Convolutional Factor Analysis Inspired Compressvie Sensing

Xin Yuan, Bell Labs

@ Beijing, ICIP 2017

September 5, 2017

Joint work with Yunchen Pu @Duke

Outline

Compressive Sensing Model

The compressive sensing (CS) problem can be formulated as:

$$\min \frac{1}{2}\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{c}\|_\star, \qquad \text{s.t.} \quad \boldsymbol{x} = \mathbf{B}\boldsymbol{c}, \tag{1}$$

where

- $\mathbf{A} \in \mathbb{R}^{M\times N}$ is the sensing matrix and usually $M \ll N$.
- $\boldsymbol{x}$ is the desired signal
- $\boldsymbol{c}$ denotes the coefficients which are sparse ($\|\cdot\|_\star = \|\cdot\|_1$) or low rank ($\|\cdot\|_\star$ symbolizes the nuclear norm)

Given $\boldsymbol{c}$, we can recover $\boldsymbol{x}$ via $\mathbf{B}$.

Motivation

- Deep learning methods, especially the convolutional and *deconvolutional* networks, have achieved excellent recognition results on benchmark datasets.
- Deconvolutional networks used in an *unsupervised manner*, aim to reconstruct the input signals as well as extracting features.
- Most existing dictionary learning algorithms learn dictionaries on *small patches*. Compressive sensing, however, usually imposes compression on the *entire image*. Similarly, the convolutional factor analysis (CFA) models learn dictionaries on *entire images*, too.

Thereby, it is feasible and appropriate to leverage this CFA technique to reconstruct desired signals $\boldsymbol{x}$ from compressed measurements $\boldsymbol{y}$.

Y. Pu, X. Yuan, A. Stevens, C. Li, and L. Carin, "A deep generative deconvolutional image model," *AISTATS* 2016

Convolutional Factor Analysis of Images

Considering the image case investigated in CS, let $\mathbf{X}_n \in \mathbb{R}^{N_x \times N_y \times N_c}$ denote the three-dimensional (3D) image, which can be a gray-scale image ($N_c = 1$), an RGB image ($N_c = 3$), or a hyperspectral image (N_c denoting the spectral channels). Under the convolutional factor model, we jointly consider N images, and for n^{th} image

$$\mathbf{X}_n = \sum_{k=1}^{K} \mathbf{D}_k * \mathbf{S}_{k,n} + \mathbf{E}_n, \tag{2}$$

where $\mathbf{D}_k \in \mathbb{R}^{n_x \times n_y \times N_c}$ is the convolutional dictionary (kernels or filters), $\mathbf{S}_{k,n} \in \mathbb{R}^{(N_x+n_x-1)\times(N_y+n_y-1)\times N_c}$ denotes the coefficients (features) and $\mathbf{E}_n$ signifies the residual or noise.
The two-dimensional convolution '$*$' is performed on each slice ($n_c = 1, \ldots, N_c$) of $\mathbf{D}_k$ and $\mathbf{S}_{k,n}$. Note the spatial size of $\mathbf{S}_{k,n}$ is $(N_x + n_x - 1) \times (N_y + n_y - 1)$ such that the image $\mathbf{X}_n$ will be of 'valid' size after convolution.

Impose Sparsity during Training

Without considering CS, the problem during training can be modeled as

$$\min \ \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{X}_n - \sum_{k=1}^{K} \mathbf{D}_k * \mathbf{S}_{k,n}\|_2^2 + \lambda\|\mathbf{S}\|_1, \tag{3}$$

where $\|\mathbf{S}\|_1 = \sum_{n,k} \|\mathbf{S}_{k,n}\|_1$. This leads to

$$\mathcal{L}(\mathbf{D}, \mathbf{S}, \lambda) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{X}_n - \sum_{k=1}^{K} \mathbf{D}_k * \mathbf{S}_{k,n}\|_2^2 + \lambda \sum_{n,k} \|\mathbf{S}_{k,n}\|_1. \tag{4}$$

