Supplementay Material: Deep Tensor ADMM-Net for Snapshot Compressive Imaging

Jiawei Ma† Xiao-Yang Liu† Zheng Shou Xin Yuan
Columbia University Columbia University Columbia University Nokia Bell Labs
{jm4743, xl2427, zs2262}@columbia.edu xyuan@bell-labs.com

Contents

A. Sensing matrix Video SCI System

As described in the Fig. 1 of the paper, the training video $\mathcal{X}$ is modulated by the sensing matrix $\mathcal{C}$. The value of each unit in the sensing matrix is modeled according to hardware design in the system or preset by the simulation data mask.

For our simulation task, the sensing matrix is preset with each unit being binary, *i.e.*, either 0 or 1 [4], which were used across all simulation data sets for different algorithms. Due to hardware calibration, the sensing matrix of real-world images are modeled by continuous values between 0 and 1.

B. Tensor Operations

Our decoder aims at reconstructing a 3D tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times B}$ and rests on the definition of *Tensor Nuclear-Norm* in Def. 1 in the paper. A 3D tensor $\mathcal{X}$ can be viewed as an $n_1 \times n_2$ matrix of tubes lying in the third-dimension. For two tubes (vectors) of the same size $\boldsymbol{a}$, $\boldsymbol{b} \in \mathbb{R}^{1 \times 1 \times B}$, we denote $\boldsymbol{c} = \boldsymbol{a} \bullet \boldsymbol{b}$ as the *circular convolution* between these two tubes. Accordingly, we introduce the following operators [2, 3, 7].

Definition B.1. ***Tensor Product (t-product)*** *[2, 3, 7]. The t-product $\mathcal{C} = \mathcal{A} * \mathcal{B}$ of $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times B}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times n_3 \times B}$ is a tensor of size $n_1 \times n_3 \times B$, where $\mathcal{C}(i, j, :) = \sum_{s=1}^{n_2} \mathcal{A}(i, s, :) \bullet \mathcal{B}(s, j, :)$, $\forall (i, j) \in \{1, \ldots, n_1\} \times \{1, \ldots, n_3\}$.*

The *t-product* of two tensors is analogous to a matrix-matrix multiplication, except that the scalar multiplication is replaced by the circular convolution operation.

The **identity tensor** $\mathcal{I} \in \mathbb{R}^{n_1 \times n_1 \times B}$ is a tensor whose first frontal slice $\mathcal{I}^{(1)}$ is the $n_1 \times n_1$ identity matrix and all other frontal slices, $\mathcal{I}^{(b)}$ for $b \in \{2, ..., B\}$, are zero matrices. The **tensor transpose** of a 3-D tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times B}$ is defined as $\mathcal{T}^{\dagger} \in \mathbb{R}^{n_2 \times n_1 \times B}$ and obtained by transposing each frontal slice $\mathcal{T}^{(b)}$ and then reversing the order of transposed slices from 2 through B, *i.e.*, $(\mathcal{T}^{\dagger})^{(b)} = (\mathcal{T}^{(B-b+2)})^{H}$ for $b \in \{1, ..., B\}$. Accordingly, $\mathcal{T} \in \mathbb{R}^{n_1 \times n_1 \times B}$ is **orthogonal tensor** when $\mathcal{T}^{\dagger} * \mathcal{T} = \mathcal{T} * \mathcal{T}^{\dagger} = \mathcal{I}$. With those notations, we give the definition of *t-SVD*.

Definition B.2. ***Tensor Singular Value Decomposition (t-SVD in Fig. 1)*** *[3, 7, 2]. The t-SVD of $\mathcal{M} \in \mathbb{R}^{n_1 \times n_2 \times B}$ is given by $\mathcal{M} = \mathcal{U} * \Theta * \mathcal{V}^{\dagger}$, where $\mathcal{U}$ and $\mathcal{V}$ are orthogonal tensors of sizes $n_1 \times n_1 \times B$ and $n_2 \times n_2 \times B$, respectively. $\Theta \in \mathbb{R}^{n_1 \times n_2 \times B}$ is a rectangular ($n_1 = n_2$ in our experiment) tensor, whose frontal slices $\Theta^{(b)}$ for $b \in \{1, ..., B\}$ are all diagonal matrices. Similar to the matrix SVD, the number of non-zero tubes of Θ are called **tubal rank**, denoted as r.*

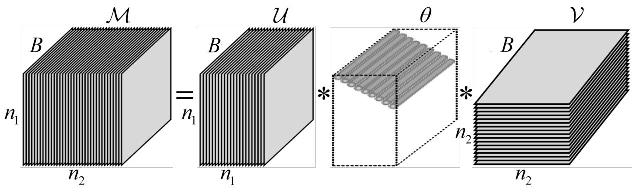

Figure 1. Illustration of tensor-SVD.

