# LEARNING THE WEIGHT MATRIX FOR SPARSITY AVERAGING IN COMPRESSIVE IMAGING

*Dimitris Perdios*[⋆]*, Adrien Besson*[⋆]*, Philippe Rossinelli*[⋆]*, and Jean-Philippe Thiran*[⋆†]

[⋆]Signal Processing Laboratory (LTS5), Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[†]Department of Radiology, University Hospital Center (CHUV), Switzerland

## ABSTRACT

We propose to map the fast iterative shrinkage-thresholding algorithm to a deep neural network (DNN), with a sparsity prior in a concatenation of wavelet bases, in the context of compressive imaging. We exploit the DNN architecture to learn the optimal weight matrix of the corresponding reweighted $\ell_1$-minimization problem. We later use the learned weight matrix for the image reconstruction process, which is recast as a simple $\ell_1$-minimization problem. The approach, denoted as learned extended FISTA, shows promising results in terms of image quality, compared to state-of-the-art algorithms, and significantly reduces the reconstruction time required to solve the reweighted $\ell_1$-minimization problem.

***Index Terms*—** Compressed sensing, deep learning, fast iterative shrinkage-thresholding algorithm

## 1. INTRODUCTION

Compressed sensing (CS) extends the principle of Nyquist sampling to non-bandlimited signals exploiting the idea that most signals have concise representations, expressed in terms of sparsity, in well-chosen models [1, 2]. Let us denote as $\boldsymbol{x} \in \mathbb{R}^N$ the signal under scrutiny, measured with a linear operator $\mathsf{A} \in \mathbb{R}^{M \times N}$ such that $\boldsymbol{y} = \mathsf{A}\boldsymbol{x} + \boldsymbol{n}$, where $\boldsymbol{n} \in \mathbb{R}^M$ is the observation noise. Let us assume that $\boldsymbol{x}$ obeys a sparse representation in a basis $\Psi \in \mathbb{R}^{N \times N}$, *i.e.* that $\boldsymbol{x} = \Psi\boldsymbol{c}$ with $||\boldsymbol{c}||_0 = K < N$. For a sufficient number of measurements $M$, CS demonstrates that one can perfectly recover $\boldsymbol{x}$ from $\boldsymbol{y}$, with high probability, by solving the following analysis problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^N} \lambda ||\Psi^\dagger \boldsymbol{x}||_1 + \frac{1}{2}||\mathsf{A}\boldsymbol{x} - \boldsymbol{y}||_2^2, \tag{1}$$

where $\Psi^\dagger$ designates the adjoint of $\Psi$ and $||.||_q$ denotes the $\ell_q$-norm calculated as $||\boldsymbol{x}||_q = \sqrt[q]{\sum_{i=1}^N |x_i|^q}$. It has been demonstrated that Gaussian and Bernoulli random matrices are particularly suited to the CS framework. In this specific case, sampling rates requirements scale logarithmically with the size of the signal [3].

Candès *et al.* have later extended the CS framework to coherent and redundant dictionaries $\Psi$ [4] and have shown that solving a reweighted $\ell_1$-minimization problem may improve the signal reconstruction [5]. Carrillo *et al.* [6] have used this framework for compressive imaging by exploiting sparsity of images in multiple mutually coherent wavelet bases. Their technique, coined as sparsity averaging for reweighted analysis (SARA) aims at solving the following problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^N} ||\mathsf{W}\Psi^\dagger \boldsymbol{x}||_1 + \frac{1}{2}||\mathsf{A}\boldsymbol{x} - \boldsymbol{y}||_2^2, \tag{2}$$

where $\Psi \in \mathbb{R}^{N \times L}$, $L = qN$ is a concatenation of $q$ bases $\Psi_q$ and $\mathsf{W} \in \mathbb{R}^{L \times L}$ is a block-diagonal matrix made of $q$ blocks of size $N \times N$ with positive entries. Carrillo *et al.* [6] have shown that SARA outperforms many state-of-the-art image reconstruction algorithms.