Impose Sparsity during Training

Without considering CS, the problem during training can be modeled as

$$\min \ \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{X}_n - \sum_{k=1}^{K} \mathbf{D}_k * \mathbf{S}_{k,n}\|_2^2 + \lambda\|\mathbf{S}\|_1, \tag{3}$$

where $\|\mathbf{S}\|_1 = \sum_{n,k} \|\mathbf{S}_{k,n}\|_1$. This leads to

$$\mathcal{L}(\mathbf{D}, \mathbf{S}, \lambda) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{X}_n - \sum_{k=1}^{K} \mathbf{D}_k * \mathbf{S}_{k,n}\|_2^2 + \lambda \sum_{n,k} \|\mathbf{S}_{k,n}\|_1. \tag{4}$$

Introduce an auxiliary variable $\mathbf{Z}$,

$$\min \ \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{X}_n - \sum_{k=1}^{K} \mathbf{D}_k * \mathbf{S}_{k,n}\|_2^2 + \lambda\|\mathbf{Z}\|_1, \qquad \text{s.t.} \ \ \mathbf{Z} = \mathbf{S}. \tag{5}$$

Consider the Lagrange multiplier $\{\mathbf{V}, \eta\}$ and denote $\boldsymbol{s} = \mathrm{vec}(\mathbf{S}), \boldsymbol{z} = \mathrm{vec}(\mathbf{Z}), \boldsymbol{v} = \mathrm{vec}(\mathbf{V})$. This leads to

$$\begin{aligned}\mathcal{L}(\mathbf{D}, \mathbf{S}, \mathbf{Z}, \mathbf{V}, \lambda, \eta) = \tfrac{1}{2}\textstyle\sum_{n=1}^{N} \|\mathbf{X}_n - \sum_{k=1}^{K} \mathbf{D}_k * \mathbf{S}_{k,n}\|_2^2 \\ +\lambda\|\mathbf{Z}\|_1 + \tfrac{\eta}{2}\|\mathbf{S} - \mathbf{Z}\|_2^2 + \boldsymbol{v}^\top(\boldsymbol{s} - \boldsymbol{z}).\end{aligned} \tag{6}$$

Subproblems

Define $\mathbf{U} = (1/\eta)\mathbf{V}$,

$$\mathcal{L}(\mathbf{D}, \mathbf{S}, \mathbf{Z}, \mathbf{U}, \lambda, \eta) = \frac{1}{2} \sum_{n=1}^{N} \|\mathbf{X}_n - \sum_{k=1}^{K} \mathbf{D}_k * \mathbf{S}_{k,n}\|_2^2 + \lambda \|\mathbf{Z}\|_1 + \frac{\eta}{2} \|\mathbf{S} - \mathbf{Z} + \mathbf{U}\|_2^2 - \frac{\eta}{2} \|\mathbf{U}\|_2^2. \quad (7)$$

Subproblems

Define $\mathbf{U} = (1/\eta)\mathbf{V}$,

$$\mathcal{L}(\mathbf{D}, \mathbf{S}, \mathbf{Z}, \mathbf{U}, \lambda, \eta) = \frac{1}{2} \sum_{n=1}^{N} \|\mathbf{X}_n - \sum_{k=1}^{K} \mathbf{D}_k * \mathbf{S}_{k,n}\|_2^2 + \lambda \|\mathbf{Z}\|_1 + \frac{\eta}{2} \|\mathbf{S} - \mathbf{Z} + \mathbf{U}\|_2^2 - \frac{\eta}{2} \|\mathbf{U}\|_2^2. \quad (7)$$

ADMM cyclically solves (7) via the following sub problems:

$$\mathbf{D}^{t+1} := \arg\min_{\mathbf{D}} \left(\frac{1}{2} \sum_{n=1}^{N} \|\mathbf{X}_n - \sum_{k=1}^{K} \mathbf{D}_k * \mathbf{S}_{k,n}\|_2^2 \right) \quad (8)$$

$$\mathbf{S}^{t+1} := \arg\min_{\mathbf{S}} \left(\frac{1}{2} \sum_{n=1}^{N} \|\mathbf{X}_n - \sum_{k=1}^{K} \mathbf{D}_k * \mathbf{S}_{k,n}\|_2^2 + \frac{\eta}{2} \|\mathbf{S} - \mathbf{Z} + \mathbf{U}\|_2^2 \right) \quad (9)$$

$$\mathbf{Z}^{t+1} := \arg\min_{\mathbf{Z}} \left(\lambda \|\mathbf{Z}\|_1 + \frac{\eta}{2} \|\mathbf{S} - \mathbf{Z} + \mathbf{U}\|_2^2 \right) \quad (10)$$

$$\mathbf{U}^{t+1} := \mathbf{U}^t + \eta(\mathbf{S} - \mathbf{Z}) \quad (11)$$

where t denotes the iteration.