We denote tensor $\widetilde{\mathcal{T}}$ as the frequency domain representation of $\mathcal{T}$, obtained by taking the Fourier transform along the third dimension, *i.e.*, $\widetilde{\mathcal{T}}(i, j, :) = \text{fft}(\mathcal{T}(i, j, :))$. Since the Fourier transform is essentially a linear transform, we can further derive the Fourier transformation in matrix form $\Pi \in \mathbb{C}^{n_1 n_2 B \times n_1 n_2 B}$ for $\text{Vec}(\widetilde{\mathcal{X}}) = \Pi \text{Vec}(\mathcal{X})$ and

$$\Pi = \Lambda \otimes \boldsymbol{I}_{n_1 n_2}, \tag{1}$$

where each unit in $\Lambda \in \mathbb{C}^{B \times B}$ is the coefficient of Fourier transformation, $I_{n_1 n_2}$ is an $n_1 n_2 \times n_1 n_2$ identity matrix and $\otimes$ denotes Kronecker (tensor) product. Following this, Π can be viewed as a $B \times B$ matrix of weighted identity matrices lie in the plane and we name this special structure as *Rectangular Diagonal Blocks* (*RDB*) where each block $\Pi_{\{b_1, b_2\}} = \Lambda(b_1, b_2) \boldsymbol{I}_{n_1 n_2}$ for $(b_1, b_2) \in \{1, ..., B\} \times \{1, ..., B\}$ and the inverse matrix $\Pi^{-1} = \Lambda^{-1} \otimes \boldsymbol{I}_{n_1 n_2}$ (demonstration is provided in SM) can be efficiently computed.

As shown in Fig. 3 in the paper, $\Pi_{\text{diag}(n)} \in \mathbb{R}^{B \times B}$ is further defined as the matrix composed of the n^{th} diagonal element of all the blocks. Similarly, we can generalize Π to any transformations, *e.g.*, *DCT, FFT* by replacing Λ and use subscript $_f$ labels the transformation type.

C. Demonstration in Network Implementation

For tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times B}$, we denote tensor $\widetilde{\mathcal{X}}$ as the frequency domain representation of $\mathcal{X}$, obtained by taking the Fourier transform along the third dimension, *i.e.*, $\widetilde{\mathcal{X}}(i,j,:) = \text{fft}(\mathcal{X}(i,j,:))$. Since the Fourier transform is essentially a linear transform, we have

$$\widetilde{\mathcal{X}}(i,j,:) = \Lambda \mathcal{X}(i,j,:). \tag{2}$$

where $(i,j) \in \{1,...,n_1\} \times \{1,...,n_2\}$ and $\Lambda \in \mathbb{C}^{B \times B}$ denotes the transformation matrix corresponding to the Fourier transform. Due to the invertibility of Fourier transform, it is intuitive to derive the following Conclusion 1 so that Λ^{-1}, the transformation matrix of inverse Fourier transformation, exists. For simplicity, we use $\widetilde{\mathcal{X}} = \Lambda \mathcal{X}$ to denote the process in (2) and denote Λ as unitary transformation matrix of Fourier Transformation so that $\Lambda\Lambda^{-1} = \Lambda^{-1}\Lambda = \boldsymbol{I}_B$.

Conclusion 1: Fourier transform Λ is a rectangular matrix of full rank, *i.e.*, an invertible matrix.

According to (2), we can obtain $\widetilde{\mathcal{X}}$ by applying transformation matrix on each tube

$$\text{Vec}(\widetilde{\mathcal{X}}) = \Pi \text{Vec}(\mathcal{X}), \tag{3}$$

where Π can be viewed as a $B \times B$ matrix of weighted identity matrices lie in the plane, a special structure termed as *Rectangular Diagonal Blocks* (*RDB*). This structure can be obtained by

$$\Pi = \Lambda \otimes \boldsymbol{I}_{n_1 n_2}, \tag{4}$$

where $\boldsymbol{I}_{n_1 n_2}$ is an identity matrix of size $\mathbb{R}^{n_1 n_2 \times n_1 n_2}$. Furthermore, it is easy to find