The tremendous success of deep learning (DL) approaches for many image processing tasks has lead researchers to study in what extent DL may perform signal and image reconstruction. Gregor and Lecun [7] have bridged the gap between DL and CS by introducing the learned iterative shrinkage-thresholding algorithm (LISTA), in which the classical iterative shrinkage-thresholding algorithm (ISTA) is mapped onto a deep neural network (DNN). By learning the optimal parameters of the algorithm on a training set, LISTA outperforms classical ISTA and fast ISTA (FISTA). Kamilov and Mansour [8] have recently extended the work of Gregor and Lecun by learning the non-linearity of LISTA, based on a B-spline decomposition. Yang *et al.* [9] have applied the same procedure to map the alternating direction method of multipliers (ADMM) to a DNN and have shown promising results in terms of reconstruction time and image quality for CS in magnetic resonance imaging.

An alternative to the methods mentioned above consists in training classical DNN architectures, usually used to perform classification, for image reconstruction. Based on the success of autoencoders for non-linear image classification, Mousavi *et al.* [10] have applied a 3-layers stacked denoising autoencoder for CS and have shown that such a DNN architecture is competitive against state-of-the-art techniques. Kulkarni *et al.* [11] have adapted a convolutional neural network architecture to image reconstruction and have demonstrated a significant increase of the image quality compared to classical techniques.

In this work we propose an alternative to SARA proposed by Carrillo *et al.* [6]. The underlying idea is that one way to accelerate the algorithm may consist in learning the matrix $\mathsf{W}$ during a training step, using a DNN, and later use it for the image reconstruction, instead of alternating between an update step and an optimization step. To this aim, we suggest to use FISTA, introduced by Beck and Teboulle [12], to solve Problem (2). The choice of FISTA is guided by our ability to map it to a DNN allowing us to use backpropagation algorithms to learn the optimal weight matrix.

The remainder of the paper is organized as follows. The proposed approach is described in Section 2 and compared against SARA and other state-of-the-art algorithms in Section 3. Concluding remarks are given in Section 4.

This work was supported in part by the UltrasoundToGo RTD project (no. 20NA21_145911), funded by Nano-Tera.ch.

## 2. THE PROPOSED APPROACH

### 2.1. Mapping the fast iterative shrinkage-thresholding algorithm to a deep neural network

In order to achieve the mapping, we need to unfold FISTA, described in Algorithm 1, in a similar manner to Gregor and LeCun [7].

**Algorithm 1** FISTA used to solve Problem (2)

**Require:** $\mathsf{A}, \mathsf{W}, \Psi, \boldsymbol{y}, L \geq \lambda_{max}\left(\mathsf{A}^T\mathsf{A}\right)$
  **initialization:** $i = 1, t_0 = 1, \boldsymbol{x}_{-1} = \boldsymbol{x}_0 = \mathbf{0}$
  **repeat**
    $t_i \leftarrow \frac{1+\sqrt{1+4t_{i-1}^2}}{2}$
    $\alpha_i \leftarrow \frac{t_{i-1}-1}{t_i}, \quad \beta_i \leftarrow 1+\alpha_i$
    $\boldsymbol{c}_i \leftarrow \beta_i \boldsymbol{x}_{i-1} - \alpha_i \boldsymbol{x}_{i-2}$
    $\boldsymbol{x_i} \leftarrow \mathsf{prox}^{\mathsf{W}\Psi^\dagger}_{||.||_1}\left(\boldsymbol{c}_i + \frac{1}{L}\mathsf{A}^T\left(\boldsymbol{y} - \mathsf{A}\boldsymbol{c}_i\right); \frac{1}{L}\right)$
    $i \leftarrow i+1$
  **until** stopping criterion
  **return** $\boldsymbol{x}_i$

