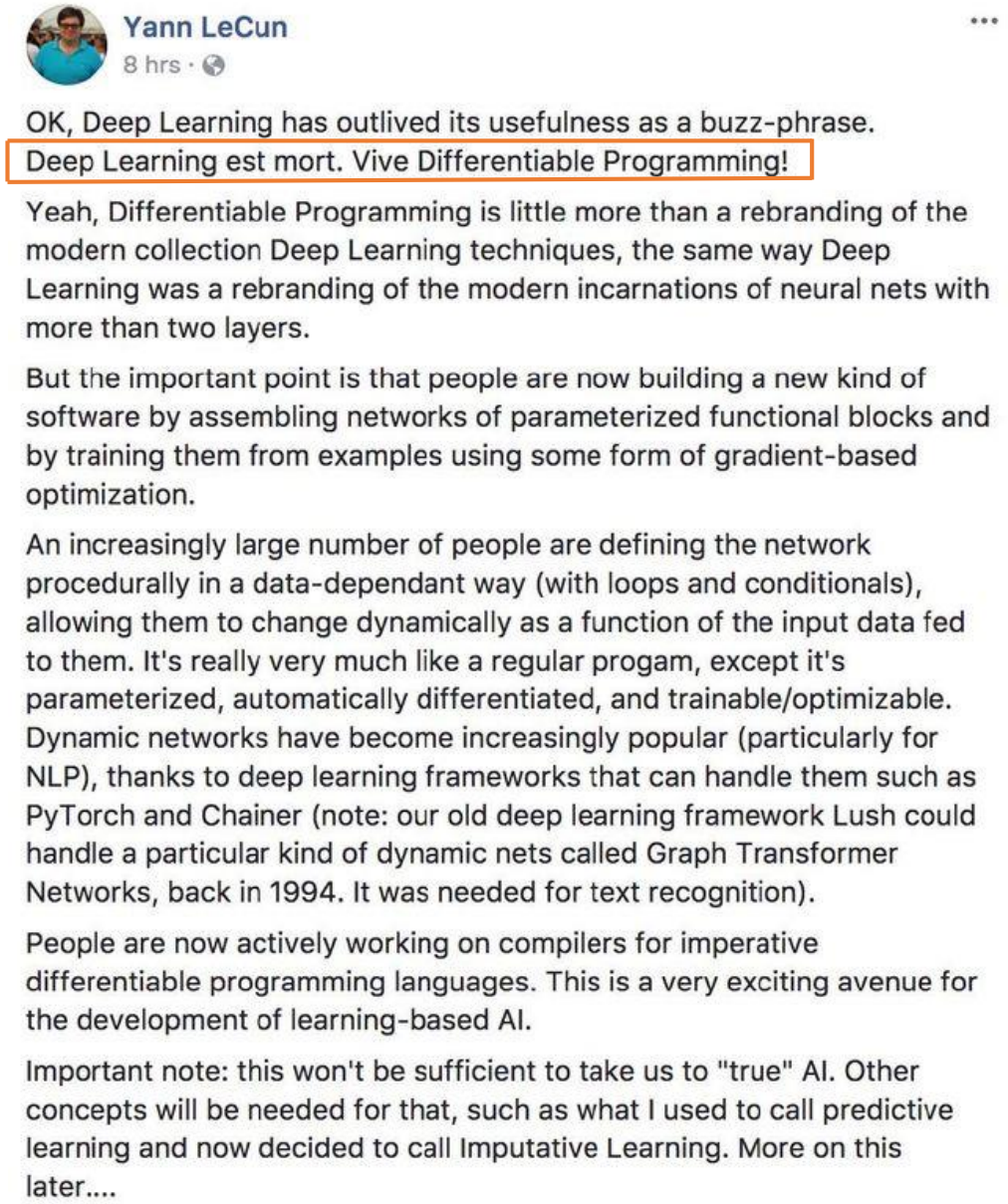
Model-driven deep learning

—— differentiable programming

2021/08/17 梁浩

Background



Yann LeCun
8 hrs ·

OK, Deep Learning has outlived its usefulness as a buzz-phrase.
Deep Learning est mort. Vive Differentiable Programming!

Yeah, Differentiable Programming is little more than a rebranding of the modern collection Deep Learning techniques, the same way Deep Learning was a rebranding of the modern incarnations of neural nets with more than two layers.

But the important point is that people are now building a new kind of software by assembling networks of parameterized functional blocks and by training them from examples using some form of gradient-based optimization.

An increasingly large number of people are defining the network procedurally in a data-dependant way (with loops and conditionals), allowing them to change dynamically as a function of the input data fed to them. It's really very much like a regular progam, except it's parameterized, automatically differentiated, and trainable/optimizable. Dynamic networks have become increasingly popular (particularly for NLP), thanks to deep learning frameworks that can handle them such as PyTorch and Chainer (note: our old deep learning framework Lush could handle a particular kind of dynamic nets called Graph Transformer Networks, back in 1994. It was needed for text recognition).

People are now actively working on compilers for imperative differentiable programming languages. This is a very exciting avenue for the development of learning-based AI.

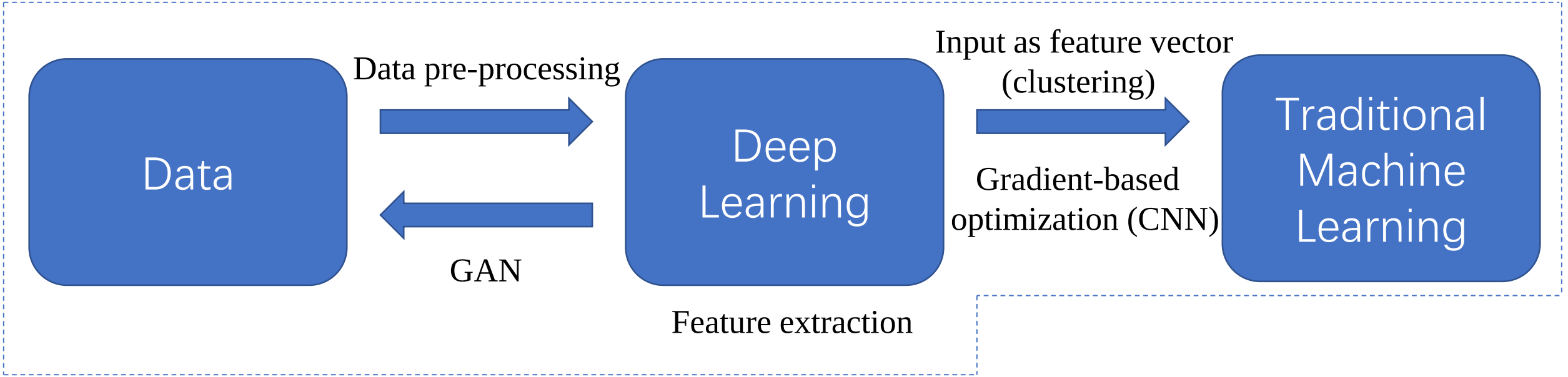
Important note: this won't be sufficient to take us to "true" AI. Other concepts will be needed for that, such as what I used to call predictive learning and now decided to call Imputative Learning. More on this later....

Model-driven deep learning (differentiable programming):

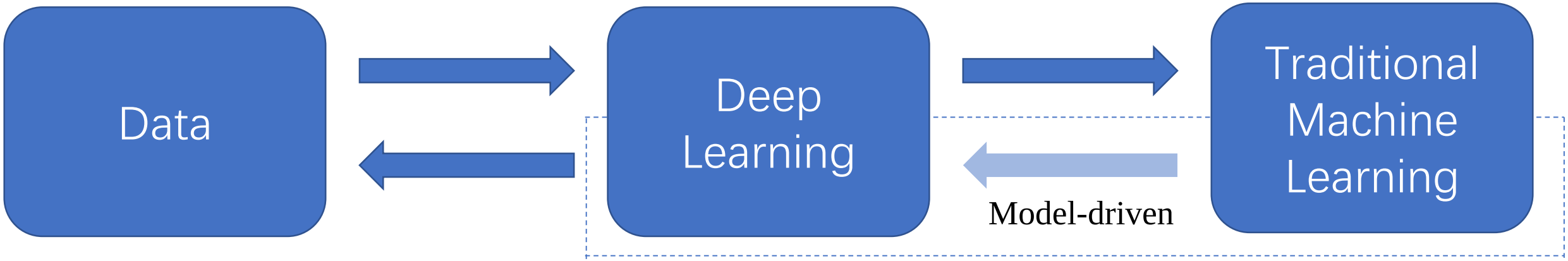
treats neural network as a language

- Deep learning: a standard data-driven method, which uses networks as a black box to rely on large amounts of data to solve problems
- Model-driven method: forms a cost function from the goal, mechanism and a prior, and then solve problems by minimizing the cost function

Black box, only data-driven



Model-driven deep learning — try to transform traditional machine learning methods into equivalent neural networks



- Strong interpretability of traditional methods
- Superior fitting capabilities of deep neural networks

The general procedure of model-driven deep learning[1]:

- Construct a model family based on the task backgrounds
- Design an algorithm family for solving the model family, and establish the corresponding convergence theory
- The algorithm family is unfolded to a deep network with which parameter learning is performed as a deep leaning approach

[1] Xu Z, Sun J. Model-driven deep-learning [J]. National Science Review, 2018, 5(1): 22-24.

Learning Fast Approximations of Sparse Coding

Karol Gregor and Yann LeCun {KGREGOR,YANN}@CS.NYU.EDU
Courant Institute, New York University, 715 Broadway, New York, NY 10003, USA

ICML 2010

Sparse coding:

For a given input vector $\boldsymbol{X} \in R^n$, to find the optimal sparse code vector $\boldsymbol{Z}^* \in R^m$ $(m > n)$ that minimizes an energy function that combines the square reconstruction error and an l_1 sparsity penalty on the code:

$$\boldsymbol{Z}^* = \underset{\boldsymbol{Z}}{\operatorname{argmin}} \frac{1}{2} ||\boldsymbol{X} - \boldsymbol{W}_d \boldsymbol{Z}||_2^2 + \alpha ||\boldsymbol{Z}||_1$$

where $\boldsymbol{W}_d$ denotes an $n \times m$ dictionary matrix.

