Sparse Recovery of Time-Frequency Representations via Recurrent Neural Networks

Yassin Khalifa, Zhenwei Zhang, Ervin Sejdić
Electrical and Computer Engineering, University of Pittsburgh, PA, USA
E-mail: esejdic@ieee.org

***Abstract*—The impressive evolution of neural networks and deep learning techniques during the last few years has offered new incomparable routes to solve many complex problems. Moreover, the fact that neural networks are structured and supervised has made it possible to perform automatic parameter tuning that guarantees convergence to the best expressive model for the problem assessed. In this work, we investigated the use of recurrent neural networks (RNNs) to solve the sequential sparse recovery problem through unfolding the iterative soft thresholding algorithm (ISTA) into a stacked RNN. Specifically, we examined the performance of the unsupervised iterative algorithm and the supervised network for a purely compressive sampling reconstruction problem of time-frequency representations. Our results demonstrated that the trained stacked neural network outperforms the iterative algorithm in the quality of the reconstructed data and points to several future directions to improve the performance.**

I. Introduction

With the rapid development of information technology, traditional signal processing schemes are not suitable for the vast amount of data. In recent years, many research contributions in statistics, machine learning, and computational neuroscience have considered the compressed sensing theory due to its various potential applications. In signal processing, sparse coding represents data with linear combinations of a few sparse coefficients. The basic aim is to determine a sparse coefficient set based on noisy observations and recover denoised signals by exploiting the sparsity of these coefficients.

In many domains, such as time series, video analysis, image captions, the inputs of systems are sequences. Unlike standard neural networks, which assumes the independence among training and test data, RNNs process the sequential data one by one, which can be able to selectively pass information across steps. Besides, RNN can model inputs consisting of sequences that are not independent.

Several work has been taken for achieving sparse signal representation such as basis pursuit method [1], iterative threshold algorithm [2], matching pursuit algorithm [3], interior-point method [4] and etc. Iterative thresholding algorithms plays an important roles in sparse signal processing. Several variants of ISTA were proposed in [5], [6]. Donoho *et al.* proposed message-passing algorithms which is inspired by iterative-thresholding and belief propagation in graphical models [7].

Most recently, several researches have extended the algorithms to neural networks. For example, Gregor *et al.* proposed a learned iterative soft-thresholding algorithm, which reduces the number of iterations and enhances the performance of the original ISTA algorithm [8] via a data-adaptive approach. Salman proposed a homotopy-based algorithm for sparse recovery of streaming signals [9]. A nonlinear diffusion model was proposed in [10], which uses parametrized linear filters learned from training sets. Kamilov *et al.* related iterations of ISTA to layers of a simple deep neural networks to learn ISTA nonlinearities from data [11]. Mousavi *et al.* developed a deep learning framework for structured signal recovery that both supports linear and mildly nonlinear measurements [12]. Palangi *et al.* applied a constitutional deep stacking network to reconstruct sparse vectors in multiple measurements [13]. Wisdom *et al.* proposed an interpretable RNN based on the sequential iterative soft-thresholding algorithm [14] instead of training a black box neural network.

In this work, we review the sparse recovery and sequential sparse recovery problems with the use of iterative soft thresholding algorithm as a solution and how this solution can be unfolded into a stacked RNN. Then, we assess the constructed network by testing its ability to construct time-frequency representations compared to a baseline method.

II. Materials and Methods

A. Sparse Recovery

The sparse recovery problem for a single static vector can be formulated as the solution of an optimization problem to find the sparse representation vector $\hat{h} \in \mathbb{R}^N$ that minimizes error between observation x and the reconstruction $AD\hat{h}$. The matrix $A \in \mathbb{F}^{M \times N}$ is the measurement matrix with $M < N$ for a typical compressive sampling problem and $D \in \mathbb{R}^{N \times N}$ is a transformation (dictionary) matrix with basis vectors as columns [14]. Using Lagrangian multiplier the problem can be stated as follows:

$$\min_h \frac{1}{2}||x - ADh||_2^2 + \lambda||h||_1 \tag{1}$$

The first part of the problem represents error to enforce data consistency and the second term (ℓ_1-norm) guarantees the sparsity of h, so that $s = Dh$ has the smallest number of basis vectors. Eqn. (1) is called basis pursuit denoising (BPDN) [1] and is equivalent to the Lagrangian of the least absolute shrinkage and selection operator (LASSO) method for sparse recovery [14], [15].

