternating ls algorithms for high order candecomp/parafac tensor factorizations," *IEEE Trans. on Signal Processing*, vol. 61, no. 19, pp. 4832–4846, 2013.

[12] M. V. Barel L. Sorber and L. D. Lathauwer, "Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-(lr,lr,1) terms, and a new generalization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 695–720, 2013.

[13] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.