ISTA algorithm[1]:

Algorithm 1 ISTA

function ISTA(X, Z, W_d, α, L)

Require: $L >$ largest eigenvalue of $W_d^T W_d$.

Initialize: $Z = 0$,

repeat

$Z = h_{(\alpha/L)}(Z - \frac{1}{L} W_d^T (W_d Z - X))$

until change in Z below a threshold

end function

$$\boldsymbol{Z}^{(k+1)} = h_\theta\left(\boldsymbol{W}_e \boldsymbol{X} + \boldsymbol{S}\boldsymbol{Z}^{(k)}\right)$$

$$\boldsymbol{W}_e = \frac{1}{L}\boldsymbol{W}_d^T \qquad \theta = \alpha/\mathrm{L}$$

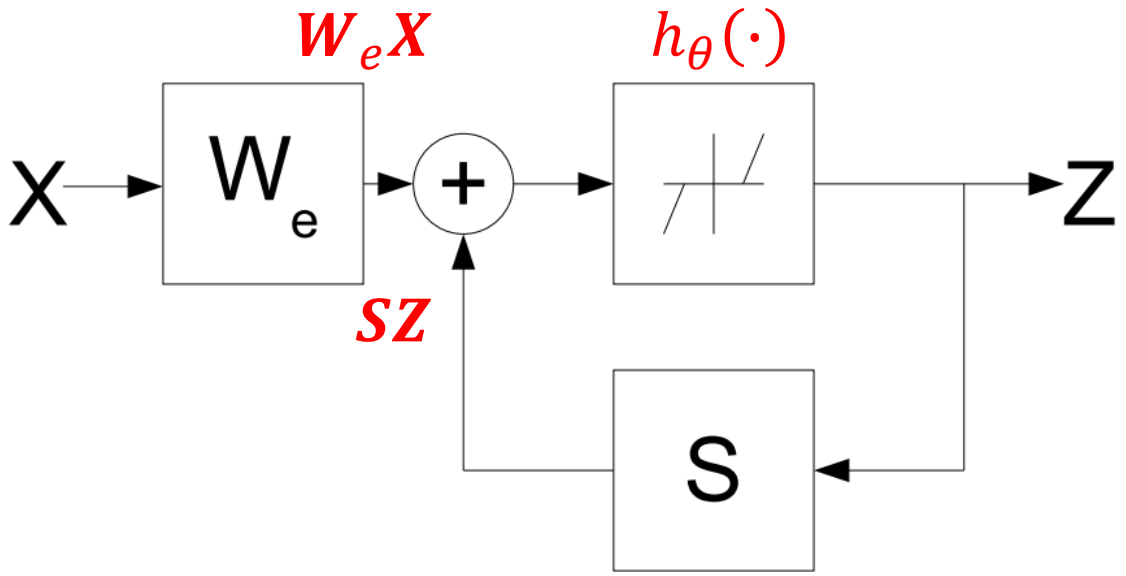
$$\boldsymbol{S} = \boldsymbol{I} - \frac{1}{L}\boldsymbol{W}_d^T \boldsymbol{W}_d$$

$$[h_\theta(\boldsymbol{V})]_i = \mathrm{sign}(V_i)\,(|V_i| - \theta_i)_+$$

[1] I. Daubechies, M. Defrise, and C. D. Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, Comm. Pure Appl. Math., 57 (2004), pp. 1413–1457.

ISTA block diagram:

$$\boldsymbol{Z}^{(k+1)} = h_\theta\left(\boldsymbol{W}_e\boldsymbol{X} + \boldsymbol{S}\boldsymbol{Z}^{(k)}\right)$$

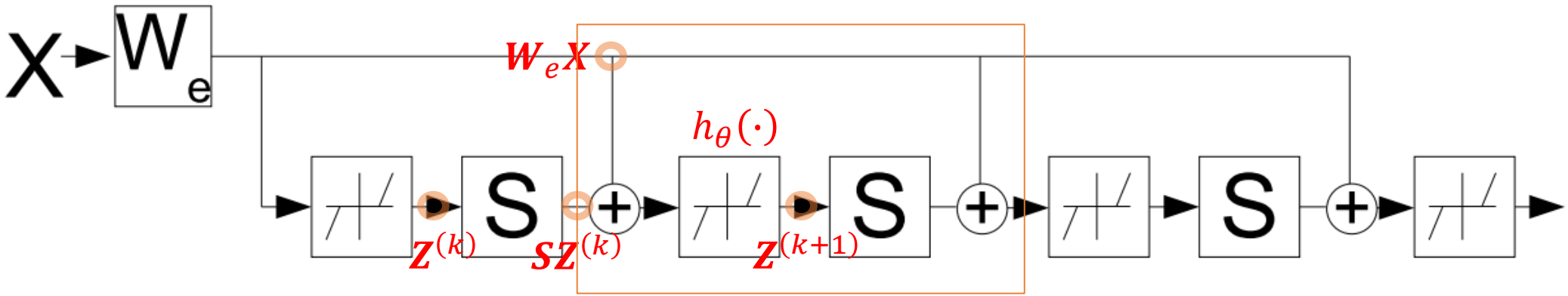


How to transform the ISTA to a neural network?

LISTA:

$$\boldsymbol{Z}^{(k+1)} = h_\theta\left(\boldsymbol{W}_e\boldsymbol{X} + \boldsymbol{S}\boldsymbol{Z}^{(k)}\right)$$

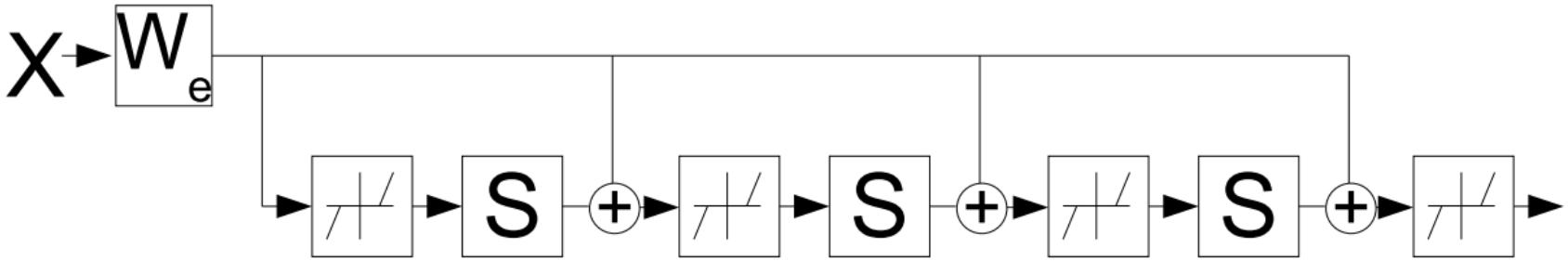
Time-unfolded version of the ISTA block diagram, truncated to a fixed number of iterations



Learnable parameters: $\boldsymbol{W}_e$, $\boldsymbol{S}$ and θ

LISTA:

$$\boldsymbol{Z}^{(k+1)} = h_\theta\left(\boldsymbol{W}_e\boldsymbol{X} + \boldsymbol{S}\boldsymbol{Z}^{(k)}\right)$$



Let $\boldsymbol{W} = (\boldsymbol{W}_e, \boldsymbol{S}, \theta)$, and the LISTA as $\boldsymbol{Z} = f_e(\boldsymbol{X}, \boldsymbol{W})$.

Loss function: the squared error between the predicted code and the optimal code averaged over a training set $(\boldsymbol{X}^1, \ldots, \boldsymbol{X}^P)$

$$L(\boldsymbol{W}) = \frac{1}{P}\sum_{p=1}^{P} L(\boldsymbol{W}, \boldsymbol{X}^p) \quad \text{with}$$

$$L(\boldsymbol{W}, \boldsymbol{X}^p) = \frac{1}{2}||\boldsymbol{Z}^{*p} - f_e(\boldsymbol{W}, \boldsymbol{X}^p)||_2^2$$

Numerical results:

Table 1. Prediction error (squared error between the optimal code and the predicted codes) for different non-linearities of the baseline encoder (7), and for LISTA and FISTA. LISTA produces a much better estimate after one iteration than FISTA.

NON-LINEARITY	100 UNITS	400 UNITS
$D\tanh(x)$	8.6	10.7
$D(\tanh(x+u)+\tanh(x-u))$	3.33	4.62
$h_\alpha(x)$	3.29	4.82
LISTA 1 ITERATION	1.50	2.45
LISTA 3 ITERATIONS	0.98	2.12
LISTA 7 ITERATIONS	0.52	1.62
FISTA 1 ITERATION	21.	22.

Baseline encoder:

$$Z = g(W_e X) \qquad (7)$$

More interestingly LISTA is stupendously better than FISTA: It takes 18 iterations of FISTA to reach the same error obtained with 1 iteration of LISTA for m = 100, and 35 iteration for m = 400.

Deep ADMM-Net for Compressive Sensing MRI

Yan Yang
Xi'an Jiaotong University
yangyan92@stu.xjtu.edu.cn

Jian Sun
Xi'an Jiaotong University
jiansun@mail.xjtu.edu.cn

Huibin Li
Xi'an Jiaotong University
huibinli@mail.xjtu.edu.cn

Zongben Xu
Xi'an Jiaotong University
zbxu@mail.xjtu.edu.cn

NIPS 2016

Compressive sensing magnetic resonance imaging (MRI):

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 + \sum_{l=1}^{L} \lambda_l g(\boldsymbol{D}_l \boldsymbol{x}) \right\}$$

$\boldsymbol{x} \in C^N$: the MRI image to be reconstructed

$\boldsymbol{y} \in C^{N'} (N' < N)$: the under-sampled k-space data

$\boldsymbol{A} = \boldsymbol{P}\boldsymbol{F} \in C^{N' \times N}$: the measurement matrix ($\boldsymbol{P}$ the under-sampling matrix, and $\boldsymbol{F}$ the Fourier transform matrix)

$\boldsymbol{D}_l$: the transform matrix for a filtering operation, e.g., discrete wavelet transform, discrete cosine transform, etc.