Solve $\mathbf{D}$

Solve $\mathbf{D}$: $\boldsymbol{x}_n = \mathrm{vec}(\mathbf{X}_n)$, $\boldsymbol{d}_k = \mathrm{vec}(\mathbf{D}_k)$, $\mathbf{D}_k * \mathbf{S}_{k,n} = \mathbf{F}_{k,n}\boldsymbol{d}_k$, $\boldsymbol{x}_n^{-k} = \boldsymbol{x}_n - \sum_{k'=1,k'\neq k}^{K} \mathbf{F}_{k',n}\boldsymbol{d}_{k'}$,

$$\boldsymbol{d}_k^{t+1} = \boldsymbol{d}_k^t + \beta \sum_{n=1}^{N} \mathbf{F}_{k,n}^\top (\boldsymbol{x}_n^{-k} - \mathbf{F}_{k,n}\boldsymbol{d}_k), \tag{12}$$

Solve $\mathbf{D}$

Solve $\mathbf{D}$: $\boldsymbol{x}_n = \mathrm{vec}(\mathbf{X}_n)$, $\boldsymbol{d}_k = \mathrm{vec}(\mathbf{D}_k)$, $\mathbf{D}_k * \mathbf{S}_{k,n} = \mathbf{F}_{k,n}\boldsymbol{d}_k$, $\boldsymbol{x}_n^{-k} = \boldsymbol{x}_n - \sum_{k'=1,k'\neq k}^{K} \mathbf{F}_{k',n}\boldsymbol{d}_{k'}$,

$$\boldsymbol{d}_k^{t+1} = \boldsymbol{d}_k^t + \beta \sum_{n=1}^{N} \mathbf{F}_{k,n}^\top (\boldsymbol{x}_n^{-k} - \mathbf{F}_{k,n}\boldsymbol{d}_k), \tag{12}$$

Implementation details in MATLAB:

$$\mathbf{F}_{k,n}\boldsymbol{d}_k = \mathbf{D}_k * \mathbf{S}_{k,n} = \mathrm{conv2}(\mathbf{S}_{k,n}, \mathbf{D}_k, \text{'valid'}) \tag{13}$$

$$\mathbf{F}_{k,n}^\top \boldsymbol{x}_n = \mathrm{conv2}(\mathrm{rot90}(\mathbf{S}_{k,n}, 2), \mathbf{X}_n, \text{'valid'}) \tag{14}$$

$$\mathbf{T}_k \boldsymbol{s}_{k,n} = \mathbf{D}_k * \mathbf{S}_{k,n} = \mathrm{conv2}(\mathbf{S}_{k,n}, \mathbf{D}_k, \text{'valid'}) \tag{15}$$

$$\mathbf{T}_k^\top \boldsymbol{x}_n = \mathrm{conv2}(\mathbf{X}_n, \mathrm{rot90}(\mathbf{D}_k, 2), \text{'full'}) \tag{16}$$

We also found that using "fft2( )" in MATLAB is at least 4$\times$ faster than "conv2( )" by providing almost the same results.

Solve $\mathbf{S}$ and $\mathbf{Z}$

- Eq. (9) is a quadratic optimization problem. $\boldsymbol{s}_{k,n} = \mathrm{vec}(\mathbf{S}_{k,n})$, $\mathbf{D}_k * \mathbf{S}_{k,n} = \mathbf{T}_k \boldsymbol{s}_{k,n}$, $\boldsymbol{x}_n^{-k} = \boldsymbol{x}_n - \sum_{k'=1,k'\neq k}^{K} \mathbf{T}_{k'} \boldsymbol{s}_{k',n}$.
  Given $\{\mathbf{T}, \boldsymbol{z}, \boldsymbol{u}\}$, the optimal $\boldsymbol{s}_{k,n}$ is the solution of the following linear system
  $$(\mathbf{T}_k^\top \mathbf{T}_k + \eta \mathbf{I}) \boldsymbol{s}_{k,n} = \mathbf{T}_k^\top \boldsymbol{x}_n^{-k} + \eta(\boldsymbol{z}_{k,n} - \boldsymbol{u}_{k,n}), \tag{17}$$
  which can be solved effectively using conjugate gradient (CG) algorithms.
- Eq. (10) can be solved via the shrinkage operator, *i.e.*, soft thresholding
  $$\mathbf{Z}^{t+1} = \mathrm{soft}(\mathbf{S}^t + \frac{\mathbf{U}^t}{\eta}, \boldsymbol{\gamma}^t), \tag{18}$$
  which can be performed element-wise, *i.e.*, for i^{th} element,
  $$z_i^{t+1} = \mathrm{sign}(s_i^t + \frac{u_i^t}{\eta})\ \max(|s_i^t + \frac{u_i^t}{\eta}| - \gamma_i^t, 0). \tag{19}$$