The solution of Eqn. (1) is based on a probabilistic model where $x = ADh + \epsilon$ and ϵ is a zero-mean Gaussian noise [14].

In addition, h is considered to have a zero-mean Laplacian prior with scale β and $\lambda = 2\sigma^2/\beta$ [14]:

$$\begin{aligned} x &\sim \mathcal{N}(ADh, \sigma^2 I), \\ h_n &\sim Laplace(0, \beta) \ for \ n = 1, \ldots, N. \end{aligned} \tag{2}$$

If we considered a sequence of not necessarily independent observations $x_t,\ t = 1, \ldots, T$, the problem is called sequential sparse recovery [14]. To model this problem, correlation between successive sparse coefficient vectors (h_t) is assumed to exist such that a signal $s_t = Dh_t$ can be linearly predicted through $s_t = Fs_{t-1} + v_t$ and v_t is the error modeled as zero-mean Gaussian noise [14]. This modifies the probabilistic model of the recovery problem so that correlation between h_t and h_{t-1} is considered as follows:

$$\begin{aligned} x_t \sim &\mathcal{N}(ADh_t, \sigma^2 I), \\ p(h_t|h_{t-1}) \propto &exp\left\{-\nu_1||h_t||_1 - \frac{\nu_2}{2}||Dh_t - FDh_{t-1}||_2^2\right\}. \end{aligned} \tag{3}$$

$$\begin{aligned} \min_{h_{1:T}} \sum_{t=1}^{T} (&\frac{1}{2}||x_t - ADh_t||_2^2 + \lambda_1||h_t||_1 \\ &+ \frac{\lambda_2}{2}||Dh_t - FDh_{t-1}||_2^2) \end{aligned} \tag{4}$$

where $\lambda_1 = 2\sigma^2\nu_1$ and $\lambda_2 = 2\sigma^2\nu_2$.

Algorithm 1: Sequential Iterative Soft Thresholding Algorithm (SISTA)

Input: observation sequence $x_{1:T}$, measurement matrix A dictionary D, predictor F, initial coefficients $\hat{h}_0$

for $t = 1$ *to* T **do**
  $\mathbf{h}_t^{(0)} \leftarrow \mathbf{D}^T\mathbf{F}\mathbf{D}\hat{\mathbf{h}}_{t-1}$ #`Initial estimate for` h_t
  for $k = 1$ *to* K **do**
    $\mathbf{z} \leftarrow [\mathbf{I} - \frac{1}{\alpha}\mathbf{D}^T(\mathbf{A}^T\mathbf{A} + \lambda_2\mathbf{I})\mathbf{D}]\mathbf{h}_t^{(k-1)} + \frac{1}{\alpha}\mathbf{D}^T\mathbf{A}^T\mathbf{x}_t$
    $\mathbf{h}_t^{(k)} \leftarrow soft_{\lambda_1/\alpha}(\mathbf{z} + \frac{\lambda_2}{\alpha}\mathbf{D}^T\mathbf{F}\mathbf{D}\hat{\mathbf{h}}_{t-1})$
  end
  $\hat{\mathbf{h}}_t \leftarrow \mathbf{h}_t^{(K)}$ #`Assign estimate for` h_t
end
return $\hat{\mathbf{h}}_{1:T}$

Many methods have been used to solve Eqns. (1) and (4) including projections onto convex sets [16], iteratively weighted least squares [17], non linear conjugate gradient with fast and cheap backtracking line-search [18], [19], and iterative soft thresholding [20]. The work in [14] used ISTA, so the same method will be considered for this work. The ISTA algorithm used for sequential sparse recovery is described in Algorithm 1 [14], where the function $soft_b(z_n)$ represents an element wise soft thresholding operation for a vector z with a threshold b and it can be defined as follows:

$$soft_b(z_n) = \frac{z_n}{|z_n|}\max(|z_n| - b, 0) \tag{5}$$

Algorithm 1 represents a modified version of the ISTA algorithm that considers a sequence of observations instead of a single static observation [14]. The algorithm uses linear prediction ($\mathbf{D}^T\mathbf{F}\mathbf{D}\hat{\mathbf{h}}_{t-1}$) to calculate the initial estimate of $h_t^{(0)}$ for each observation, also $h_t^{(K)}$ is taken as the optimal estimate of $\hat{h}_t$ and used to feed the linear prediction used to initialize $h_t^{(0)}$ for the next observation.