$g(\cdot)$: the regularization function derived from the data prior, e.g., l_1-norm, l_p-norm

λ_l: the regularization parameter

General CS-MRI model:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 + \sum_{l=1}^{L} \lambda_l g(\boldsymbol{D}_l \boldsymbol{x}) \right\}$$

Challenge

- How to choose an optimal image transform domain/subspace and the corresponding sparse regularization?
- How to determine the optimal parameters?

ADMM algorithm[1]:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 + \sum_{l=1}^{L} \lambda_l g(\boldsymbol{D}_l \boldsymbol{x}) \right\}$$

Introducing auxiliary variables $\boldsymbol{z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_L\}$

$$\min_{\boldsymbol{x},\boldsymbol{z}} \frac{1}{2} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 + \sum_{l=1}^{L} \lambda_l g(\boldsymbol{z}_l)$$

$$\text{s.t.} \quad \boldsymbol{z}_l = \boldsymbol{D}_l \boldsymbol{x}, \qquad \forall l \in [1,2,\cdots,L]$$

$\boldsymbol{\alpha} = \{\boldsymbol{\alpha}_l\}$ the Lagrangian multipliers

$\rho = \{\rho_l\}$ the penalty parameters

Its augmented Lagrangian function is

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha}) = \frac{1}{2} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 + \sum_{l=1}^{L} \lambda_l g(\boldsymbol{z}_l) - \sum_{l=1}^{L} \langle \boldsymbol{\alpha}_l, \boldsymbol{z}_l - \boldsymbol{D}_l x \rangle + \sum_{l=1}^{L} \frac{\rho_l}{2} ||\boldsymbol{z}_l - \boldsymbol{D}_l \boldsymbol{x}||_2^2$$

[1] Boyd S, Parikh N, Chu E. Distributed optimization and statistical learning via the alternating direction method of multipliers [M]. Now Publishers Inc, 2011.

$$\mathcal{L}_\rho(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}||_2^2 + \sum_{l=1}^{L}\lambda_l g(\boldsymbol{z}_l) - \sum_{l=1}^{L}\langle \boldsymbol{\alpha}_l, \boldsymbol{z}_l - \boldsymbol{D}_l x\rangle + \sum_{l=1}^{L}\frac{\rho_l}{2}||\boldsymbol{z}_l - \boldsymbol{D}_l\boldsymbol{x}||_2^2$$

ADMM alternatively optimizes $\{\boldsymbol{x},\boldsymbol{z},\boldsymbol{\alpha}\}$ by solving the three subproblems

$$\begin{cases} \boldsymbol{x}^{(n+1)} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\frac{1}{2}||\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}||_2^2 - \sum_{l=1}^{L}\left\langle \boldsymbol{\alpha}_l^{(n)}, \boldsymbol{z}_l^{(n)} - \boldsymbol{D}_l x\right\rangle + \sum_{l=1}^{L}\frac{\rho_l}{2}||\boldsymbol{z}_l^{(n)} - \boldsymbol{D}_l\boldsymbol{x}||_2^2 \\ \boldsymbol{z}^{(n+1)} = \underset{\boldsymbol{z}}{\operatorname{argmin}}\sum_{l=1}^{L}\lambda_l g(\boldsymbol{z}_l) - \sum_{l=1}^{L}\left\langle \boldsymbol{\alpha}_l^{(n)}, \boldsymbol{z}_l - \boldsymbol{D}_l\boldsymbol{x}^{(n+1)}\right\rangle + \sum_{l=1}^{L}\frac{\rho_l}{2}||\boldsymbol{z}_l - \boldsymbol{D}_l\boldsymbol{x}^{(n+1)}||_2^2 \\ \boldsymbol{\alpha}^{(n+1)} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\sum_{l=1}^{L}\left\langle \boldsymbol{\alpha}_l, \boldsymbol{D}_l\boldsymbol{x}^{(n+1)} - \boldsymbol{z}_l^{(n+1)}\right\rangle \end{cases}$$

ADMM procedure:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 + \sum_{l=1}^{L} \lambda_l g(\boldsymbol{D}_l \boldsymbol{x}) \right\}$$

$$\begin{cases} \mathbf{X^{(n)}}: \boldsymbol{x}^{(n)} = \boldsymbol{F}^T \left[\boldsymbol{P}^T \boldsymbol{P} + \sum_{l=1}^{L} \rho_l \boldsymbol{F} \boldsymbol{D}_l^T \boldsymbol{D}_l \boldsymbol{F}^T \right]^{-1} \left[\boldsymbol{P}^T \boldsymbol{y} + \sum_{l=1}^{L} \rho_l \boldsymbol{F} \boldsymbol{D}_l^T \left(\boldsymbol{z}_l^{(n-1)} - \boldsymbol{\beta}_l^{(n-1)} \right) \right] \\ \mathbf{Z^{(n)}}: \boldsymbol{z}_l^{(n)} = S\left(\boldsymbol{D}_l x^{(n)} + \boldsymbol{\beta}_l^{(n-1)}; \lambda_l / \rho_l \right) \\ \mathbf{M^{(n)}}: \boldsymbol{\beta}_l^{(n)} = \boldsymbol{\beta}_l^{(n-1)} + \eta_l \left(\boldsymbol{D}_l \boldsymbol{x}^{(n)} - \boldsymbol{z}_l^{(n)} \right) \end{cases}$$

$\boldsymbol{\beta}_l = \boldsymbol{\alpha}_l / \rho_l$: the scaled Lagrangian multiplier

$S(\cdot)$: the nonlinear shrinkage function, e.g., soft or hard thresholding function corresponding to the sparse regularization of l_1-norm and l_0-norm, respectively

η_l: the update rate

ADMM procedure:

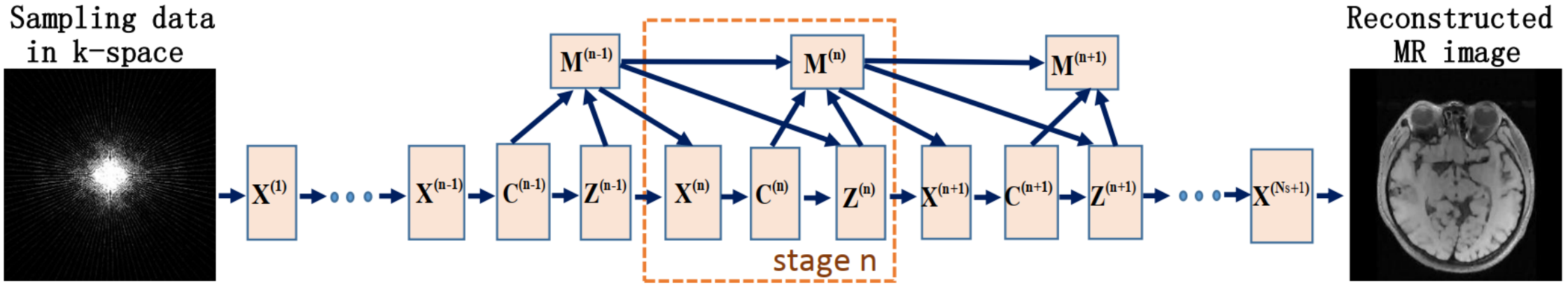
$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 + \sum_{l=1}^{L} \lambda_l g(\boldsymbol{D}_l \boldsymbol{x}) \right\}$$

$$\begin{cases} \mathbf{X^{(n)}}: \boldsymbol{x}^{(n)} = \boldsymbol{F}^T \left[\boldsymbol{P}^T \boldsymbol{P} + \sum_{l=1}^{L} \rho_l \boldsymbol{F} \boldsymbol{D}_l^T \boldsymbol{D}_l \boldsymbol{F}^T \right]^{-1} \left[\boldsymbol{P}^T \boldsymbol{y} + \sum_{l=1}^{L} \rho_l \boldsymbol{F} \boldsymbol{D}_l^T \left(\boldsymbol{z}_l^{(n-1)} - \boldsymbol{\beta}_l^{(n-1)} \right) \right] \\ \mathbf{Z^{(n)}}: \boldsymbol{z}_l^{(n)} = S\left(\boldsymbol{D}_l x^{(n)} + \boldsymbol{\beta}_l^{(n-1)}; \lambda_l / \rho_l \right) \\ \mathbf{M^{(n)}}: \boldsymbol{\beta}_l^{(n)} = \boldsymbol{\beta}_l^{(n-1)} + \eta_l \left(\boldsymbol{D}_l \boldsymbol{x}^{(n)} - \boldsymbol{z}_l^{(n)} \right) \end{cases}$$

- It is challenging to choose the transform $\boldsymbol{D}_l$ and shrinkage function $S(\cdot)$ for general regularization function $g(\cdot)$
- It is also not trivial to tune the parameters ρ_l and η_l for k-space data with different sampling ratios

Deep ADMM-Net:

$$\begin{cases} \mathbf{X^{(n)}}: \boldsymbol{x}^{(n)} = \boldsymbol{F}^T \left[\boldsymbol{P}^T \boldsymbol{P} + \sum_{l=1}^{L} \rho_l \boldsymbol{F} \boldsymbol{D}_l^T \boldsymbol{D}_l \boldsymbol{F}^T \right]^{-1} \left[\boldsymbol{P}^T \boldsymbol{y} + \sum_{l=1}^{L} \rho_l \boldsymbol{F} \boldsymbol{D}_l^T \left(\boldsymbol{z}_l^{(n-1)} - \boldsymbol{\beta}_l^{(n-1)} \right) \right] \\ \mathbf{Z^{(n)}}: \boldsymbol{z}_l^{(n)} = S\left(\boldsymbol{D}_l \boldsymbol{x}^{(n)} + \boldsymbol{\beta}_l^{(n-1)}; \lambda_l / \rho_l \right) \\ \mathbf{M^{(n)}}: \boldsymbol{\beta}_l^{(n)} = \boldsymbol{\beta}_l^{(n-1)} + \eta_l \left(\boldsymbol{D}_l \boldsymbol{x}^{(n)} - \boldsymbol{z}_l^{(n)} \right) \end{cases}$$