In Algorithm 1, $\mathsf{prox}^{\mathsf{W}\Psi^\dagger}_{||.||_1}(.;\tau)$ accounts for the proximity operator of the function $g : \boldsymbol{x} \mapsto \tau||\mathsf{W}\Psi^\dagger\boldsymbol{x}||_1$. Such a proximity operator can be rewritten as follows [13]:

$$\mathsf{prox}^{\mathsf{W}\Psi^\dagger}_{||.||_1}(\boldsymbol{x};\tau) = \sum_{k=1}^{q} \frac{\Psi_k}{\sqrt{q}} \mathsf{soft}\left(\frac{\Psi_k^\dagger}{\sqrt{q}}\boldsymbol{x}; \tau\mathsf{W}_k\right), \tag{3}$$

where $\mathsf{W}_k$ is a diagonal matrix which accounts for the $k^{th}$ block of $\mathsf{W}$. The operation $\mathsf{soft}(\boldsymbol{s};\mathsf{D})$ with a diagonal matrix $\mathsf{D}$ involves a soft-thresholding of the entries $s_i$ with respect to $\mathsf{D}_{ii}$, as described hereafter:

$$\mathsf{soft}(s_i;\tau\mathsf{D}_{ii}) = \mathsf{sign}\left(s_i\right)\left(|s_i| - \mathsf{D}_{ii}\right)_+.$$

We introduce the matrices $\mathsf{S} = \left(\mathsf{I} - \frac{1}{L}\mathsf{A}^T\mathsf{A}\right)$ and $\mathsf{G} = \frac{1}{L}\mathsf{A}^T$ and use Equation (3) to unfold FISTA, resulting in Algorithm (2).

**Algorithm 2** LEFISTA forward propagation

**Require:** $\mathsf{G}, \mathsf{S}, \mathsf{W}, \Psi, \boldsymbol{y}, L \geq \lambda_{max}\left(\mathsf{A}^T\mathsf{A}\right), T, q$
  **initialization:** $i = 1, t_0 = 1, \boldsymbol{x}_{-1} = \boldsymbol{x}_0 = \mathbf{0}$
  **repeat**
    $t_i \leftarrow \frac{1+\sqrt{1+4t_{i-1}^2}}{2}$
    $\alpha_i \leftarrow \frac{t_{i-1}-1}{t_i}, \quad \beta_i \leftarrow 1+\alpha_i$
    $\boldsymbol{z}_i \leftarrow \beta_i\mathsf{S}\boldsymbol{x}_{i-1} - \alpha_i\mathsf{S}\boldsymbol{x}_{i-2} + \mathsf{G}\boldsymbol{y}$
    **for** $k = 1$ to $q$ **do**
      $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \frac{\Psi_k}{\sqrt{q}}\mathsf{soft}\left(\frac{\Psi_k^\dagger}{\sqrt{q}}\boldsymbol{z}_i; \frac{1}{L}\mathsf{W}_k\right)$
    **end for**
    $i \leftarrow i+1$
  **until** $i = T$
  **return** $\left(\boldsymbol{x}_i\right)_{i=1}^T, \left(\boldsymbol{z}_i\right)_{i=1}^T, \left(\alpha_i\right)_{i=1}^T, \left(\beta_i\right)_{i=1}^T$

Each iteration of Algorithm (2) can be mapped to a feed-forward neural network, denoted as learned extended FISTA (LEFISTA), which architecture is described on Figure 1. On Figure 1(a), one may see the four first layers of LEFISTA, with a non-linearity function $\mathsf{f}$ defined on Figure 1(b).

### 2.2. Learning the weight matrix

In order to learn the weight matrix, we consider a training set made of $P$ pairs $\left(\boldsymbol{y}_p, \boldsymbol{x}_p^\star\right)_{p=1}^P$ where $\boldsymbol{x}_p^\star$ denotes the reference image and $\boldsymbol{y}_p$ the corresponding measurement vector. The objective of the learning procedure consists in finding $\mathsf{W}$ which minimizes the $\ell_2$-loss function over the training set.

To achieve such a minimization, the backpropagation-through-time (BPTT) algorithm is used, where the gradient of the loss function over each diagonal element of $\mathsf{W}$ is calculated using the chain rule. Using the same notations as the one described in the paper of Gregor and LeCun, *i.e.* by defining as $\delta\boldsymbol{x}$ the gradient of the loss over $\boldsymbol{x}$, as $(\delta\mathsf{W}_k)$ the gradient of the loss over the diagonal elements of each $\mathsf{W}_k$ and as $\mathsf{soft}'(\boldsymbol{s};\mathsf{D})$ the Jacobian of the soft-thresholding operator with respect to its inputs (diagonal matrix), the backpropagation can be derived as described in Algorithm 3.