B. Sequential Sparse Recovery as A Stacked RNN

In a typical RNN, the output sequence $\hat{y}_{1:T}$ is computed from input sequence $x_{1:T}$ through the following nonlinear relationship:

$$\mathbf{h}_t = \sigma_b(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{V}\mathbf{x}_t) \tag{6}$$

$$\hat{\mathbf{y}}_t = \mathbf{U}\mathbf{h}_t + \mathbf{c}, \tag{7}$$

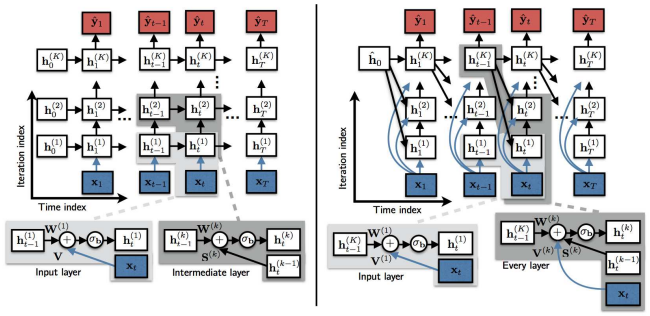


Fig. 1. The left part describes the generic stacked RNN and the right part describes the unfolded SISTA-RNN with the input x_t connected to every recurrent transition and every vertical iteration layer [14].

Nonlinearities in the previous model are introduced through the use of σ_b which is usually a non linear function (e.g. Sigmoid or ReLU) [14]. RNN is trained through a supervised set of I input-output pairs to minimize a cost function using backpropagation to calculate the so called "trainable" RNN prameters. Trainable RNN parameters include the initial hidden state h_0, the transition matrix W, the input matrix V, the output matrix U, bias c, and the nonlinearity parameter b. RNN can also be extended by stacking multiple hidden layers to create more efficient networks in means of deep transitions between consecutive hidden states [21]–[24].

For a multilayer stacked RNN (shown in the left diagram of Fig. 1), a linear transformation $S^{(k)}$ relates the hidden state of the current layer $h_t^{(k)}$ to the hidden state of the previous layer $h_t^{(k-1)}$ in addition to the preactivation of linearity [14]. Also, the output sequence $\hat{y}_{1:T}$ is calculated from the hidden states of the final layer $h_{1:T}^{(K)}$. This increases the trainable parameters of the network as it appears in the following model:

$$\mathbf{h}_t^k = \begin{cases} \sigma_b(\mathbf{W}^{(1)}\mathbf{h}_{t-1}^{(1)} + \mathbf{V}\mathbf{x}_t), & \text{k=1,} \\ \sigma_b(\mathbf{W}^{(k)}\mathbf{h}_{t-1}^{(k)} + \mathbf{S}^k\mathbf{h}_t^{(k-1)}), & \text{k=2,...,K ,} \end{cases} \tag{8}$$

$$\hat{\mathbf{y}}_t = \mathbf{U}\mathbf{h}_t^{(K)} + \mathbf{c} \tag{9}$$

$$\theta_{RNN} = \{\hat{\mathbf{h}}_0, \mathbf{b}^{1:K}, \mathbf{W}^{1:K}, \mathbf{V}^{1:K}, \mathbf{S}^{1:K}, \mathbf{U}, \mathbf{c}\} \tag{10}$$

Considering the SISTA algorithm for our sparse recovery problem, it can be represented as a stacked RNN using a set conditions that will make the RNN an unfolded version of

the SISTA algorithm [14]. In other words these conditions guarantee that the network output $\hat{y}_{1:T}$ will be the same as the SISTA reconstructed data $\hat{s}_{1:T} = \mathbf{D}\hat{\mathbf{h}}_{1:T}$. These set of conditions include:

1) The nonlinearity function is the soft thresholding operation at (5), with a bias $b_n = (\lambda_1/\alpha)$ for $n = 1, \ldots, N$.
2) The input sequence $x_{1:T}$ is involved in calculation of each hidden state $h_{1:T}^{(1:K)}$ through the matrices $V^{(1:K)}$.
3) previous hidden state estimate is used as $\hat{h}_{t-1} = h_{t-1}^{(K)}$ instead of $\hat{h}_{t-1} = h_{t-1}^{(k)}$ in RNN.
4) using the linear prediction relation $\mathbf{P} = \mathbf{D}^T\mathbf{F}\mathbf{D}$ so that:

$$\mathbf{V}^{(k)} = \frac{1}{\alpha}\mathbf{D}^T\mathbf{A}^T, \quad \forall k, \tag{11}$$

$$\mathbf{S}^{(k)} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^T(\mathbf{A}^T\mathbf{A} + \lambda_2\mathbf{I}\mathbf{D}), \quad k > 1, \tag{12}$$

$$\mathbf{W}^{(1)} = \frac{\alpha + \lambda_2}{\alpha}\mathbf{P} - \frac{1}{\alpha}\mathbf{D}^T(\mathbf{A}^T\mathbf{A} + \lambda_2\mathbf{I})\mathbf{D}\mathbf{P}, \tag{13}$$

$$\mathbf{W}^{(k)} = \frac{\lambda_2}{\alpha}\mathbf{P}, \quad k > 1, \tag{14}$$

$$\mathbf{U} = \mathbf{D} \quad , \ \mathbf{c} = 0 \tag{15}$$

Using the supervised dataset, trainable parameters of the network θ_{RNN} can be learned through minimizing a cost function of the error subject to equivalence with the structure of the original SISTA algorithm. This optimization problem is represented as:

$$\begin{aligned} \min_{\theta} \quad & \sum_{i=1}^{I} f(\hat{\mathbf{h}}_{1:T,i}, \mathbf{y}_{1:T,i}) \\ \text{subject to} \quad & \hat{\mathbf{h}}_{1:T,i} = g_\theta(\hat{\mathbf{x}}_{1:T,i}), \quad \text{i=1, ..., I,} \end{aligned} \tag{16}$$

where f is the mean squared error between the network outputs $\hat{y}_{1:T,i} = \mathbf{U}\hat{\mathbf{h}}_{1:T,i} + \mathbf{c}$ and the training references $y_{1:T,i} = s_{1:T,i}$ and g_θ is the computational paradigm of the SISTA algorithm [14].

C. SISTA Network Training

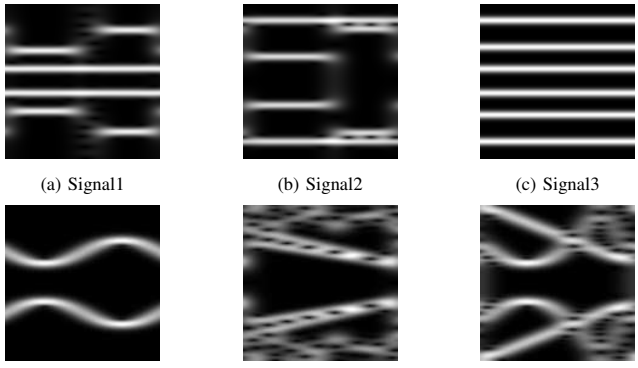
In this work, we used the same experimental setup in [14], [9] with slight modifications in training data designation to serve our objective. The training data is the complete set of images in the Caltech-256 dataset [25], which is a collection of multi-size 30607 color images. The dataset was divided randomly into two parts, the first part has 80% of the data and was used for training. The other part has 20% of the data and was used for validation. All images were converted to grayscale and resized to 128×128 after clipping out central squared regions [14].

The used dictionary $\mathbf{D}$ was composed of Daubechies-8 orthogonal (4 level) wavelets and the prediction matrix $\mathbf{F}$ was used as the identity matrix due to the expected slow column to column changes in natural images [14]. The observation or measurement matrix $\mathbf{A}$ was created with a compression ratio of 4 and dimensions of 32×128 . The values of $\mathbf{A}$ were chosen randomly with equal probability from $\pm 1/3\sqrt{32}$ so that the norm of the matrix $\mathbf{AD}$ is less than 1 and the SISTA algorithm will converge with a fixed step $\alpha = 1$ [14], [17]. ℓ_1-homotopy algorithm described in [9], was used as an unsupervised baseline to compare the results of the stacked RNN with oracle initialization ($\hat{\mathbf{h}}_0$ is set to first column of $\mathbf{D}_T\mathbf{s}_0$) and a 3 time steps joint optimization at once [14].