- Reconstruction operation $\boldsymbol{X}^{(n)}$
- Convolution operation $\boldsymbol{C}^{(n)}$ defined by $\left\{ \boldsymbol{D}_l \boldsymbol{x}^{(n)} \right\}_{l=1}^{L}$
- Nonlinear transform operation $\boldsymbol{Z}^{(n)}$ defined by $S(\cdot)$
- Multiplier update operation $\boldsymbol{M}^{(n)}$

Deep ADMM-Net — Reconstruction layer $\boldsymbol{X}^{(n)}$:

$$\boldsymbol{x}^{(n)} = \boldsymbol{F}^T \left[\boldsymbol{P}^T \boldsymbol{P} + \sum_{l=1}^{L} \rho_l \boldsymbol{F} \boldsymbol{H}_l^T \boldsymbol{H}_l \boldsymbol{F}^T \right]^{-1} \left[\boldsymbol{P}^T \boldsymbol{y} + \sum_{l=1}^{L} \rho_l \boldsymbol{F} \boldsymbol{H}_l^T \left(\boldsymbol{z}_l^{(n-1)} - \boldsymbol{\beta}_l^{(n-1)} \right) \right]$$

$\boldsymbol{H}_l$: the l-th filter

First stage update:

$$\boldsymbol{x}^{(1)} = \boldsymbol{F}^T \left(\boldsymbol{P}^T \boldsymbol{P} + \sum_{l=1}^{L} \rho_l^{(1)} \boldsymbol{F} \boldsymbol{H}_l^{(1)T} \boldsymbol{H}_l^{(1)} \boldsymbol{F}^T \right)^{-1} (\boldsymbol{P}^T \boldsymbol{y})$$

ADMM update formula: $\boldsymbol{x}^{(n)} = \boldsymbol{F}^T \left[\boldsymbol{P}^T \boldsymbol{P} + \sum_{l=1}^{L} \rho_l \boldsymbol{F} \boldsymbol{D}_l^T \boldsymbol{D}_l \boldsymbol{F}^T \right]^{-1} \left[\boldsymbol{P}^T \boldsymbol{y} + \sum_{l=1}^{L} \rho_l \boldsymbol{F} \boldsymbol{D}_l^T \left(\boldsymbol{z}_l^{(n-1)} - \boldsymbol{\beta}_l^{(n-1)} \right) \right]$

Deep ADMM-Net — Convolution layer $\boldsymbol{C}^{(n)}$:

Transform an image into transform domain

$$\boldsymbol{c}_l^{(n)} = \boldsymbol{D}_l^{(n)} \boldsymbol{x}^{(n)}$$

$\boldsymbol{D}_l$: a learnable filter matrix in stage n

- Note that the filters $\boldsymbol{H}_l$ and $\boldsymbol{D}_l$ do not be constrained to be the same to increase the network capacity

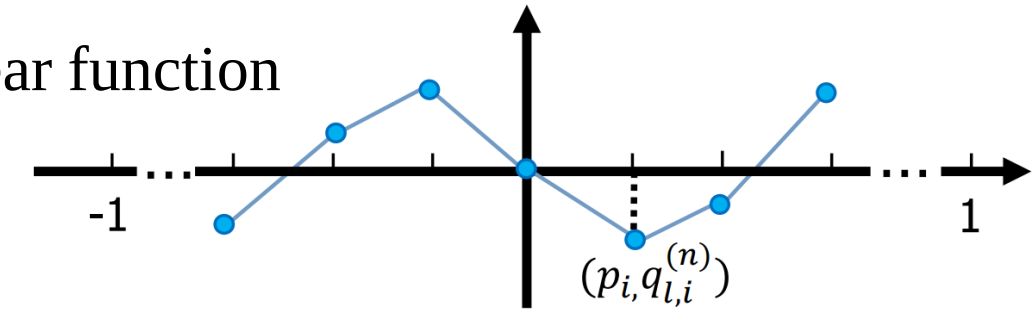
---

ADMM update formula: $\boldsymbol{z}_l^{(n)} = S\left(\boldsymbol{D}_l x^{(n)} + \boldsymbol{\beta}_l^{(n-1)}; \lambda_l/\rho_l\right)$

Deep ADMM-Net — Nonlinear transform layer $\boldsymbol{Z}^{(n)}$:

Learn more general function using piecewise linear function

$$\boldsymbol{z}_l^{(n)} = S_{PLF}\left(\boldsymbol{c}_l^{(n)} + \boldsymbol{\beta}_l^{(n-1)}; \left\{p_i, q_{l,i}^{(n)}\right\}_{i=1}^{N_c}\right)$$



$$S_{PLF}\left(a; \left\{p_i, q_{l,i}^{(n)}\right\}_{i=1}^{N_c}\right) = \begin{cases} a + q_{l,1}^{(n)} - p_1, & a < p_1, \\ a + q_{l,N_c}^{(n)} - p_{N_c}, & a > p_{N_c}, \\ q_{l,k}^{(n)} + \dfrac{(a - p_k)\left(q_{l,k+1}^{(n)} - q_{l,k}^{(n)}\right)}{p_{k+1} - p_k}, & p_1 \le a \le p_{N_c}, \end{cases}$$

$$k = \left\lfloor \frac{a - p_1}{p_2 - p_1} \right\rfloor$$

$\{p_i\}_{i=1}^{N_c}$: the predefined positions uniformly located within $[-1,1]$

$\left\{q_{l,i}^{(n)}\right\}_{i=1}^{N_c}$: the values at these positions for l-th filter in n-th stage

---

ADMM update formula: $\boldsymbol{z}_l^{(n)} = S\left(\boldsymbol{D}_l x^{(n)} + \boldsymbol{\beta}_l^{(n-1)}; \lambda_l/\rho_l\right)$

Deep ADMM-Net — Multiplier update layer $\boldsymbol{M}^{(n)}$:

Lagrangian multiplier updating procedure

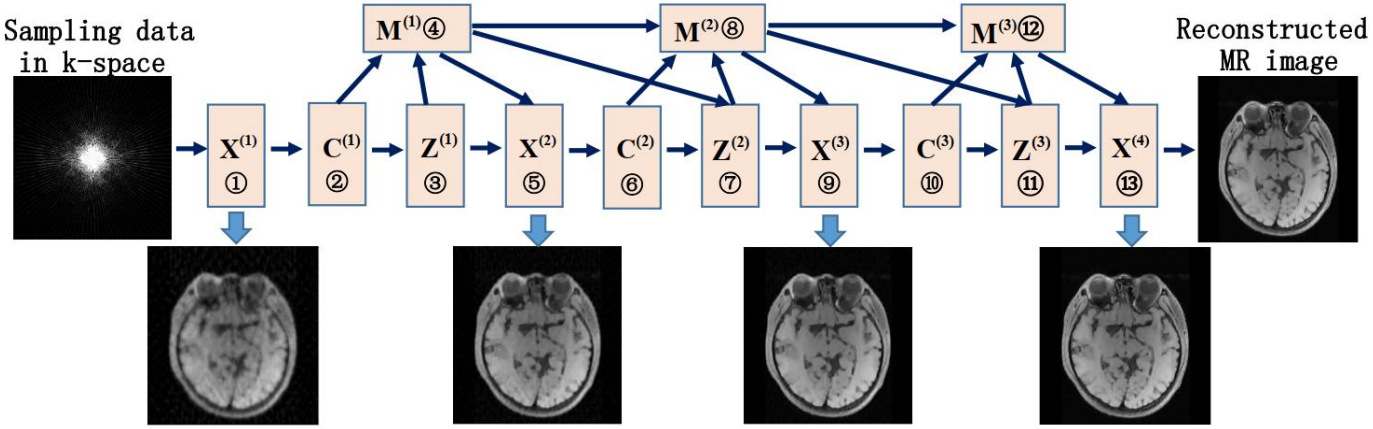
$$\boldsymbol{\beta}_l^{(n)} = \boldsymbol{\beta}_l^{(n-1)} + \eta_l{}^{(n)} \left(\boldsymbol{c}_l{}^{(n)} - \boldsymbol{z}_l^{(n)} \right)$$

$\eta_l{}^{(n)}$: a learnable parameter

ADMM update formula: $\boldsymbol{\beta}_l^{(n)} = \boldsymbol{\beta}_l^{(n-1)} + \eta_l \left(\boldsymbol{D}_l \boldsymbol{x}^{(n)} - \boldsymbol{z}_l^{(n)} \right)$

Network parameters and training:

$$E(\Theta) = \frac{1}{|\Gamma|} \sum_{(y,x^{gt})\in\Gamma} \frac{\sqrt{||\hat{x}(y,\Theta) - x^{gt}||_2^2}}{\sqrt{||x^{gt}||_2^2}}$$



- Reconstruction layer $\boldsymbol{X}^{(n)}$: $\boldsymbol{x}^{(n)} = \boldsymbol{F}^T\left[\boldsymbol{P}^T\boldsymbol{P} + \sum_{l=1}^{L} \rho_l \boldsymbol{F}\boldsymbol{H}_l^T\boldsymbol{H}_l\boldsymbol{F}^T\right]^{-1}\left[\boldsymbol{P}^T\boldsymbol{y} + \sum_{l=1}^{L} \rho_l \boldsymbol{F}\boldsymbol{H}_l^T\left(\boldsymbol{z}_l^{(n-1)} - \boldsymbol{\beta}_l^{(n-1)}\right)\right]$
- Convolution layer $\boldsymbol{C}^{(n)}$: $\boldsymbol{c}_l^{(n)} = \boldsymbol{D}_l^{(n)}\boldsymbol{x}^{(n)}$
- Nonlinear transform layer $\boldsymbol{Z}^{(n)}$: $\boldsymbol{z}_l^{(n)} = S_{PLF}\left(\boldsymbol{c}_l^{(n)} + \boldsymbol{\beta}_l^{(n-1)}; \left\{p_i, q_{l,i}^{(n)}\right\}_{i=1}^{N_c}\right)$
- Multiplier update layer $\boldsymbol{M}^{(n)}$: $\boldsymbol{\beta}_l^{(n)} = \boldsymbol{\beta}_l^{(n-1)} + \eta_l{}^{(n)}\left(\boldsymbol{c}_l{}^{(n)} - \boldsymbol{z}_l^{(n)}\right)$

Table 1: Performance comparisons on brain data with different sampling ratios.