**Algorithm 3** LEFISTA Backpropagation algorithm

**Require:** $\left(\boldsymbol{x}_i\right)_{i=1}^T, \left(\boldsymbol{z}_i\right)_{i=1}^T, \left(\alpha_i\right)_{i=1}^T, \left(\beta_i\right)_{i=1}^T, \mathsf{G}, \mathsf{S}, \mathsf{W}, \Psi, q$
  **initialization:** $i = T, \delta\boldsymbol{x}_{T+1} = \boldsymbol{x}_T - \boldsymbol{x}^\star, \left(\delta\mathsf{W}_k\right)_{k=1}^q = \mathbf{0}$, $\left(\delta\boldsymbol{z}_i\right)_{i=1}^{T+1} = \mathbf{0}$
  **repeat**
    **for** $k = 1$ to $q$ **do**
      $\delta\boldsymbol{h}_k \leftarrow \frac{\Psi_k}{q}\mathsf{soft}'\left(\frac{\Psi_k^\dagger}{\sqrt{q}}\boldsymbol{z}_i; \frac{1}{L}\mathsf{W}_k\right)\cdot\Psi_k^\dagger\delta\boldsymbol{x}_i$
      $\delta\boldsymbol{z}_i \leftarrow \delta\boldsymbol{z}_i + \delta\boldsymbol{h}_k$
      $\delta\mathsf{W}_k \leftarrow \delta\mathsf{W}_k - \mathsf{sign}\left(\frac{\Psi_k^\dagger}{\sqrt{q}}\boldsymbol{z}_i\right)\cdot\frac{\Psi_k^\dagger}{\sqrt{q}}\delta\boldsymbol{h}_k$
    **end for**
    $\delta\boldsymbol{x}_i \leftarrow \beta_i\mathsf{S}\delta\boldsymbol{z}_i - \alpha_i\mathsf{S}\delta\boldsymbol{z}_{i+1}$
    $i \leftarrow i-1$
  **until** $i = 1$
  **return** $\left(\delta\mathsf{W}_k\right)_{k=1}^q$

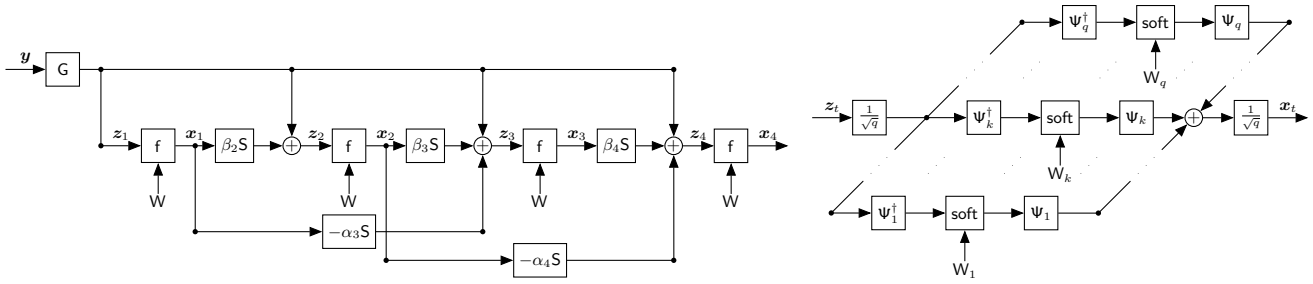
The learning procedure consists in alternating, over a fixed number of epochs, between estimating $\boldsymbol{x}$ for all the training samples with a fixed $\mathsf{W}$ using Algorithm 2 and updating $\mathsf{W}$ for a fixed set of estimates using Algorithm 3. The update step consists in applying a gradient descent: $\mathsf{W}_k \leftarrow \mathsf{W}_k - \epsilon\delta\mathsf{W}_k$ with $\epsilon$ the learning rate and $\left(\delta\mathsf{W}_k\right)_{k=1}^q$ computed with the BPTT algorithm.