Extend the CFA model to CS

In Compressive Sensing:

$$\boldsymbol{y}_n = \mathbf{A}\boldsymbol{x}_n, \tag{20}$$

where $\boldsymbol{x}_n$ denotes the n^{th} vectorized image.

$$\min \quad \frac{1}{2}\sum_{n=1}^{N} \|\boldsymbol{y}_n - \mathbf{A}\boldsymbol{x}_n\|_2^2 + \lambda \sum_{n=1}^{N}\sum_{k=1}^{K} \|\boldsymbol{s}_{k,n}\|_1, \tag{21}$$

$$\text{s.t.} \quad \boldsymbol{x}_n = \textstyle\sum_{k=1}^{K} \mathbf{T}_k \boldsymbol{s}_{k,n}. \tag{22}$$

This leads to the following objective function

$$\mathcal{L}(\mathbf{T}, \mathbf{S}, \lambda) = \frac{1}{2}\sum_{n=1}^{N} \|\boldsymbol{y}_n - \mathbf{A}\sum_{k=1}^{K} \mathbf{T}_k \boldsymbol{s}_{k,n}\|_2^2 + \lambda \sum_{n,k} \|\boldsymbol{s}_{k,n}\|_1. \tag{23}$$

Two Models of CS-CFA

We consider to solve (23) in two ways:

- The convolutional dictionary ($\{\mathbf{D}_k\}_{k=1}^K$ or $\{\mathbf{T}_k\}_{k=1}^K$) is pre-learned by training data. In this case, (23) aims to solve

$$\arg\min_{\{\boldsymbol{s}\}} \frac{1}{2}\sum_{n=1}^{N}\|\boldsymbol{y}_n - \mathbf{A}\sum_{k=1}^{K}\mathbf{T}_k\boldsymbol{s}_{k,n}\|_2^2 + \lambda\sum_{n,k}\|\boldsymbol{s}_{k,n}\|_1. \tag{24}$$

- The convolutional dictionary ($\{\mathbf{D}_k\}_{k=1}^K$ or $\{\mathbf{T}_k\}_{k=1}^K$) is unknown *a priori* and will be learned *in situ* from the raw measurements $\{\boldsymbol{y}_n\}_{n=1}^N$. In this case, (23) aims to solve

$$\arg\min_{\{\mathbf{T},\boldsymbol{s}\}} \frac{1}{2}\sum_{n=1}^{N}\|\boldsymbol{y}_n - \mathbf{A}\sum_{k=1}^{K}\mathbf{T}_k\boldsymbol{s}_{k,n}\|_2^2 + \lambda\sum_{n,k}\|\boldsymbol{s}_{k,n}\|_1. \tag{25}$$

CS Inversion with Pre-Learned Convolutional Dictionary

$$\min \frac{1}{2} \sum_{n=1}^{N} \|\boldsymbol{y}_n - \mathbf{A} \sum_{k=1}^{K} \mathbf{T}_k \boldsymbol{s}_{k,n}\|_2^2 + \lambda \sum_{k,n} \|\boldsymbol{z}_{k,n}\|_1, \quad \text{s.t. } \boldsymbol{s}_{k,n} = \boldsymbol{z}_{k,n}, \quad \forall k, n.$$