$$\boldsymbol{I}_m \otimes \boldsymbol{I}_n = \boldsymbol{I}_{mn}. \tag{5}$$

In the RDB structure, we denote $\Pi_{\{b_1,b_2\}} = \Lambda(b_1,b_2) \cdot \boldsymbol{I}$ as the (b_1,b_2)-th block of Π where $(b_1,b_2) \in \{1,...,B\} \times \{1,...,B\}$, *i.e.*, each block $\Pi_{\{b_1,b_2\}}$ is an weighted identity matrix and the value of each unit in the diagonal is $\Lambda(b_1,b_2)$. We use $\Pi_{\text{diag}(n)} \in \mathbb{C}^{B \times B}$ to denote the set of the n-th diagonal unit of all the block, *i.e.*,

$$\Pi_{\text{diag}(n)} = \begin{bmatrix} \Pi_{\{1,1\}}(n,n) & \Pi_{\{1,2\}}(n,n) & \dots & \Pi_{\{1,B\}}(n,n) \\ \Pi_{\{2,1\}}(n,n) & \Pi_{\{1,2\}}(n,n) & \dots & \Pi_{\{1,B\}}(n,n) \\ \vdots & \vdots & \ddots & \vdots \\ \Pi_{\{B,1\}}(n,n) & \Pi_{\{B,2\}}(n,n) & \dots & \Pi_{\{B,B\}}(n,n) \end{bmatrix}. \tag{6}$$

Subsequently, we can conclude

$$\Pi_{\text{diag}(n)} = \Lambda, \;\; n \in \{1,...,n_1 n_2\}. \tag{7}$$

Combining (5), we have

$$\begin{aligned} \Pi^{-1}\left(\Pi_{\text{diag}(n)} \otimes \boldsymbol{I}_{n_1 n_2}\right) &= (\Lambda^{-1}\Lambda) \otimes \boldsymbol{I}_{n_1 n_2} \\ = \Lambda\Lambda^{-1} \otimes \boldsymbol{I}_{n_1 n_2} &= \boldsymbol{I}_B \otimes \boldsymbol{I}_{n_1 n_2} = \boldsymbol{I}_{n_1 n_2 B} \end{aligned} \tag{8}$$

Considering the RDB structure of Π, we can have

$$(\Lambda \otimes \boldsymbol{I}_{n_1 n_2})(\Lambda^{-1} \otimes \boldsymbol{I}_{n_1 n_2}) = \boldsymbol{I}_{n_1 n_2 B} \tag{9}$$

According to (9), we can make use of the RDB structure of Π and transfer the inverse matrix calculation of a $n_1 n_2 B \times n_1 n_2 B$ matrix to the inverse matrix calculation of a $B \times B$ matrix and the transform type can be generalized to any orthogonal transformation matrix, *e.g.*, DCT. Π is a special case for RDB structure since each block is an identity matrix. We now generalize this form to $\boldsymbol{S}$ where the value of each block's diagonal is not equal with each other. Still, making use of the RDB structure, we have the efficient computation method in the paper and a further conclusion.

Method: The calculation of $\boldsymbol{S}^{-1} \in \mathbb{C}^{n_1 n_2 B \times n_1 n_2 B}$ can be converted to the calculation of $(\boldsymbol{S}_{\text{diag}(n)})^{-1} \in \mathbb{R}^{B \times B}$, *i.e.*, $(\boldsymbol{S}^{-1})_{\text{diag}(n)} = (\boldsymbol{S}_{\text{diag}(n)})^{-1}$, for $n \in \{1,...,n_1 n_2\}$. In this way, the computation complexity can be reduced from $\mathcal{O}(n_1^3 n_2^3 B^3)$ to $n_1 n_2 \mathcal{O}(B^3)$.

Conclusion 2: The RDB matrix $\boldsymbol{S}$ is invertible if and only if $\boldsymbol{S}_{\text{diag}(n)}$ is invertible for $n \in \{1,...,n_1 n_2\}$.

In our implementation, we add $10^{-6}\boldsymbol{I}_B$ for each $\boldsymbol{S}_{\text{diag}(n)}$ to avoid the situation where *Conclusion 2* is not satisfied.

D. Model Building

Deep Tensor ADMM-Net is built on the TNN-ADMM algorithm. The algorithm is designed in tensor domain and some special operation, described in Section B, is introduced. As shown in Fig. 2 in the paper, there are multiple patterns connected within one stage and each pattern is developed from our TNN-ADMM algorithm. In this section, we will give the derivation of traditional TNN-ADMM algorithm so that a clear algorithm-model mapping is provided.