Method	20%		30%		40%		50%		Test time
	NMSE	PSNR	NMSE	PSNR	NMSE	PSNR	NMSE	PSNR	
Zero-filling	0.1700	29.96	0.1247	32.59	0.0968	34.76	0.0770	36.73	0.0013s
TV [2]	0.0929	35.20	0.0673	37.99	0.0534	40.00	0.0440	41.69	0.7391s
RecPF [4]	0.0917	35.32	0.0668	38.06	0.0533	40.03	0.0440	41.71	0.3105s
SIDWT	0.0885	35.66	0.0620	38.72	0.0484	40.88	0.0393	42.67	7.8637s
PBDW [6]	0.0814	36.34	0.0627	38.64	0.0518	40.31	0.0437	41.81	35.3637s
PANO [10]	0.0800	36.52	0.0592	39.13	0.0477	41.01	0.0390	42.76	53.4776s
FDLCP [8]	0.0759	36.95	0.0592	39.13	0.0500	40.62	0.0428	42.00	52.2220s
BM3D-MRI [11]	0.0674	37.98	0.0515	40.33	0.0426	41.99	0.0359	43.47	40.9114s
Init-Net$_{13}$	0.1394	31.58	0.1225	32.71	0.1128	33.44	0.1066	33.95	0.6914s
ADMM-Net$_{13}$	0.0752	37.01	0.0553	39.70	0.0456	41.37	0.0395	42.62	0.6964s
ADMM-Net$_{14}$	0.0742	37.13	0.0548	39.78	0.0448	41.54	0.0380	42.99	0.7400s
ADMM-Net$_{15}$	0.0739	37.17	0.0544	39.84	0.0447	41.56	0.0379	43.00	0.7911s

Table 2: Comparisons of NMSE and PSNR on chest data with 20% sampling ratio.

Method	TV	RecPF	PANO	FDLCP	ADMM-Net$_{15}$-B	ADMM-Net$_{15}$	ADMM-Net$_{17}$
NMSE	0.1019	0.1017	0.0858	0.0775	0.0790	0.0775	0.0768
PSNR	35.49	35.51	37.01	37.77	37.68	37.84	37.92

Expansion:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 + \sum_{l=1}^{L} \lambda_l g(\boldsymbol{D}_l \boldsymbol{x}) \right\}$$

ADMM-Net: introduce auxiliary variables $\boldsymbol{z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_L\}$

$$\min_{\boldsymbol{x},\boldsymbol{z}} \frac{1}{2} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 + \sum_{l=1}^{L} \lambda_l g(\boldsymbol{z}_l)$$

$$\text{s.t.} \quad \boldsymbol{z}_l = \boldsymbol{D}_l \boldsymbol{x}, \qquad \forall l \in [1,2,\cdots,L]$$

ADMM-CSNet[1]: introduce auxiliary variable $\boldsymbol{z} = \boldsymbol{x}$

$$\min_{\boldsymbol{x},\boldsymbol{z}} \frac{1}{2} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}||_2^2 + \sum_{l=1}^{L} \lambda_l g(\boldsymbol{D}_l \boldsymbol{z})$$

$$\text{s.t.} \quad \boldsymbol{z} = \boldsymbol{x}$$

[1] Yang Y, Sun J, Li H, et al. ADMM-CSNet: A deep learning approach for image compressive sensing [J]. IEEE transactions on pattern analysis and machine intelligence, 2018, 42(3): 521-538.

ISTA-Net: Interpretable Optimization-Inspired Deep Network for Image Compressive Sensing

Jian Zhang, Bernard Ghanem
King Abdullah University of Science and Technology (KAUST), Saudi Arabia
jian.zhang@kaust.edu.sa, bernard.ghanem@kaust.edu.sa

CVPR 2018

Contributions:

- A novel ISTA-Net, which adopts the structure of ISTA update steps to design a learnable deep network manifestation
- By exploiting CS realm in the residual domain, an enhanced version ISTA-Net^{+} is derived to further improve CS performance
- The proximal mapping problem associated to a nonlinear sparsifying transform is solved in a general and efficient way

CS reconstruction problem:

$$\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{\Phi}\mathbf{x}-\mathbf{y}||_2^2+\lambda||\boldsymbol{\Psi}\mathbf{x}||_1$$

ISTA algorithm[1] procedure:

$$\mathrm{prox}_{\lambda\phi}(\boldsymbol{r})=\underset{\boldsymbol{x}}{\mathrm{argmin}}\,\frac{1}{2}||\boldsymbol{x}-\boldsymbol{r}||_2^2+\lambda\phi(\boldsymbol{x})$$

$$\boldsymbol{r}^{(k)}=\boldsymbol{x}^{(k-1)}-\rho\boldsymbol{\Phi}^{\top}\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k-1)}-\boldsymbol{y}\right)$$

$$\boldsymbol{x}^{(k)}=\arg\min_{\boldsymbol{x}}\frac{1}{2}||\boldsymbol{x}-\boldsymbol{r}^{(k)}||_2^2+\lambda||\boldsymbol{\Psi}\mathbf{x}||_1$$

[1] I. Daubechies, M. Defrise, and C. D. Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, Comm. Pure Appl. Math., 57 (2004), pp. 1413–1457.

$$\boldsymbol{x}^{(k)} = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{x} - \boldsymbol{r}^{(k)}||_2^2 + \lambda||\boldsymbol{\Psi}\boldsymbol{x}||_1$$

- Inspired by the powerful representation power of CNN and its universal approximation property

$$\mathcal{F}(\boldsymbol{x}) = \boldsymbol{\Psi}\boldsymbol{x}$$

$\mathcal{F}(\cdot)$ as a combination of two linear convolutional operators separated by a ReLU function

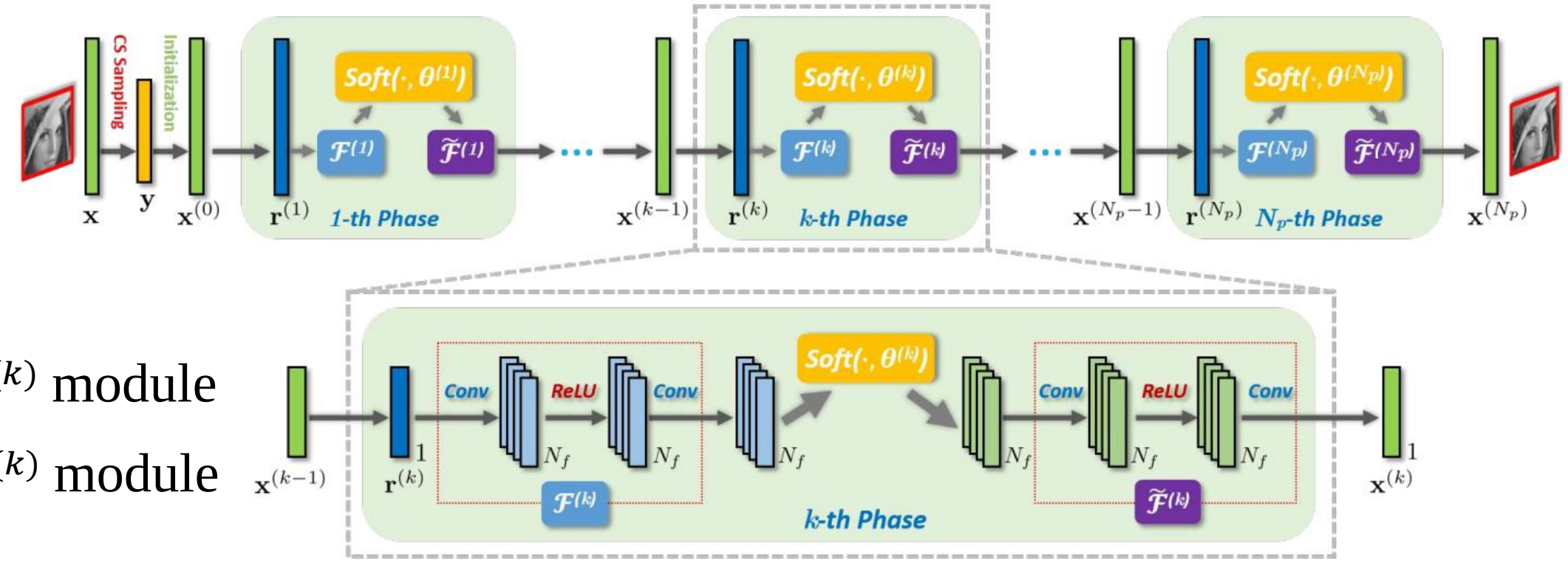
- Then

$$\boldsymbol{x}^{(k)} = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{x} - \boldsymbol{r}^{(k)}||_2^2 + \lambda||\mathcal{F}(\boldsymbol{x})||_1$$

ISTA-Net framework:

$$\boldsymbol{r}^{(k)} = \boldsymbol{x}^{(k-1)} - \rho \boldsymbol{\Phi}^{\top}\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k-1)} - \boldsymbol{y}\right)$$

$$\boldsymbol{x}^{(k)} = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{x} - \boldsymbol{r}^{(k)}||_2^2 + \lambda||\mathcal{F}(\boldsymbol{x})||_1$$