This results in the objective function

$$\begin{aligned}\mathcal{L}(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{u}, \lambda, \eta) = {} & \tfrac{1}{2} \textstyle\sum_{n=1}^{N} \|\boldsymbol{y}_n - \mathbf{A} \sum_{k=1}^{K} \mathbf{T}_k \boldsymbol{s}_{k,n}\|_2^2 \\ & + \lambda \sum_{k,n} \|\boldsymbol{z}_{k,n}\|_1 + \frac{\eta}{2} \sum_{k,n} \|\boldsymbol{s}_{k,n} - \boldsymbol{z}_{k,n} + \boldsymbol{u}_{k,n}\|_2^2 + \text{Const.}\end{aligned}$$

CS Inversion with Pre-Learned Convolutional Dictionary

$$\min \frac{1}{2} \sum_{n=1}^{N} \|\boldsymbol{y}_n - \mathbf{A} \sum_{k=1}^{K} \mathbf{T}_k \boldsymbol{s}_{k,n}\|_2^2 + \lambda \sum_{k,n} \|\boldsymbol{z}_{k,n}\|_1, \quad \text{s.t. } \boldsymbol{s}_{k,n} = \boldsymbol{z}_{k,n}, \quad \forall k, n.$$

This results in the objective function

$$\begin{aligned}\mathcal{L}(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{u}, \lambda, \eta) = \tfrac{1}{2} \textstyle\sum_{n=1}^{N} \|\boldsymbol{y}_n - \mathbf{A} \sum_{k=1}^{K} \mathbf{T}_k \boldsymbol{s}_{k,n}\|_2^2 \\ + \lambda \sum_{k,n} \|\boldsymbol{z}_{k,n}\|_1 + \frac{\eta}{2} \sum_{k,n} \|\boldsymbol{s}_{k,n} - \boldsymbol{z}_{k,n} + \boldsymbol{u}_{k,n}\|_2^2 + \text{Const.}\end{aligned}$$

ADMM cyclically solves this via the following sub-problems:

$$\begin{aligned}\boldsymbol{s}^{t+1} := \arg\min_{\boldsymbol{s}} \big(\tfrac{1}{2} \textstyle\sum_{n=1}^{N} \|\boldsymbol{y}_n - \mathbf{A} \sum_{k=1}^{K} \mathbf{T}_k \boldsymbol{s}_{k,n}\|_2^2 \\ + \tfrac{\eta}{2} \textstyle\sum_{n,k} \|\boldsymbol{s}_{k,n} - \boldsymbol{z}_{k,n} + \boldsymbol{u}_{k,n}\|_2^2\big) \qquad (26)\end{aligned}$$

$$\boldsymbol{z}^{t+1} := \arg\min_{\boldsymbol{z}} \big(\lambda \sum_{k,n} \|\boldsymbol{z}_{k,n}\|_1 + \tfrac{\eta}{2} \textstyle\sum_{k,n} \|\boldsymbol{s}_{k,n} - \boldsymbol{z}_{k,n} + \boldsymbol{u}_{k,n}\|_2^2\big) \qquad (27)$$

$$\boldsymbol{u}^{t+1} := \boldsymbol{u}^{t} + \eta(\boldsymbol{s} - \boldsymbol{z}) \qquad (28)$$

Solve $\mathbf{S}$

Via defining

$$\boldsymbol{y}_n^{-k} \stackrel{\text{def}}{=} \boldsymbol{y}_n - \mathbf{A} \sum_{k'=1, k' \neq k}^{K} \mathbf{T}_{k'} \boldsymbol{s}_{k',n},$$

given $\{\mathbf{T}, \boldsymbol{z}, \boldsymbol{u}, \mathbf{A}\}$, the optimal $\boldsymbol{s}_{k,n}$ is the solution of the following linear system

$$(\mathbf{T}_k^\top \mathbf{A}^\top \mathbf{A} \mathbf{T}_k + \eta \mathbf{I}) \boldsymbol{s}_{k,n} = \mathbf{T}_k^\top \mathbf{A}^\top \boldsymbol{y}_n^{-k} + \eta (\boldsymbol{z}_{k,n} - \boldsymbol{u}_{k,n}). \tag{29}$$

Again, this can be solved effectively via CG algorithms.