In traditional TNN-ADMM algorithm, the tensor nuclear norm is defined based on the representation domain transformed by Fourier Transform. In other words, there is only one fixed transformation and the problem minimization is formulated as

$$\underset{\mathcal{X},\ \widetilde{\mathcal{Z}}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{y}-\Phi\boldsymbol{x}\|_2^2+\|\overline{\mathcal{Z}}\|_* \quad \text{s.t. } \widetilde{\mathcal{X}}=\widetilde{\mathcal{Z}}, \tag{10}$$

where $\widetilde{\mathcal{Z}}$ is the auxiliary variable in ADMM framework. Similarly, $\widetilde{\mathcal{Z}}$ denotes the signal on the frequency domain and $\mathcal{Z}$ is not applicable. According to (10), the problem can be solved by the following recursions.

$$\mathcal{X}^k = \underset{\mathcal{X}}{\operatorname{argmin}} \left\{ \left\langle \widetilde{\mathcal{U}}^t, \widetilde{\mathcal{X}}-\widetilde{\mathcal{Z}}^t \right\rangle + \frac{\rho}{2}\|\widetilde{\mathcal{X}}-\widetilde{\mathcal{Z}}^{k-1}\|_F^2 + \frac{1}{2}\|\boldsymbol{y}-\Phi\boldsymbol{x}\|_F^2 \right\}, \tag{11}$$

$$\widetilde{\mathcal{Z}}^k = \underset{\widetilde{\mathcal{Z}}}{\operatorname{argmin}} \left\{ \frac{\rho}{2}\|\widetilde{\mathcal{X}}^k-\widetilde{\mathcal{Z}}\|_F^2 + \left\langle \widetilde{\mathcal{U}}^{k-1}, \widetilde{\mathcal{X}}^k-\widetilde{\mathcal{Z}} \right\rangle + \|\overline{\mathcal{Z}}\|_* \right\}, \tag{12}$$

$$\widetilde{\mathcal{U}}^k = \widetilde{\mathcal{U}}^{k-1} + (\widetilde{\mathcal{X}}^k - \widetilde{\mathcal{Z}}^k), \tag{13}$$

where ρ is the coefficient of Lagrange multiplier in ADMM framework. To solve the minimization task in (11) - (13), we have the following steps.

Step 1. The $\mathcal{X}$*-minimization* in (11) is a least-square projection and the closed-form solution is:

$$\mathcal{X}^k = \boldsymbol{S}^k(\Phi^\top\boldsymbol{y}+\rho^k\Lambda^k(\widetilde{\mathcal{Z}}^{k-1}-\widetilde{\mathcal{U}}^{k-1})), \tag{14}$$

$$\boldsymbol{S}^k = (\Phi^\top\Phi+\rho^k\Lambda^k\Lambda^{k\top})^{-1}, \tag{15}$$

where Π denotes the TransMat of Fourier Transformation. The subscript is eliminated to emphasis this TransMat is a fixed matrix.

Step 2. With reference to **Definition 4** and **Theorem 1** in the paper, the solution of (12), it can be described as

$$\widetilde{\mathcal{Z}}^k = \mathcal{D}_{\frac{1}{\rho}}(\widetilde{\mathcal{U}}^{k-1}+\widetilde{\mathcal{X}}^k). \tag{16}$$

Since the TNN is essentially the sum of singular values of all the frontal slices of $\mathcal{X}$, the shrinkage operator can be applied to each frontal slice of $(\widetilde{\mathcal{U}}^{k-1}+\widetilde{\mathcal{X}}^k)$ in parallel and independently.

Here, we apply FISTA algorithm in [1] to improve the converge speed by the following recursions:

$$\begin{aligned} \mathcal{Z}^{(b),t} &= \mathcal{D}_\tau(\widetilde{\mathcal{M}}^{(t)}), \\ \widetilde{\mathcal{Z}}^{(b),t+1} &= \mathcal{Z}^{(b),t} + \frac{\alpha^{(t-1)}-1}{\alpha^{(t)}}(\mathcal{Z}^{(b),t}-\mathcal{Z}^{(b),t-1}). \end{aligned} \tag{17}$$

where $\alpha^{(t)}$ is updated by

$$\alpha^{(t)} = \frac{1}{2}\Big(1+\sqrt{1+(2\alpha^{(t-1)})^2}\Big), \tag{18}$$

Step 3. The solution of *dual variable update* is intuitive according to (13). Since the Fourier transform of $\mathcal{Z}$ is required in Step 2, we also apply FISTA [1] algorithm to accelerate the convergence speed of $\mathcal{U}$ in its update

$$\begin{aligned} \mathcal{U}^{(b),t+1} &= \widetilde{\mathcal{U}}^{(b),t} + \rho(\mathcal{X}^{(b),t+1}-\mathcal{Z}^{(b),t+1}), \\ \widetilde{\mathcal{U}}^{(b),t+1} &= \mathcal{U}^{(b),t} + \frac{\alpha^{(t-1)}-1}{\alpha^{(t)}}(\mathcal{U}^{(b),t}-\mathcal{U}^{(b),t-1}). \end{aligned} \tag{19}$$