- $\boldsymbol{r}^{(k)}$ module
- $\boldsymbol{x}^{(k)}$ module

ISTA-Net — $\boldsymbol{r}^{(k)}$ Module:

$$\boldsymbol{r}^{(k)} = \boldsymbol{x}^{(k-1)} - \rho \boldsymbol{\Phi}^{\top}\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k-1)} - \boldsymbol{y}\right)$$

To preserve the ISTA structure while increasing network flexibility, the step size ρ is allowed to vary across iterations

$$\boldsymbol{r}^{(k)} = \boldsymbol{x}^{(k-1)} - \rho^{(k)} \boldsymbol{\Phi}^{\top}\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k-1)} - \boldsymbol{y}\right)$$

ISTA-Net — $\boldsymbol{x}^{(k)}$ Module: $$\boldsymbol{x}^{(k)} = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{x} - \boldsymbol{r}^{(k)}||_2^2 + \lambda||\mathcal{F}(\boldsymbol{x})||_1$$

- Note that $\boldsymbol{r}^{(k)}$ is the immediate reconstruction result of $\boldsymbol{x}^{(k)}$
- Thus, $\boldsymbol{x}^{(k)} - \boldsymbol{r}^{(k)}$ follows an independent normal distribution with common zero mean and variance σ^2
- Approximation (Theorem 1):

$$||\mathcal{F}(\boldsymbol{x}) - \mathcal{F}(\boldsymbol{r}^{(k)})||_2^2 \approx \alpha||\boldsymbol{x} - \boldsymbol{r}^{(k)}||_2^2$$

- Equivalent transformation

$$\boldsymbol{x}^{(k)} = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{F}(\boldsymbol{x}) - \mathcal{F}(\boldsymbol{r}^{(k)})||_2^2 + \theta||\mathcal{F}(\boldsymbol{x})||_1$$

Theorem 1 *Let $X_1, ..., X_n$ be independent normal random variables with common zero mean and variance σ^2. If $\vec{X} = [X_1, ..., X_n]^\top$ and given any matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{s \times m}$, define a new random variable $\vec{Y} = \mathbf{B}ReLU(\mathbf{A}\vec{X}) = \mathbf{B}\max(\mathbf{0}, \mathbf{A}\vec{X})$. Then, $\mathbb{E}[\|\vec{Y} - \mathbb{E}[\vec{Y}]\|_2^2]$ and $\mathbb{E}[\|\vec{X} - \mathbb{E}[\vec{X}]\|_2^2]$ are linearly related,* i.e. *$\mathbb{E}[\|\vec{Y} - \mathbb{E}[\vec{Y}]\|_2^2] = \alpha\mathbb{E}[\|\vec{X} - \mathbb{E}[\vec{X}]\|_2^2]$, where α is only a function of $\mathbf{A}$ and $\mathbf{B}$. (Please refer to the **supplementary material** for the proof and more details.)*

ISTA-Net — $\boldsymbol{x}^{(k)}$ Module: $$\boldsymbol{x}^{(k)} = \arg\min_{\boldsymbol{x}} \frac{1}{2} ||\mathcal{F}(\boldsymbol{x}) - \mathcal{F}\left(\boldsymbol{r}^{(k)}\right)||_2^2 + \theta ||\mathcal{F}(\boldsymbol{x})||_1$$

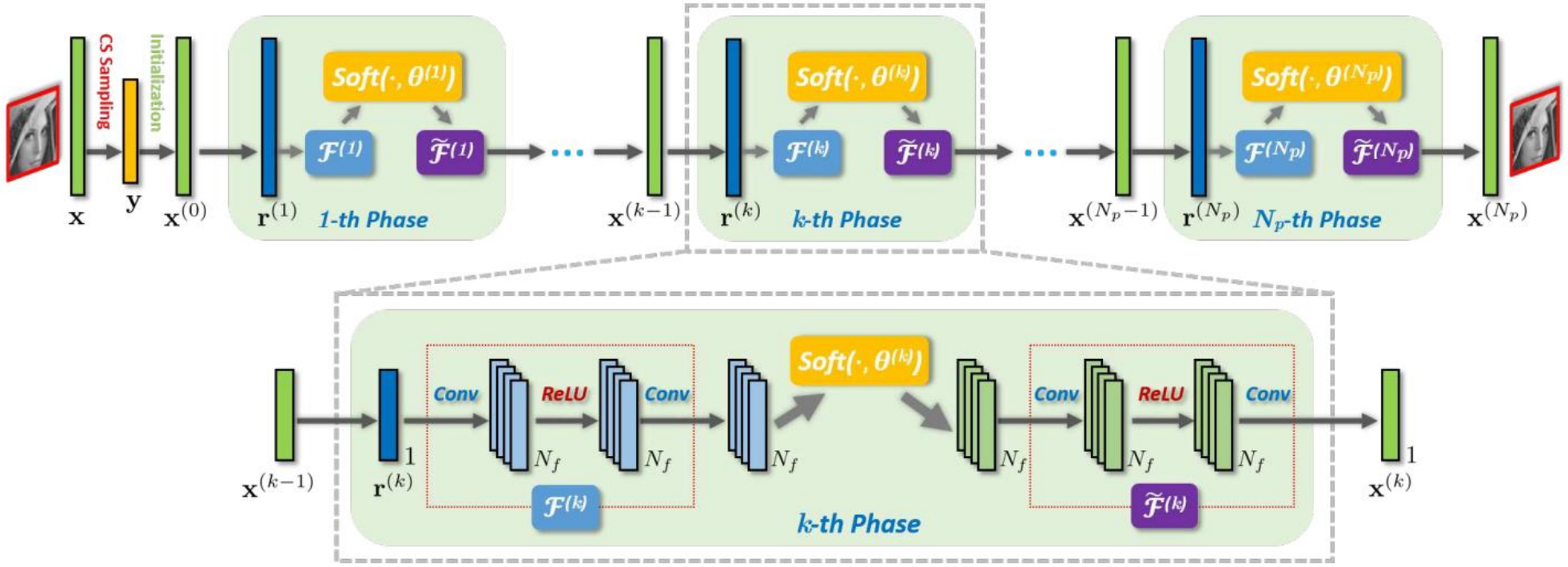
- Then, we get a closed-form version of $\mathcal{F}\left(\boldsymbol{x}^{(k)}\right)$

$$\mathcal{F}\left(\boldsymbol{x}^{(k)}\right) = \text{soft}\left(\mathcal{F}\left(\boldsymbol{r}^{(k)}\right), \theta\right)$$

- Motivated by the invertible characteristics of the wavelet transform, the left inverse of $\mathcal{F}(\cdot)$ may be introduced, such that $\tilde{\mathcal{F}} \circ \mathcal{F} = I$, yielding

$$\boldsymbol{x}^{(k)} = \tilde{\mathcal{F}}\left(\text{soft}\left(\mathcal{F}\left(\boldsymbol{r}^{(k)}\right), \theta\right)\right)$$

Parameters in ISTA-Net:



- $\boldsymbol{r}^{(k)}$ module: $\boldsymbol{r}^{(k)} = \boldsymbol{x}^{(k-1)} - \rho^{(k)}\boldsymbol{\Phi}^{\top}\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k-1)} - \boldsymbol{y}\right)$
- $\boldsymbol{x}^{(k)}$ module: $\boldsymbol{x}^{(k)} = \tilde{\mathcal{F}}\left(\mathrm{soft}\left(\mathcal{F}\left(\boldsymbol{r}^{(k)}\right), \theta\right)\right)$

Loss function design:

- The training data pairs $\{(\boldsymbol{y}_i, \boldsymbol{x}_i)\}_{i=1}^{N_b}$, and $\gamma = 0.01$

$$\mathcal{L}_{total}(\boldsymbol{\Theta}) = \mathcal{L}_{discrepancy} + \gamma \mathcal{L}_{constraint}$$

with:

$$\begin{cases} \mathcal{L}_{discrepancy} = \frac{1}{N_b N} \sum_{i=1}^{N_b} ||\boldsymbol{x}_i^{(N_p)} - \boldsymbol{x}_i||_2^2 & \text{Reconstruction error} \\ \mathcal{L}_{constraint} = \frac{1}{N_b N} \sum_{i=1}^{N_b} \sum_{k=1}^{N_p} ||\tilde{\mathcal{F}}^{(k)}\left(\mathcal{F}^{(k)}(\boldsymbol{x}_i)\right) - \boldsymbol{x}_i||_2^2 & \text{Invertible error} \end{cases}$$

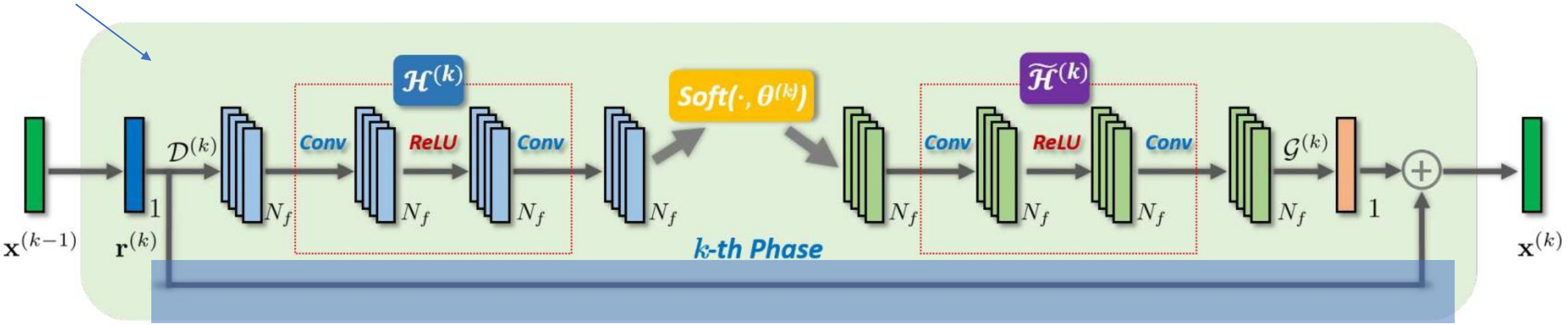
Enhanced version ISTA-Net^{+}: $\boldsymbol{x}^{(k)} = \arg\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{F}(\boldsymbol{x}) - \mathcal{F}\left(\boldsymbol{r}^{(k)}\right)||_2^2 + \theta||\mathcal{F}(\boldsymbol{x})||_1$