CS Inversion by Learning Convolutional Dictionary from Measurements

$$\mathcal{L}(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{u}, \mathbf{T}, \lambda, \eta) = \tfrac{1}{2} \sum_{n=1}^{N} \|\boldsymbol{y}_n - \mathbf{A} \sum_{k=1}^{K} \mathbf{T}_k \boldsymbol{s}_{k,n}\|_2^2 + \lambda \sum_{k,n} \|\boldsymbol{z}_{k,n}\|_1 + \frac{\eta}{2} \sum_{k,n} \|\boldsymbol{s}_{k,n} - \boldsymbol{z}_{k,n} + \boldsymbol{u}_{k,n}\|_2^2 + \text{Const.}$$

Recalling the definition in (13), in addition to the subproblems described in (26)-(28), we also need to update $\{\mathbf{D}_k\}_{k=1}^K$,

$$\boldsymbol{d}^{t+1} := \arg\min_{\boldsymbol{d}} \left(\frac{1}{2} \sum_{n=1}^{N} \|\boldsymbol{y}_n - \mathbf{A} \sum_{k=1}^{K} \mathbf{F}_{k,n} \boldsymbol{d}_k\|_2^2\right). \tag{30}$$

Similar to (8), this can be solved by the gradient descent:

$$\boldsymbol{d}_k^{t+1} = \boldsymbol{d}_k^t + \beta \sum_{n=1}^{N} \mathbf{F}_{k,n}^\top \mathbf{A}^\top (\boldsymbol{y}_n^{-k} - \mathbf{A}\mathbf{F}_{k,n}\boldsymbol{d}_k), \tag{31}$$

where $\boldsymbol{y}_n^{-k} = \boldsymbol{y}_n - \mathbf{A} \sum_{k'=1, k' \neq k}^{K} \mathbf{F}_{k',n} \boldsymbol{d}_{k'}$.

CS-CFA

Require: Input measurements $\{\boldsymbol{y}_n\}_{n=1}^{N}$, sensing matrix $\mathbf{A}$, parameters $\{\beta, \eta\}$

Initialize $\mathbf{D}, \mathbf{S}, \mathbf{Z}, \mathbf{U}$.
for $t = 1$ **to** MaxIter **do**
    Update $\mathbf{D}$ by Eq. (31).
    Update $\mathbf{S}$ by Eq. (29).
    Update $\mathbf{Z}$ by shrinkage operator, Eq. (19).
    Update $\mathbf{U}$ by Eq. (28).
end for

Flow Chart of Joint Model

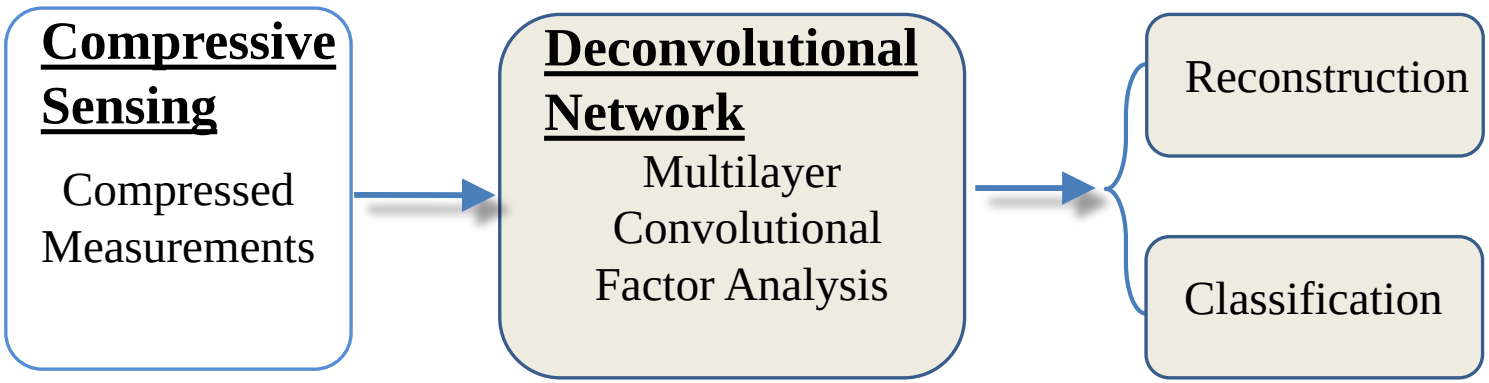

Figure: Flow chart of Joint Model.