### 2.3. Image reconstruction

DNN-based approaches require to work on image patches, which typical sizes range between $16\times16$ and $64\times64$ pixels, due to memory requirements during training. This conditions the image reconstruction process to be patch-based. In this work, we exploit the block-compressed-sensing (BCS) framework, introduced by Gan [14] which permits to use CS techniques in a block-based manner.

The idea consists in training LEFISTA with a given measurement matrix $\mathsf{A} \in \mathbb{R}^{M_B \times N_B^2}$ on patches of size $N_B \times N_B$ pixels. Once the network is trained, we divide each image of the test set into $B$ non-overlapping patches of size $N_B \times N_B$ pixels. We compressed each patch with the matrix $\mathsf{A}$ used in the training phase and we reconstruct each patch independently by applying Algorithm 2 with $\mathsf{W}$ learned in the training phase. The whole image is obtained by placing each reconstructed patch at its initial location in the canvas.

One remarkable property of such a technique is that it reduces the resolution of Problem (2) to the resolution of Problem (1), much faster to solve.




(a) (b)

**Fig. 1**. LEFISTA network architecture: (a) the 4-first layers of LEFISTA, with $\beta_i = (1 + \alpha_i)$ and (b) the non-linearity f involved in LEFISTA, composed of a soft-thresholding operator in multiple wavelet bases.

## 3. PERFORMANCE EVALUATION

### 3.1. Experimental settings

The proposed LEFISTA is tested with different number of layers, *i.e.* 30, 40 and 50. The patch size is set to $64 \times 64$ pixels and the measurement matrix $\mathsf{A} \in \mathbb{R}^{M_B \times N_B^2}$ is chosen with Gaussian random entries, with the measurement rate $M_B/N_B^2$ varying between 0.1 and 0.5. In order to fit with the framework described by Carrillo *et al.* [6], $\Psi$ is composed of a concatenation of 8 wavelet bases with Daubechies 1 to Daubechies 8 as mother function. Two decomposition levels are considered. LEFISTA is implemented[1] on TensorFlow and experiments have been performed on a NVIDIA Titan X GPU card. It is trained using mini-batch learning on 43 560 patches extracted from 1200 images of the ILSVRC 2014 ImageNet dataset. The chosen optimization algorithm is Adam with a learning rate of $10^{-5}$ and a batch size of 32. The number of epochs is fixed to 20.

The quality of the reconstruction is evaluated on a test set composed of 4 images, namely Barbara, Goldhill, Peppers and Lena. The peak-signal-to-noise ratio (PSNR) and the structural-similarity index (SSIM) are used as image quality metrics.

### 3.2. Comparison to SARA

LEFISTA is firstly compared to a tiled version of SARA, where the algorithm is applied on the same non-overlapping patches. In this case, the matrix A used for SARA reconstruction is the same Gaussian random matrix as before, the reconstruction algorithm is Douglas-Rachford with 200 iterations and 10 updates of the weight matrix.

The results, expressed in terms of PSNR and SSIM, and displayed on Table 1, show that LEFISTA outperforms SARA in most cases, especially at low measurement rates.

From Figure 2, which shows the test images reconstructed with SARA and LEFISTA (50 layers) respectively, for a measurement rate of 0.3, one can see that the visual quality is similar between the two methods. One may notice that the images reconstructed with LEFISTA are slightly more blurred than the ones reconstructed with SARA, whereas the blocking artifacts are less important with LEFISTA than with SARA. This can justify the difference observed in terms of PSNR.

One remarkable aspect of LEFISTA is that, since it solves a $\ell_1$-minimization problem rather than a reweigthed one, it takes a few seconds to reconstruct an image while SARA takes around one hour.