The SISTA RNN was trained with $K = 3$ (hidden layers or iterations) and soft-thresholding as the non-linear function. The network parameters were initialized randomly and all implementations were done using Python and Theano [26]. The cost function for training was used MSE as mentioned before and optimized via stochastic gradient descent and back-propagation with an initial learning rate of 10^{-4}, a minibatch size of 50, and RMSProp [27] with momentum 0.9 and averaging parameter 0.1 [14].

D. SISTA Network Testing

To assess the ability of the trainded SISTA-RNN for sparse recovery and the quality of its output compared to the original data, we decided to use a set of time-frequency representations, specifically, spectrograms [28] with the same dimensions used for training of the network (128×128). This set consists of six 2D sampled frames, each of them contains a group of synthesized sinusoids with constant or modulated frequencies and different phase shifts. The time-frequency representations of sample signals are shown in Fig. 2. Test data was introduced to the network directly as an input to the measurement matrix A instead of passing through the dictionary matrix D due to the fact that we are using time frequency data to test the performance of the network, and the same data was used with ℓ_1-homotopy algorithm too.



(a) Signal1 (b) Signal2 (c) Signal3

(d) Signal4 (e) Signal5 (f) Signal6

Fig. 2. Time Frequency data (6 2D frames) used to test the SISTA-RNN

The quality of the resultant data was assessed using signal to noise ratio (SNR) and peak SNR which proved to be sensitive to data degradation and fidelty [29], [30]. SNR and PSNR are defined as follows:

$$SNR = 10\log_{10}\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(f_{i,j} - mean(f))^2}{\sum_{i=1}^{N}\sum_{j=1}^{N}(g_{i,j} - f_{i,j})^2} \tag{17}$$

$$MSE = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(g_{i,j} - f_{i,j})^2 \tag{18}$$

$$PSNR = -10\log_{10}\frac{MSE}{g_{max}^2} \tag{19}$$

where $g_{i,j}$ and $f_{i,j}$ represents an element of the reconstructed data and the original data respectively. Also, g_{max} is the maximum possible amplitude for time frequency data (1).

III. Results and Discussion

After training and feeding the network with the test data according to the procedure described previously, we obtained reconstructed time-frequency representations as shown in Fig. 3. The whole set of quality assessment results is shown in table I that compares between the quality measures of reconstruction using both SISTA-RNN and ℓ_1-homotopy. From these results, we can clearly see that SISTA-RNN outperforms the considered baseline when it comes to the reconstruction of time-frequency representations, besides it has a relatively fast learning rate compared to the conventional RNN.

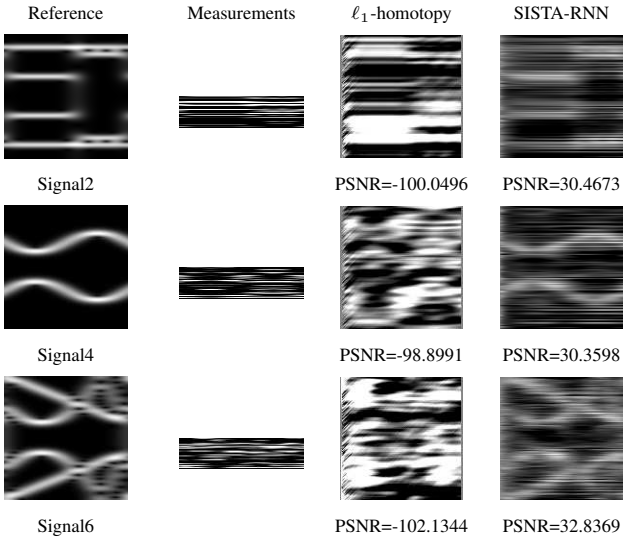


Fig. 3. Some reconstruction results from both ℓ_1-homotopy and SISTA-RNN

TABLE I
SNR and PSNR values of reconstruction results for both ℓ_1-homotopy and SISTA-RNN

Signal ID	SNR		PSNR	
	ℓ_1-homotopy	SISTA-RNN	ℓ_1-homotopy	SISTA-RNN
Signal1	-125.2385	3.0381	-98.9472	30.2975
Signal2	-127.5343	2.9826	-100.0496	30.4673
Signal3	-121.5789	7.4312	-100.1287	28.8813
Signal4	-128.1463	1.1126	-98.8991	30.3598
Signal5	-133.8410	5.4853	-104.0578	35.2685
Signal6	-129.9589	5.0124	-102.1344	32.8369