Algorithm 1 ADMM algorithm for TNN minimization : TNN-ADMM ($\boldsymbol{y}$,Φ,T)

```
Input: Measurements $\boldsymbol{y} \in \mathbb{R}^{n_1 n_2}$, sensing matrix $\Phi \in \mathbb{R}^{n_1 n_2 \times n_1 n_2 B}$, maximum number of iterations $T$.
1: Initialization:
     $\alpha^{(0)} = 1$,
     $\mathcal{X}^{(0)}, \widetilde{\mathcal{Z}}^{(0)}, \mathcal{Z}^{(0)}, \widetilde{\mathcal{U}}^{(0)}, \mathcal{U}^{(0)} \leftarrow \mathbf{0}$,
2: for $t = 1$ to $T$
     Update $\mathcal{X}^{(t+1)}$ via (14),
     Update $\widetilde{\mathcal{M}}_b^{(t)} = (\widetilde{\mathcal{X}}_b^{(t)} + \widetilde{\mathcal{U}}_b^{(t-1)}),\ b \in \{1,...,B\}$,
     Update $\alpha^{(t+1)}$ via (18)
     Update $\widetilde{\mathcal{Z}}^{(t+1)}$ via (17),
     Update $\widetilde{\mathcal{U}}^{(t+1)}$ via (19),
   end for
Output: $\mathcal{X}^{(T+1)}$
```

Taking the advantage of learning ability of neural network, we train the model on multiple transformed domains and the relationship between them, instead of just presetting the Fourier domain as our frequency representation domain. By adopting the multiply domains in the Deep Tensor ADMM-Net, we correspondingly modify the minimization task and the subsequently the solution details. Besides, the FISTA algorithm in (18) is replaced by a trainable variable η in our network.

E. Real-life Experiment Description

Our proposed neural network is applied to real-world data captured by the camera [5, 6]. The measurement of three high-speed scenes are shown in the Fig. 7. The exposure time of the camera is 33ms and the imaging systems capture a single compressed frame per 33ms. With this coded/compressed measurement, we can recover 14 or 22 frames high-speed videos. These videos of real-life scenes happen within 33ms and it is impossible for a conventional camera to capture them. In this section, we provide details of each scene, as shown in Fig. 8 - 10 in the main paper.

Wheel: Different characters are attached on a fan while the fan is doing high-speed rotation. The one-frame measurement is used to recover 14 frames.

Ball: Two plastic balls drop freely and hit the ground. Then, the rotation and rebounding happen on the two objects respectively. We recover this process in 22 frames and the result shown in Fig. 9 is selected every other frame.

Hammer: A hammer, swinging like pendulum, is used to knock down a tank and the tank falls down. The compression ratio is 22:1 and the reconstruction result shown in Fig.10 is selected every other frame.

For gray scale image in **Wheel**, tensor ADMM-Net generates clear image reconstruction with high efficiency while the degree of ghosting is reduced. For color image reconstruction, *i.e.*, **Ball** and **Hammer**, tensor ADMM-Net provides a more clear and smooth reconstruction result while much noise exists in the reconstruction of GAP-TV.

References

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[2] Fei Jiang, Xiao-Yang Liu, Hongtao Lu, and Ruimin Shen. Efficient multi-dimensional tensor sparse coding using t-linear combination. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[3] Misha E Kilmer, Karen Braman, Ning Hao, and Randy 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.

[4] Yang Liu, Xin Yuan, Jinli Suo, David Brady, and Qionghai Dai. Rank minimization for snapshot compressive imaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[5] Patrick Llull, Xuejun Liao, Xin Yuan, Jianbo Yang, David Kittle, Lawrence Carin, Guillermo Sapiro, and David J Brady. Coded aperture compressive temporal imaging. *Optics Express*, 21(9):10526–10545, 2013.

[6] Xin Yuan, Patrick Llull, Xuejun Liao, Jianbo Yang, David J Brady, Guillermo Sapiro, and Lawrence Carin. Low-cost compressive sensing for color video and depth. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3318–3325, 2014.

[7] Zemin Zhang, Gregory Ely, Shuchin Aeron, Ning Hao, and Misha 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.