[1] https://github.com/dperdios/lefista

**Table 1**. Comparison, in terms of PSNR and SSIM, of LEFISTA with 30, 40 and 50 layers against a tiled version of SARA and state-of-the-art block compressed sensing algorithms, for Barbara, Goldhill, Peppers and Lena images, for different measurement rates.

| Algorithm | Measurement rate | | | | | |
|---|---|---|---|---|---|---|
| | PSNR [dB] | | | SSIM [–] | | |
| | 0.1 | 0.3 | 0.5 | 0.1 | 0.3 | 0.5 |
| **Barbara** | | | | | | |
| Tiled SARA | 16.15 | 24.90 | **29.70** | 0.38 | 0.79 | **0.91** |
| BCS-SPL-DWT | 21.87 | 24.31 | 27.06 | 0.40 | 0.61 | 0.75 |
| BCS-SPL-DDWT | 22.11 | 24.74 | 27.84 | 0.40 | 0.62 | 0.76 |
| MS-BCS-SPL-DWT | 22.17 | 24.86 | 27.99 | 0.41 | 0.63 | 0.77 |
| LEFISTA-LN30 | 22.52 | 25.38 | 28.94 | 0.59 | 0.77 | 0.88 |
| LEFISTA-LN40 | 22.65 | 25.61 | 29.19 | 0.60 | 0.78 | 0.89 |
| LEFISTA-LN50 | **22.77** | **25.73** | 29.29 | **0.61** | **0.79** | 0.90 |
| **Goldhill** | | | | | | |
| Tiled SARA | 18.56 | 29.76 | 33.19 | 0.45 | 0.81 | 0.90 |
| BCS-SPL-DWT | 24.57 | 30.40 | 33.06 | 0.42 | 0.68 | 0.80 |
| BCS-SPL-DDWT | 25.18 | 30.45 | 33.11 | 0.42 | 0.68 | 0.80 |
| MS-BCS-SPL-DWT | 26.74 | 30.57 | 33.19 | 0.44 | 0.68 | 0.80 |
| LEFISTA-LN30 | 26.41 | 30.20 | 33.35 | 0.64 | 0.80 | 0.89 |
| LEFISTA-LN40 | 26.63 | 30.63 | 33.89 | 0.65 | 0.82 | 0.90 |
| LEFISTA-LN50 | **26.77** | **30.93** | **34.17** | **0.66** | **0.83** | **0.91** |
| **Peppers** | | | | | | |
| Tiled SARA | 18.71 | 32.81 | 35.35 | 0.44 | 0.84 | 0.90 |
| BCS-SPL-DWT | 27.73 | 33.38 | 35.92 | 0.45 | 0.65 | 0.76 |
| BCS-SPL-DDWT | 28.09 | 33.52 | 36.26 | 0.45 | 0.65 | 0.77 |
| MS-BCS-SPL-DWT | 28.22 | 33.63 | 36.33 | 0.45 | 0.65 | 0.77 |
| LEFISTA-LN30 | 27.32 | 32.53 | 35.52 | 0.74 | 0.85 | 0.89 |
| LEFISTA-LN40 | 27.93 | 33.21 | 36.07 | 0.75 | 0.86 | 0.90 |
| LEFISTA-LN50 | **28.45** | **33.72** | **36.34** | **0.77** | **0.87** | **0.91** |
| **Lena** | | | | | | |
| Tiled SARA | 17.93 | 33.04 | 36.84 | 0.51 | 0.89 | 0.94 |
| BCS-SPL-DWT | 26.63 | 32.81 | 36.21 | 0.44 | 0.68 | 0.80 |
| BCS-SPL-DDWT | 27.49 | 33.24 | 36.67 | 0.45 | 0.69 | 0.81 |
| MS-BCS-SPL-DWT | 27.53 | **33.46** | **36.86** | 0.46 | 0.70 | 0.81 |
| LEFISTA-LN30 | 27.41 | 32.27 | 35.90 | 0.75 | 0.87 | 0.92 |
| LEFISTA-LN40 | 27.76 | 32.93 | 36.48 | 0.77 | 0.89 | 0.93 |
| LEFISTA-LN50 | **28.04** | 33.34 | 36.78 | **0.78** | **0.89** | **0.94** |

**Fig. 2**. Test images (from left to right: Barbara, Goldhill, Peppers and Lena) reconstructed for a measurement rate $M_B/N_B^2 = 0.3$ (first row) with SARA and (second row) with LEFISTA (50 layers).