Joint Model Using Softmax

For joint classification and reconstruction task, we have labeled (compressed) data $\{\boldsymbol{y}_n, c_n\}_{n=1}^{N}$, where $c_n = \{1, \ldots, C\}$ considering C classes in total. Introducing the classifier weight matrix $\mathbf{H} \in \mathbb{R}^{C \times N_s}$ and the bias vector $\boldsymbol{\alpha} \in \mathbb{R}^{N_s}$, we have

$$p(c_n = i | \boldsymbol{s}_n, \mathbf{H}, \boldsymbol{\alpha}) = \mathrm{softmax}_i(\boldsymbol{h}_i \boldsymbol{s}_n^{(L)} + \boldsymbol{\alpha}) = \frac{\exp(\boldsymbol{h}_i \boldsymbol{s}_n^{(L)} + \alpha_i)}{\sum_j \exp(\boldsymbol{h}_j \boldsymbol{s}_n^{(L)} + \alpha_j)}, \quad (32)$$

where $\boldsymbol{h}_i$ denotes the i^{th} row of the weight matrix $\mathbf{H}$ and α_i symbolizes the i^{th} element of the vector $\boldsymbol{\alpha}$. These weights $\{\mathbf{H}, \boldsymbol{\alpha}\}$ can be learned jointly with the CFA network, thus constituting a supervised CS-CFA model. Similarly, a C-class SVM can also be used.

