Beyond ℓ_1: Data Driven Sparse Signal Recovery using DeepInverse

Ali Mousavi and Richard G. Baraniuk
Rice University
Houston, TX 77005

***Abstract*—We study a novel sparse signal recovery framework called *DeepInverse* that learns the inverse transformation from measurement vectors to signals using a deep convolutional network. We compare DeepInverse with ℓ_1-minimization from the phase transition point of view and demonstrate that it outperforms ℓ_1-minimization in the regions of phase transition plot where ℓ_1-minimization cannot recover the exact solution.**

I. Introduction

Sparse recovery is the problem of estimating a sparse signal $\mathbf{x} \in \mathbb{R}^N$ from a set of undersampled linear random measurements $\mathbf{y} = \mathbf{\Phi x} \in \mathbb{R}^M$, where $\mathbf{\Phi}$ is an $M \times N$ measurement matrix. By sparse, we mean that $\mathbf{x} = \mathbf{\Psi s}$, where $\mathbf{\Psi}$ is a basis and only $K \ll N$ of the coefficients $\mathbf{s}$ are nonzero. An alternative to the NP-hard, ℓ_0-minimization of finding the sparsest signal $\widehat{\mathbf{x}}$ such that $\mathbf{y} = \mathbf{\Phi}\widehat{\mathbf{x}}$ is the convex-relaxed, ℓ_1-minimization $\min \|\widehat{\mathbf{x}}\|_1,$ s.t. $\mathbf{y} = \mathbf{\Phi}\widehat{\mathbf{x}}$ [1], [2].

The price we pay for using ℓ_1-minimization instead of ℓ_0-minimization is reduced recovery performance, namely that ℓ_1-minimization requires more measurements M to recover a K-sparse signal than ℓ_0-minimization. Let $\delta = \frac{M}{N}$ denote the undersampling ratio and let $\rho = \frac{K}{M}$ indicate the normalized sparsity level. The two-dimensional *phase transition* plot $(\delta, \rho) \in [0,1]^2$ has two phases: a success phase and a failure phase, where ℓ_1-minimization can and cannot recover the exact signal, respectively. In other words, ℓ_1-minimization successfully recovers the sparse signal if its normalized sparsity level is less than a certain threshold. Figure 2 displays a typical phase transition plot for ℓ_1-minimization [3].

In this paper, we study a novel signal recovery framework we call *DeepInverse* in the context of sparse recovery. DeepInverse learns the inverse transformation from measurement vectors $\mathbf{y}$ to signals $\mathbf{x}$ using a deep convolutional network. When the network is trained on a set of sparse signals, it learns both a representation for the signals and an inverse map from measurement vectors to sparse signals. Compared to ℓ_1-minimization, our experiments below indicate that DeepInverse offers better sparse signal recovery performance for signals whose normalized sparsity is significantly larger than the threshold imposed by the ℓ_1-minimization phase transition. In other words, DeepInverse has better performance than ℓ_1-minimization on the failure side of ℓ_1 phase transition. Furthermore, our experiments show that DeepInverse can recover signals from measurement vectors tens of times faster than conventional sparse recovery algorithms. The tradeoff for the ultrafast run time is a one-time, computationally intensive, off-line training procedure typical to deep networks. This makes our approach applicable for real-time sparse recovery problems.

II. DeepInverse Framework

In this section we briefly describe the DeepInverse framework [4] for sparse signal recovery (see Figure 1) . DeepInverse takes as input a set of measurements $\mathbf{y}$ in $\mathbb{R}^M$ and outputs the signal estimate $\widehat{\mathbf{x}}$ in $\mathbb{R}^N$. To increase the dimensionality of the input from $\mathbb{R}^M$ to $\mathbb{R}^N$, we apply the adjoint operator $\mathbf{\Phi}^\intercal$ in the first layer. To preserve the dimensionality of the processing in $\mathbb{R}^N$, we dispense with the downsampling max-pooling operations made popular in modern deep convolutional networks (DCNs) [5]. We assume that the measurement matrix $\mathbf{\Phi}$ is fixed. Therefore, each $\mathbf{y}_i$ $(1 \leq i \leq M)$ is a linear combination of $\mathbf{x}_j$s $(1 \leq j \leq N)$. By training a DCN, we learn a nonlinear mapping from the signal proxy $\tilde{\mathbf{x}} = \mathbf{\Phi}^\intercal\mathbf{y}$ to the original sparse signal $\mathbf{x}$.