### 3.3. Comparison to state-of-the-art block compressed sensing algorithms

We also compare the proposed approach to state-of-the-art BCS reconstruction methods, namely BCS with smooth projected Landweber with the dual tree wavelet transform (BCS-SPL-DDWT) and the discrete wavelet transform (BCS-SPL-DWT) and multiscale BCS with the smooth projected Landweber and the discrete wavelet transform (MS-BCS-SPL-DWT) [15]. The code used for both methods is provided by their authors.

The results, displayed on Table 1, show that LEFISTA outperforms BCS reconstruction methods in terms of both PSNR and SSIM. It can be noticed that the increase is more significant on the SSIM. One possible explanation is that BCS-SPL-DWT, BCS-SPL-DDWT and MS-BCS-SPL-DWT use a Wiener filter at each iteration of the reconstruction algorithm to avoid blocking artifacts. This results in an over-smoothing of the image which has a significant impact on the SSIM.

One may notice that, as expected, LEFISTA may achieve better image quality by increasing the number of layers.

## 4. CONCLUSION

In this work, we propose to map the fast iterative shrinkage-thresholding algorithm to a deep neural network (DNN). We use this architecture to learn the weight matrix of a weighted $\ell_1$-minimization problem, with a prior of sparsity in a concatenation of wavelet bases. This learned matrix is later used in the problem, which is recast as a classical $\ell_1$-minimization problem, much faster to solve than the reweighted $\ell_1$-minimization problem. The proposed approach leads to promising results in terms of image quality and opens the way to more advanced models, where the DNN architecture is not only used to learn the weight matrix but also to learn the non-linearities and the different matrices involved in the network.

## 5. REFERENCES


[1] E. J. Candès, "Compressive sampling," in *Proc. Int. Congr. Math.*, vol. 17, 2009, pp. 295–303.

[2] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 21–30, 2008.

[3] E. J. Candès and J. Romberg, "Sparsity and incoherence in compressive sampling," *Inverse Probl.*, vol. 23, no. 3, pp. 969–985, 2007.

[4] E. J. Candès, Y. C. Eldar, D. Needell, and P. Randall, "Compressed sensing with coherent and redundant dictionaries," *Appl. Comput. Harmon. Anal.*, vol. 31, no. 1, pp. 59–73, 2011.

[5] E. J. Candès, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted $\ell_1$-minimization," *J. Fourier Anal. Appl.*, vol. 14, no. 5-6, pp. 877–905, 2008.

[6] R. E. Carrillo, J. D. McEwen, D. Van De Ville, J.-P. Thiran, and Y. Wiaux, "Sparsity averaging for compressive imaging," *IEEE Signal Process. Lett.*, vol. 20, no. 6, pp. 591–594, 2013.

[7] K. Gregor and Y. LeCun, "Learning Fast Approximations of Sparse Coding," in *27th Int. Conf. Mach. Learn. ICML 2010*, 2010, pp. 399–406.

[8] U. S. Kamilov and H. Mansour, "Learning optimal nonlinearities for iterative thresholding algorithms," *IEEE Signal Process. Lett.*, vol. 23, no. 5, pp. 747–751, 2016.

[9] Y. Yang, J. Sun, H. Li, and Z. Xu, "Deep ADMM-Net for compressive sensing MRI," in *Adv. Neural Inf. Process. Syst. 29*, 2016, pp. 10–18.

[10] A. Mousavi, A. B. Patel, and R. G. Baraniuk, "A deep learning approach to structured signal recovery," in *2015 53rd Annu. Allert. Conf. Commun. Control. Comput.*, 2015, pp. 1336–1343.

[11] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, "Reconnet: Non-iterative reconstruction of images from compressively sensed measurements," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016.

[12] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[13] M. Fadili and J.-L. Starck, "Monotone operator splitting for optimization problems in sparse recovery," in *16th IEEE Int. Conf. Image Process.*, 2009, pp. 1461–1464.

[14] L. Gan, "Block compressed sensing of natural images," in *15th Int. Conf. Digit. Signal Process.*, 2007, pp. 403–406.

[15] J. E. Fowler, S. Mun, and E. W. Tramel, "Multiscale block compressed sensing with smoothed projected Landweber reconstruction," in *19th Eur. Signal Process. Conf.*, 2011, pp. 564–568.