Reconstruction

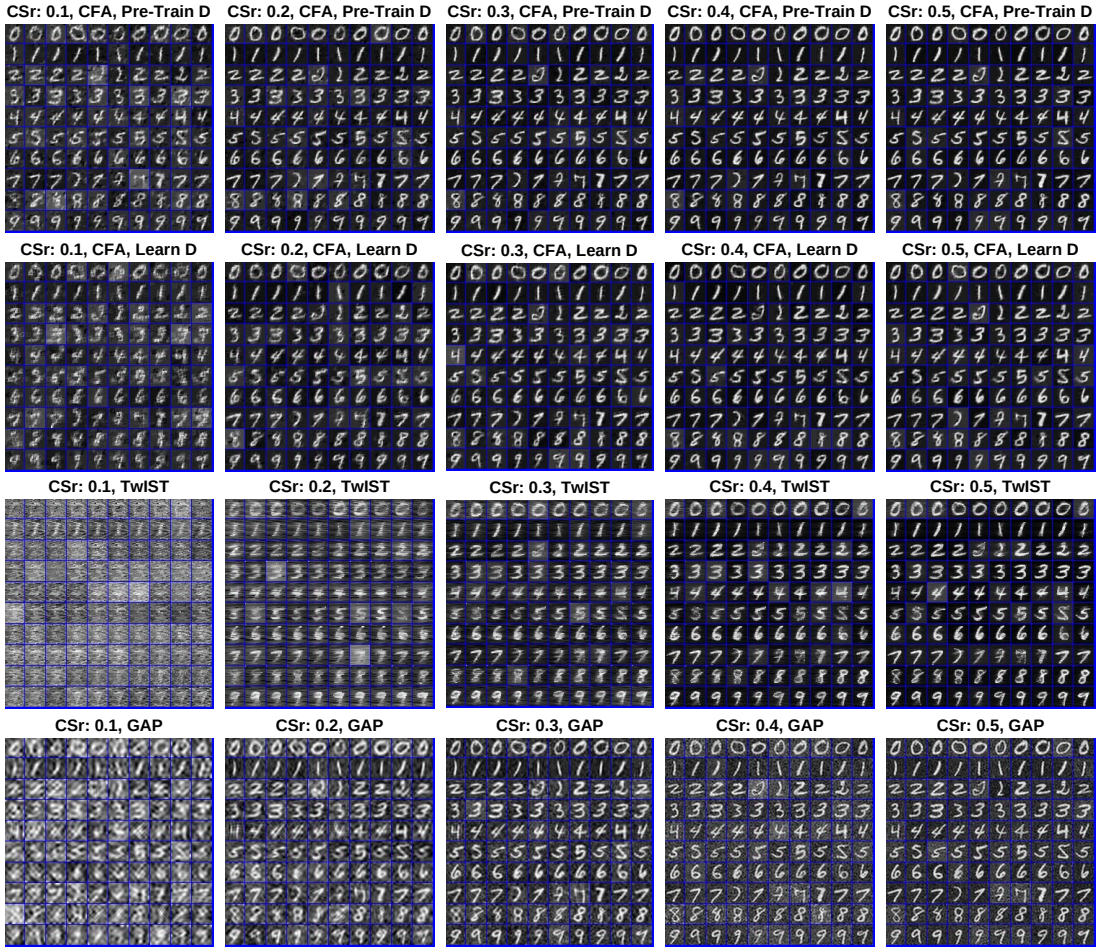

Figure: Reconstruction results of MNIST dataset at various CSr. Row 1: proposed CS-CFA with pre-trained dictionary; row 2: proposed CS-CFA with *in situ* learned dictionary (from the measurements directly); row 3: TwIST; row 4: GAP.

Reconstruction: Compare with Others

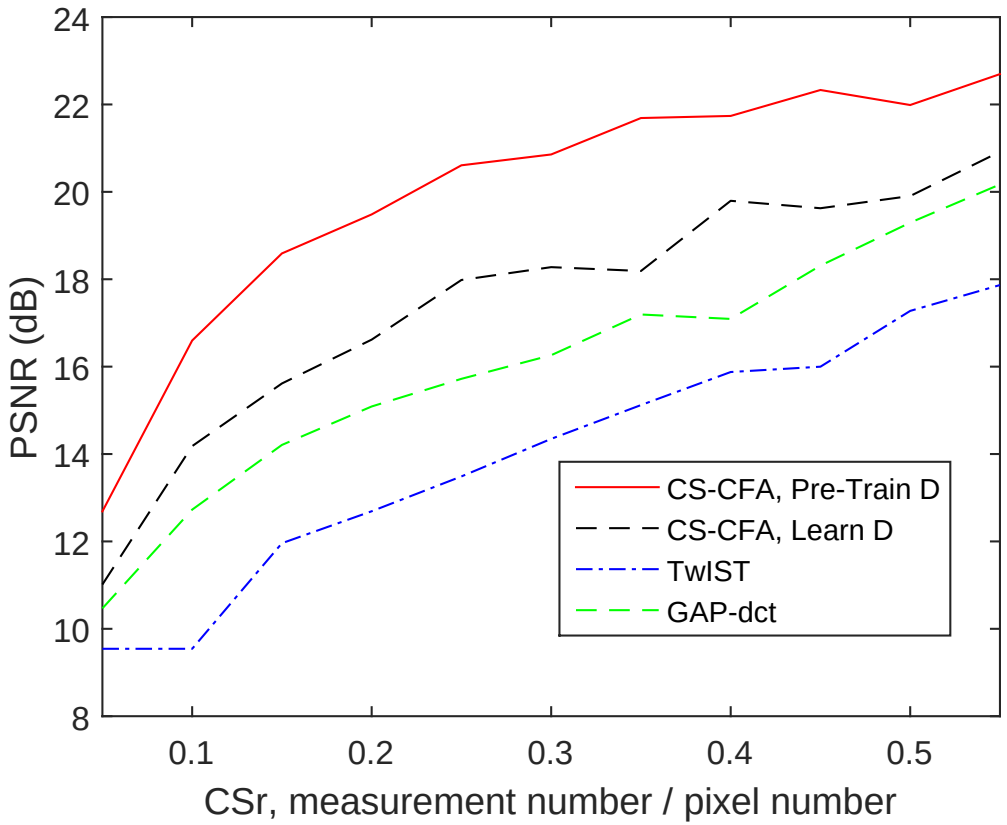

Figure: Average PSNR of reconstructed results versus CSr with different algorithms on MNIST dataset.

Joint Reconstrcution and Classification

Table: Reconstruction PSNR (dB) and classification accuracy (%) at various CSr on MNIST.

CSr	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	no CS
Rec. PSNR	15.65	19.73	21.55	22.51	23.53	24.02	24.22	24.63	24.84	-
Softmax	75.88	90.88	92.89	93.78	94.47	94.46	94.35	94.38	94.39	93.84
Linear SVM	70.30	88.57	91.64	93.45	93.79	93.76	93.12	93.52	93.56	92.13
Nonlinear SVM	77.62	92.38	94.93	95.29	96.37	96.77	96.84	96.82	96.79	96.32

Q & A

xyuan@bell-labs.com
https://www.bell-labs.com/usr/x.yuan