Among the many possibilities for the deep network architecture, we use one layer to implement the adjoint operator $\mathbf{\Phi}^\intercal$ and five convolutional layers with their corresponding batch normalization [6] layers. Each convolutional layer applies a leaky-ReLU [7] nonlinearity to its output. The i-th entry of the t-th feature map in the first convolutional layer receives the signal proxy $\tilde{\mathbf{x}}$ as its input and outputs $(\mathbf{x}_{c_1})_i^t = \mathcal{S}(\text{L-ReLU}((\mathbf{W_1^t} * \tilde{\mathbf{x}})_i + (\mathbf{b_1^t})_i))$, where $\mathbf{W_1^k} \in \mathbb{R}^P$ and $\mathbf{b_1^k} \in \mathbb{R}^{N+P-1}$ denote the filter and bias values corresponding to the t-th feature map of the first layer and L-ReLU$(x) = x$ if $x > 0$ and $= 0.01x$ if $x \leq 0$. Finally, the subsampling operator $\mathcal{S}(\cdot)$ takes the output of L-ReLU$(\cdot)$ to the original signal size by ignoring the borders created by zero-padding the input. The feature maps for the other convolutional layers are processed in a similar manner. If we denote the set of weights and biases in the DCN by Ω, then we can define a nonlinear mapping from the measurements to the original signal by $\widehat{\mathbf{x}} = \mathcal{M}(\mathbf{y}, \Omega)$. To learn the weights and biases, we employ backpropagation algorithm to minimize the mean-squared error (MSE) of the estimate $\widehat{\mathbf{x}}$.

III. Experiments

We now compare the performance of DeepInverse to the LASSO [8] ℓ_1 solver (implemented using the coordinate descent algorithm of [9]) over a grid of regularization parameters. In all the experiments, we assume that the optimal regularization parameter of LASSO is given by an oracle. Our DeepInverse network has five layers. The first and third layers have 32 filters, each having 1 and 16 channels of size 125, respectively. The second and fourth layers have 16 filters, each having 32 channels of size 125. The fifth layer has 1 filter that has 16 channels of size 125. We trained and tested DeepInverse using wavelet sparsified versions of 1D signals of size $N = 512$ extracted from random rows of CIFAR-10 images [10]. The training set contains 100,000 signals, and the test set contains 20,000 signals. The circles in Figure 2 denote the problem instances, i.e., (δ, ρ), on which we compare DeepInverse with the LASSO. By design, these problem instances are on the "failure" side of the ℓ_1 phase transition.

Table I shows the average normalized mean squared error (NMSE) and the average recovery time for the test set signals using both methods. DeepInverse outperforms LASSO (with the optimal regularization parameter) in all of the configurations determined in Figure 2. Figure 3 shows examples of signal recoveries using DeepInverse and LASSO. Finally, Figure 4 plots the MSE of DeepInverse in different training epochs.