IV. Conclusion

In this work, we showed how unfolding the sequential iterative soft thresholding algorithm into a stacked multilayer RNN (SISTA-RNN) can be useful for the reconstruction of time-frequency representations. The results confirmed that training a supervised model can be a powerful way to improve the convergence of sparse recovery algorithm even when using random parameter initialization. Moreover, SISTA-RNN presented reconstructions of better quality compared to the conventional implementations of the sparse recovery problem.

References

[1] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, Jan. 2001.

[2] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[3] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[4] K. Fountoulakis, J. Gondzio, and P. Zhlobich, "Matrix-free interior point method for compressed sensing problems," *Mathematical Programming Computation*, vol. 6, no. 1, pp. 1–31, 2014.

[5] J. M. Bioucas-Dias and M. A. Figueiredo, "A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration," *IEEE Transactions on Image Processing*, vol. 16, no. 12, pp. 2992–3004, 2007.

[6] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[7] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18 914–18 919, 2009.

[8] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 399–406.

[9] M. S. Asif and J. Romberg, "Sparse recovery of streaming signals using ℓ_1-homotopy," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4209–4223, 2014.

[10] Y. Chen, W. Yu, and T. Pock, "On learning optimized reaction diffusion processes for effective image restoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5261–5269.

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

[12] A. Mousavi, A. B. Patel, and R. G. Baraniuk, "A deep learning approach to structured signal recovery," in *Proceedings of 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2015, pp. 1336–1343.

[13] H. Palangi, R. Ward, and L. Deng, "Convolutional deep stacking networks for distributed compressive sensing," *Signal Processing*, vol. 131, pp. 181–189, 2017.

[14] S. Wisdom, T. Powers, J. Pitton, and L. Atlas, "Building recurrent networks by unfolding iterative thresholding for sequential sparse recovery," in *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[15] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[16] E. J. Candes and J. K. Romberg, "Signal recovery from random projections," vol. 5674, 2005, pp. 76–86.

[17] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[18] T. c. Chang, L. He, and T. Fang, "MR image reconstruction from sparse radial samples using Bregman iteration," in *Proceedings of the 14th Annual Meeting of ISMRM*, 2006.

[19] M. M. Bronstein, A. M. Bronstein, M. Zibulevsky, and H. Azhari, "Reconstruction in diffraction ultrasound tomography using non-uniform FFT," *IEEE Transactions on Medical Imaging*, vol. 21, no. 11, pp. 1395–1401, 2002.

[20] A. Chambolle, R. A. DeVore, N. Lee, and B. J. Lucier, "Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 319–335, 1998.

[21] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression." *Neural Computation*, vol. 4, no. 2, pp. 234–242, 1992.

[22] R. Pascanu, Ç. Gülçehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv*, vol. 1312.6026, 2013.

[23] S. El Hihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies," in *Proceedings of the 8th International Conference on Neural Information Processing Systems*, ser. NIPS'95. Cambridge, MA, USA: MIT Press, 1995, pp. 493–499.
[24] A. Graves, "Generating sequences with recurrent neural networks," *arXiv*, vol. 1308.0850, 2013.
[25] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep. 7694, 2007.
[26] T. D. Team, "Theano: A python framework for fast computation of mathematical expressions," *arXiv*, vol. 1605.02688, 2016.
[27] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, 2012.
[28] E. Sejdić, I. Djurović, and J. Jiang, "Time–frequency feature representation using energy concentration: An overview of recent advances," *Digital Signal Processing*, vol. 19, no. 1, pp. 153–183, 2009.
[29] Y. Zhang, H. Cheng, J. Huang, and X. Tang, "An effective and objective criterion for evaluating the performance of denoising filters," *Pattern Recognition*, vol. 45, no. 7, pp. 2743 – 2757, 2012.
[30] C. P. Loizou and C. S. Pattichis, "Despeckle filtering algorithms and software for ultrasound imaging," *Synthesis Lectures on Algorithms and Software in Engineering*, vol. 1, no. 1, pp. 1–166, 2008.