- Based on the fact that the residuals of natural images and videos are more compressible
- Assume that $\boldsymbol{x}^{(k)} = \boldsymbol{r}^{(k)} + \boldsymbol{w}^{(k)} + \boldsymbol{e}^{(k)}$, where $\boldsymbol{e}^{(k)}$ denotes some noise
- $\boldsymbol{w}^{(k)}$ denotes some missing high-frequency component, which may be extracted by a linear operator $\boldsymbol{w}^{(k)} = \mathcal{R}\left(\boldsymbol{x}^{(k)}\right) = \mathcal{G} \circ \mathcal{D}\left(\boldsymbol{x}^{(k)}\right)$
- By modeling $\mathcal{F}(\cdot) = \mathcal{H} \circ \mathcal{D}$

$$\min_{\boldsymbol{x}} \frac{1}{2}||\mathcal{H}(\mathcal{D}(\boldsymbol{x})) - \mathcal{H}\left(\mathcal{D}\left(\boldsymbol{r}^{(k)}\right)\right)||_2^2 + \theta||\mathcal{H}\left(\mathcal{D}(\boldsymbol{x})\right)||_1 \Longrightarrow \boldsymbol{x}^{(k)} \Longrightarrow \boldsymbol{x}^{(k)} = \boldsymbol{r}^{(k)} + \mathcal{R}\left(\boldsymbol{x}^{(k)}\right)$$

$$\boldsymbol{x}^{(k)} = \boldsymbol{r}^{(k)} + \mathcal{G}\left(\widetilde{\mathcal{H}}\left(\text{soft}\left(\mathcal{H}\left(\mathcal{D}\left(\boldsymbol{r}^{(k)}\right)\right), \theta\right)\right)\right)$$

Parameters in ISTA-Net+:



skip connections (coincide with Res-Net)

- $\boldsymbol{r}^{(k)}$ module: $\boldsymbol{r}^{(k)} = \boldsymbol{x}^{(k-1)} - \rho^{(k)}\boldsymbol{\Phi}^{\top}\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k-1)} - \boldsymbol{y}\right)$
- $\boldsymbol{x}^{(k)}$ module: $\boldsymbol{x}^{(k)} = \boldsymbol{r}^{(k)} + \mathcal{G}\left(\widetilde{\mathcal{H}}\left(\mathrm{soft}\left(\mathcal{H}\left(\mathcal{D}\left(\boldsymbol{r}^{(k)}\right)\right), \theta\right)\right)\right)$

ISTA-Net v.s. ISTA-Net$^+$:

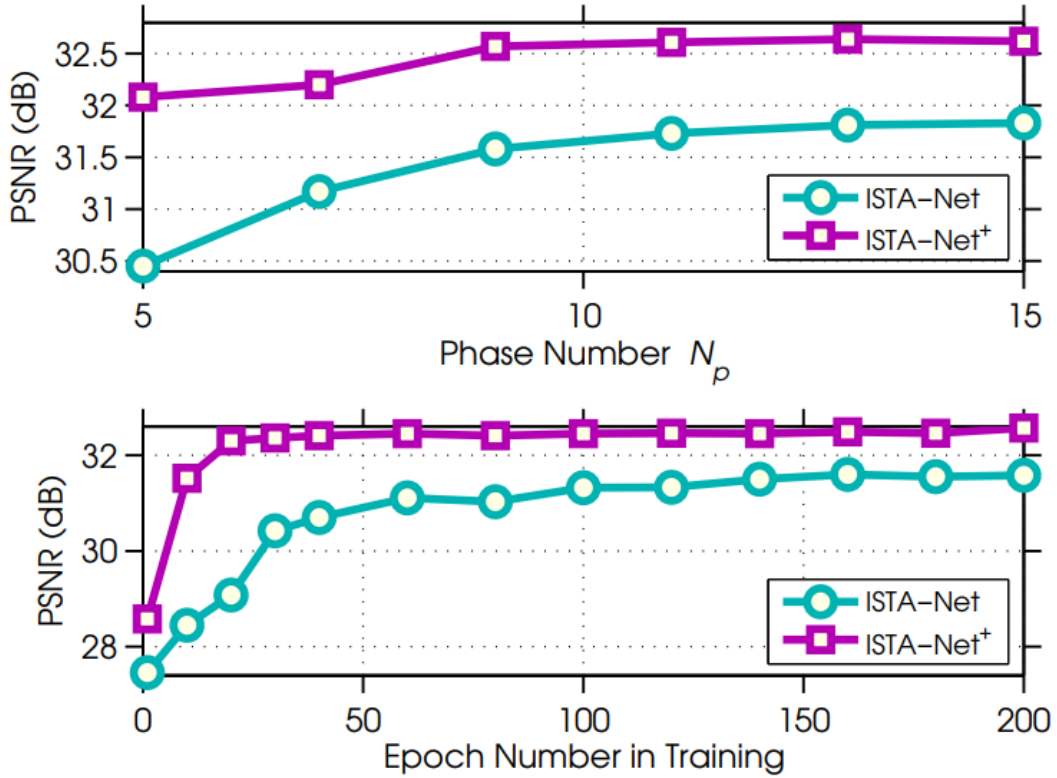


Figure 4. PSNR comparison between ISTA-Net and ISTA-Net$^+$ with various numbers of phases and epochs during training.

Table 1. Average PSNR (dB) performance comparisons on Set11 with different CS ratios. The best performance is labeled in bold and the second best is underlined. Note that the last two columns is a run-time analysis of all the competing methods, showing the average time to reconstruct a 256×256 image and the corresponding frames-per-second (FPS).

Algorithm	**CS Ratio**							**Time**	**FPS**
	50%	40%	30%	25%	10%	4%	1%	CPU/GPU	CPU/GPU
TVAL3 [22]	33.55	31.46	29.23	27.92	22.99	18.75	16.43	3.135s/———	0.32/——
D-AMP [28]	35.92	33.56	30.39	28.46	22.64	18.40	5.21	51.21s/———	0.02/——
IRCNN [49]	36.23	34.06	31.18	30.07	24.02	17.56	7.70	———/68.42s	—-/0.015
SDA [30]	28.95	27.79	26.63	25.34	22.65	20.12	17.29	——/0.0032s	—-/312.5
ReconNet [21]	31.50	30.58	28.74	25.60	24.28	20.63	17.27	———/0.016s	——/62.5
ISTA-Net	<u>37.43</u>	<u>35.36</u>	<u>32.91</u>	<u>31.53</u>	<u>25.80</u>	<u>21.23</u>	<u>17.30</u>	0.923s/0.039s	1.08/25.6
ISTA-Net$^+$	**38.07**	**36.06**	**33.82**	**32.57**	**26.64**	**21.31**	**17.34**	1.375s/0.047s	0.73/21.3

Table 2. Average PSNR (dB) performance comparison of various network-based algorithms on the BSD68 dataset.

Algorithm	**CS Ratio**				
	50%	40%	30%	10%	4%
SDA [30]	28.35	27.41	26.38	23.12	21.32
ReconNet [21]	29.86	29.08	27.53	24.15	21.66
ISTA-Net	<u>33.60</u>	<u>31.85</u>	<u>29.93</u>	<u>25.02</u>	<u>22.12</u>
ISTA-Net$^+$	**34.01**	**32.21**	**30.34**	**25.33**	**22.17**

Table 3. Average PSNR (dB) comparison between ADMM-Net [44] and our proposed ISTA-Nets for CS-MRI.

Algorithm	**CS Ratio**				**Time**
	20%	30%	40%	50%	GPU
ADMM-Net	37.17	39.84	41.56	43.00	0.9535s
ISTA-Net	<u>38.30</u>	<u>40.52</u>	<u>42.12</u>	<u>43.60</u>	0.1246s
ISTA-Net$^+$	**38.73**	**40.89**	**42.52**	**44.09**	0.1437s

FISTA-Net: Learning a Fast Iterative Shrinkage Thresholding Network for Inverse Problems in Imaging

Jinxi Xiang, *Graduate Student Member, IEEE*, Yonggui Dong, *Member, IEEE*, and Yunjie Yang, *Member, IEEE*

TMI 2021

Contributions:

- In FISTA-Net, the gradient matrix is learned by substituting the fixed classic gradient throughout iterations
- The core parameters of FISTA-Net, e.g., the step size, thresholding values, are regularized to converge properly
- A momentum module is added in FISTA-Net to accelerate convergence

Sparse representation: $\hat{\boldsymbol{x}}_{\mathcal{F}_{l1}} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\{||\mathbf{A}\boldsymbol{x} - \mathbf{b}||_2^2 + \lambda||\mathcal{F}(\boldsymbol{x})||_1\}$

FISTA-Net:

- Gradient descent module

$$\boldsymbol{r}^{(k)} = \boldsymbol{y}^{(k)} - \left(\mathbf{W}^{(k)}\right)^T\left(\mathbf{A}\boldsymbol{y}^{(k)} - \mathbf{b}\right)$$

- Proximal mapping module

$$\boldsymbol{x}^{(k)} = \mathcal{T}_{\theta^{(k)}}\left(\boldsymbol{r}^{(k)}\right)$$

- Momentum module

$$\boldsymbol{y}^{(k+1)} = \mathbf{x}^{(k)} + \rho^{(k)}\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right)$$

FISTA algorithm[1]:

$$\boldsymbol{x}^{(k)} = \mathcal{T}_{\alpha}\left(\boldsymbol{y}^{(k)} - \mu\mathbf{A}^T\left(\mathbf{A}\boldsymbol{y}^{(k)} - \mathbf{b}\right)\right)$$

$$t^{(k+1)} = \frac{1 + \sqrt{1 + 4(t^{(k)})^2}}{2}$$

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{x}^{(k)} + \left(\frac{t^{(k)} - 1}{t^{(k+1)}}\right)\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right)$$

[1] A fast iterative shrinkage-thresholding algorithm for linear inverse problems [J]. SIAM journal on imaging sciences, 2009, 2(1): 183-202.