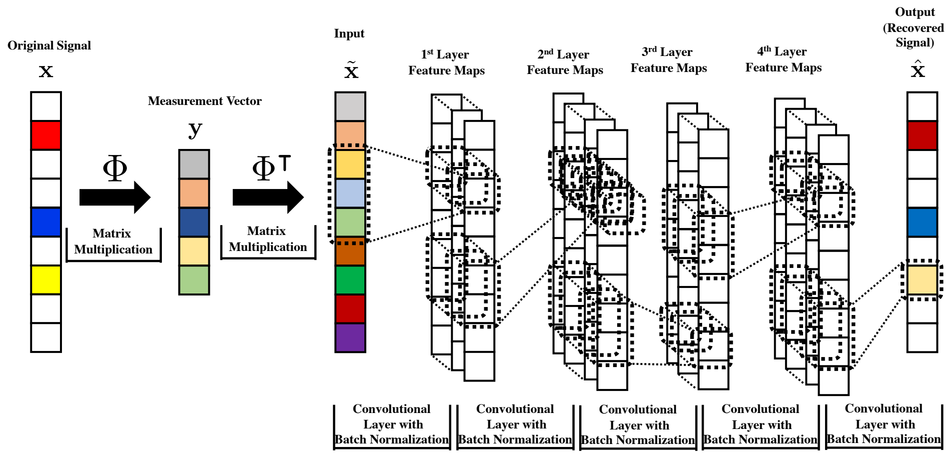

Fig. 1: *DeepInverse* learns an approximate inverse transformation from measurement vectors $\mathbf{y}$ to signals $\mathbf{x}$ using a deep convolutional network.

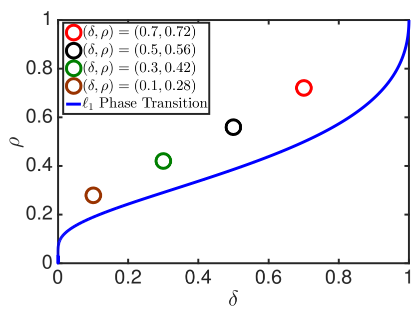

Fig. 2: ℓ_1 sparse recovery phase transition. The circles denote our test configurations, which are all on the "failure" side of the transition.

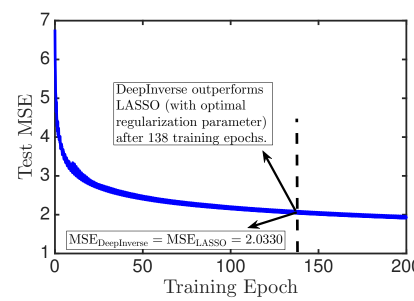

Fig. 4: Test MSE of DeepInverse during training epochs for $(\delta, \rho) = (0.7, 0.72)$. DeepInverse outperforms LASSO after only 138 epochs.

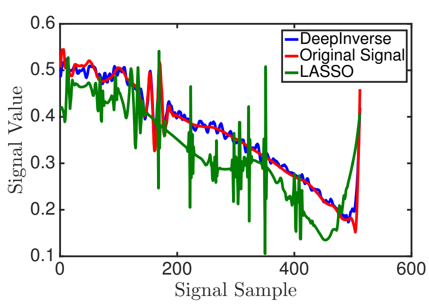

Fig. 3: Example signals recovered by DeepInverse and LASSO (with optimal regularization parameter).

TABLE I: Average NMSE and recovery time (in ms) of test set signals. DeepInverse outperforms LASSO in all cases.

(δ, ρ)	NMSE		Time (ms)	
	DeepInverse	LASSO	DeepInverse	LASSO
(0.1,0.28)	**0.0094**	0.0428	**3**	27.4
(0.3,0.42)	**0.0140**	0.0466	**3**	67.5
(0.5,0.56)	**0.0112**	0.0312	**3**	45.1
(0.7,0.72)	**0.0104**	0.0164	**3**	57.2

References

[1] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Computing*, vol. 20, pp. 33–61, 1998.

[2] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inform. Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[3] D. L. Donoho and J. Tanner, "Neighborliness of randomly projected simplices in high dimensions," *Proc. Natl. Acad. Sci.*, vol. 102, no. 27, pp. 9452–9457, Jul. 2005.

[4] A. Mousavi and R. G. Baraniuk, "Learning to invert: signal recovery via deep convolutional networks," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP), to be published*, 2017.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[7] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Machine Learning*, 2013, vol. 30.

[8] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *J. Roy. Stat. Soc., Series A*, vol. 58, no. 1, pp. 267–288, 1996.

[9] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *J. STAT. SOFTW.*, vol. 33, no. 1, pp. 1, 2010.

[10] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *tech. rep.*, 2009.