FISTA-Net — Gradient descent module $\boldsymbol{r}^{(k)}$:

$$\boldsymbol{r}^{(k)} = \boldsymbol{y}^{(k)} - \left(\mathbf{W}^{(k)}\right)^T \left(\mathbf{A}\boldsymbol{y}^{(k)} - \mathbf{b}\right)$$

$\mathbf{W}^{(k)}$ can be decomposed as the product of a scalar $\mu^{(k)}$ and a matrix $\tilde{\mathbf{W}}$ independent of layer index k[1]: $\mathbf{W}^{(k)} = \mu^{(k)}\tilde{\mathbf{W}}$

The matrix $\tilde{\mathbf{W}}$ is pre-computed by solving

$$\tilde{\mathbf{W}} \in \underset{\mathbf{W}\in\mathbb{R}^{N\times M}}{\arg\min} ||\mathbf{W}^T\mathbf{A}||_F^2$$
$$\text{s.t.} \left(\mathbf{W}_{:,m}\right)^T \mathbf{A}_{:,m} = 1, \forall m = 1,2,\cdots,M$$

[1] Liu J, Chen X. ALISTA: Analytic weights are as good as learned weights in LISTA [C]//International Conference on Learning Representations (ICLR). 2019.

FISTA-Net — Proximal mapping module $\boldsymbol{x}^{(k)}$:

$$\boldsymbol{x}^{(k)} = \mathcal{T}_{\theta^{(k)}}\left(\boldsymbol{r}^{(k)}\right)$$

- Similar to ISTA-Net
- Linear convolutional + ReLU
- Invertible operation

FISTA-Net — Momentum module $\boldsymbol{y}^{(k+1)}$:

- Accelerated by introducing a momentum term

$$\boldsymbol{y}^{(k+1)} = \mathbf{x}^{(k)} + \rho^{(k)}\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right)$$

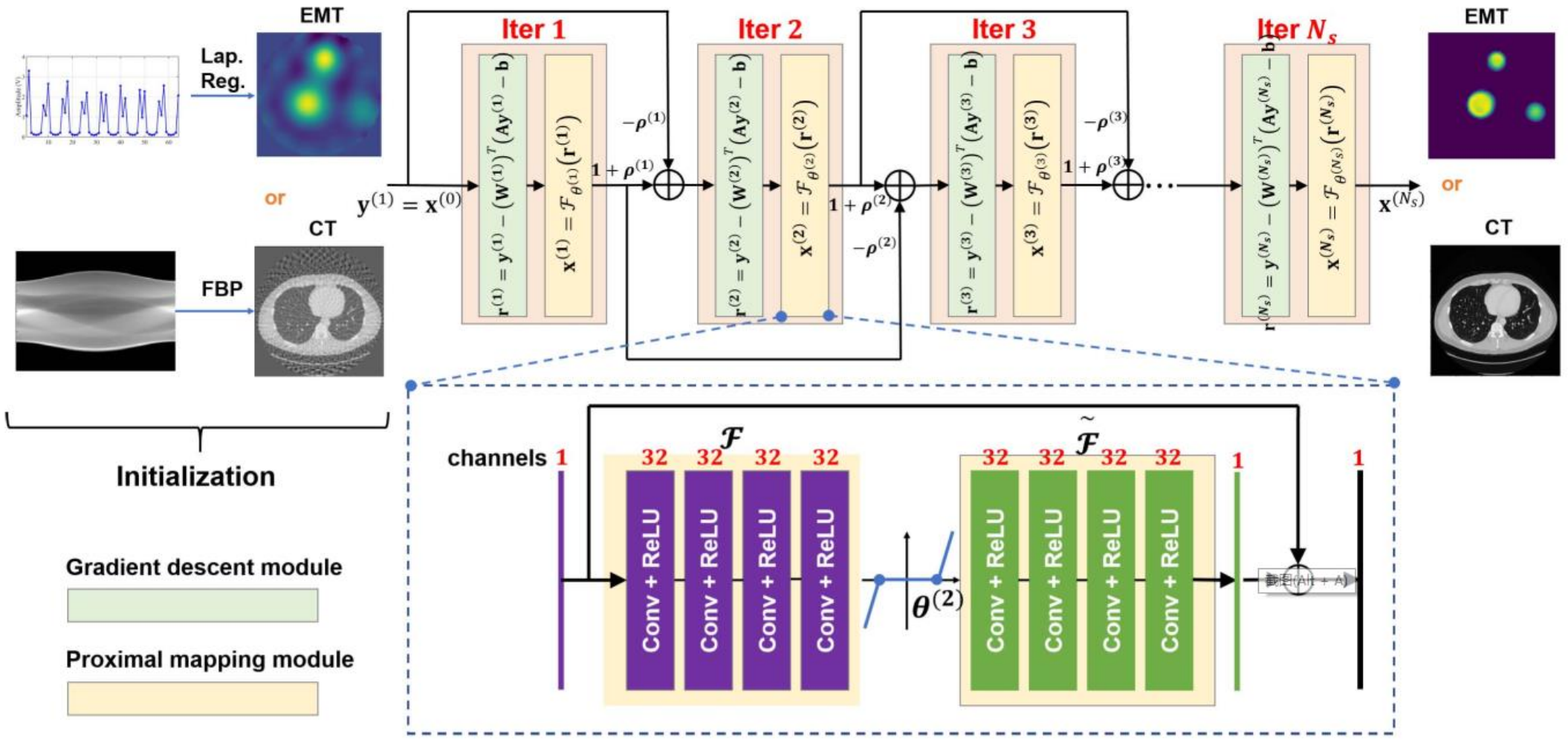
FISTA-Net — Parameter constraints:

- ISTA-Net could possibly generate non-positive step size and thresholding values during iteration, which contradicts the definition of these variables

$$\mu^{(k)} = sp(w_1 k + c_1), w_1 < 0$$
$$\theta^{(k)} = sp(w_2 k + c_2), w_2 < 0$$
$$\rho^{(k)} = \frac{sp(w_3 k + c_3) - sp(w_3 + c_3)}{sp(w_3 k + c_3)}, w_3 > 0$$

where the softplus function $sp(x) = \ln(1 + \exp(x))$ and $\rho^{(k)} \in (0,1)$

FISTA-Net framework:



FISTA-Net Loss Function:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{mse}} + \lambda_1 \mathcal{L}_{\text{sym}} + \lambda_2 \mathcal{L}_{\text{spa}}$$

with

$$\begin{cases} \mathcal{L}_{\text{mse}} = ||\boldsymbol{x}_{N_s} - \boldsymbol{x}_{gt}||_2^2 \\ \mathcal{L}_{\text{tsf}} = \lambda_1 \mathcal{L}_{\text{sym}} + \lambda_2 \mathcal{L}_{\text{spa}} \end{cases}$$

$$= \lambda_1 \sum_{k=1}^{N_s} ||\tilde{\mathcal{F}}\left(\mathcal{F}\left(\boldsymbol{r}^{(k)}\right)\right) - \boldsymbol{r}^{(k)}||_2^2 + \lambda_2 \sum_{k=1}^{N_s} ||\mathcal{F}\left(\boldsymbol{r}^{(k)}\right)||_1$$

TABLE II

QUANTITATIVE METRICS OF DIFFERENT METHODS ON EMT DATASET. THE BEST RESULTS OF COMPARATIVE METHODS ARE HIGHLIGHTED IN BLUE

Testing set	Index	Lap. Reg. ($\mathbf{x_0}$)	FISTA-TV [60]	FBPConvNet [21]	ISTA-Net [31]	**FISTA-Net**
	# of Pars.	1	1	482449	262094	**74599**
set 1	PSNR	16.140	17.996	20.574	20.074	**21.304**
	SSIM	0.556	0.661	0.873	0.772	**0.912**
	RMSE	0.155	0.1259	0.094	0.099	**0.086**
set 2	PSNR	14.975	16.508	19.432	19.215	**20.013**
	SSIM	0.472	0.630	0.841	0.725	**0.882**
	RMSE	0.178	0.149	0.106	0.109	**0.099**

TABLE V

QUANTITATIVE METRICS OF DIFFERENT METHODS ON MAYO CLINIC CT DATASET. THE BEST RESULTS OF COMPARATIVE METHODS ARE HIGHLIGHTED IN BLUE

Testing set	Index	FBP ($\mathbf{x_0}$)	FISTA-TV [60]	FBPConvNet [21]	ISTA-Net [31]	**FISTA-Net**
	# of pars	0	1	482449	262094	**74599**
60 view	PSNR	27.533	33.251	33.976	33.734	**37.319**
	SSIM	0.704	0.866	0.942	0.920	**0.951**
	RMSE	0.042	0.021	0.020	0.021	**0.013**
120 view	PSNR	29.152	35.217	36.091	35.452	**40.189**
	SSIM	0.815	0.949	0.950	0.948	**0.968**
	RMSE	0.035	0.017	0.157	0.016	**0.009**

References

[1] (LISTA) Gregor K, LeCun Y. Learning fast approximations of sparse coding [C]//Proceedings of the 27th international conference on international conference on machine learning. 2010: 399-406.

[2] (ADMM-Net) Yang Y, Sun J, Li H, et al. Deep ADMM-Net for compressive sensing MRI [C]//Proceedings of the 30th International Conference on Neural Information Processing Systems. 2016: 10-18.

[3] (ISTA-Net) Zhang J, Ghanem B. ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 1828-1837.

[4] (FISTA-Net) Xiang J, Dong Y, Yang Y. FISTA-Net: Learning A fast iterative shrinkage thresholding network for inverse problems in imaging [J]. IEEE Transactions on Medical Imaging, 2021, 40(5): 1329-1339.