LEARNING TO INVERT: SIGNAL RECOVERY VIA DEEP CONVOLUTIONAL NETWORKS

Ali Mousavi and Richard G. Baraniuk

Department of Electrical and Computer Engineering
Rice University
Houston, TX 77005

ABSTRACT

The promise of compressive sensing (CS) has been offset by two significant challenges. First, real-world data is not exactly sparse in a fixed basis. Second, current high-performance recovery algorithms are slow to converge, which limits CS to either non-real-time applications or scenarios where massive back-end computing is available. In this paper, we attack both of these challenges head-on by developing a new signal recovery framework we call *DeepInverse* that learns the inverse transformation from measurement vectors to signals using a *deep convolutional network*. When trained on a set of representative images, the network learns both a representation for the signals (addressing challenge one) and an inverse map approximating a greedy or convex recovery algorithm (addressing challenge two). Our experiments indicate that the DeepInverse network closely approximates the solution produced by state-of-the-art CS recovery algorithms yet is hundreds of times faster in run time. The tradeoff for the ultrafast run time is a computationally intensive, off-line training procedure typical to deep networks. However, the training needs to be completed only once, which makes the approach attractive for a host of sparse recovery problems.

Index Terms— Deep Learning, Compressive Sensing, Convolutional Neural Networks

1. INTRODUCTION

An inverse problem that has many important applications is recovering $\mathbf{x} \in \mathbb{R}^N$ from a set of undersampled linear measurements $\mathbf{y} = \mathbf{\Phi x} \in \mathbb{R}^M$, where $\mathbf{\Phi}$ is an $M \times N$ *measurement matrix* and $M \ll N$. This problem is ill-posed in general and hence, in order to successfully recover the original signal $\mathbf{x}$, it should have some type of structure such that its dimensionality can be reduced without losing information.

Compressive sensing (CS) [1–3] is a special case of this problem in which the signal has a sparse representation, i.e., there exists an $N \times N$ basis matrix $\mathbf{\Psi} = [\psi_1|\psi_2|\dots|\psi_N]$ such that $\mathbf{x} = \mathbf{\Psi s}$ and only $K \ll N$ of the coefficients $\mathbf{s}$ are nonzero. Recovering the signal $\mathbf{x}$ from the measurements $\mathbf{y}$ is effected by a sparsity-regularized convex optimization or greedy algorithm [4–6].

The promise of CS has been offset by two significant challenges. The first challenge is that real-world data is not exactly sparse in a fixed basis. Some work has been pursued on learning data-dependent dictionaries to sparsify signals [7–9], but the redundancy of the resulting approaches degrades recovery performance. The second challenge is that current high-performance recovery algorithms (e.g., [10]) are slow to converge, which limits CS to either non-real-time applications or scenarios where massive back-end computing is available.

In this paper, we attack both of these challenges head-on by developing a new signal recovery framework we call *DeepInverse* that learns the inverse transformation from measurement vectors $\mathbf{y}$ to signals $\mathbf{x}$ using a *deep convolutional network*. When trained on a set of representative images, the network learns both a representation for the signals $\mathbf{x}$ (addressing challenge one) and an inverse map approximating a greedy or convex recovery algorithm (addressing challenge two).

Our experiments below indicate that the DeepInverse network closely approximates the solution produced by state-of-the-art CS recovery algorithms yet is hundreds of times faster in run time. The tradeoff for the ultrafast run time is a computationally intensive, off-line training procedure typical to deep networks. However, the training needs to be completed only once, which makes the approach attractive for a host of sparse recovery problems.

2. PRIOR WORK

The first paper to study the problem of structured signal recovery from a set of undersampled measurements using a deep learning approach was [11]. This work employed a stacked denoising autoencoder (SDA) as an unsupervised feature learner. The main drawback of the SDA approach is that its network consists of fully-connected layers, meaning that all units in two consecutive layers are connected to each other. Thus, as the signal size grows, so does the network. This imposes a large computational complexity on

Email: {ali.mousavi, richb} @rice.edu

the training (backpropagation) algorithm and can also lead to overfitting. The solution to this issue that was implemented in [11] is to divide the signal into smaller non-overlapping or overlapping blocks and then sense/reconstruct each block separately. While such an approach deals with the curse of dimensionality, a blocky measurement matrix $\mathbf{\Phi}$ is unrealistic in many applications.

The above work was followed by [12], who used a fully-connected layer along with convolutional neural networks (CNNs) to recover signals from compressive measurements. Their approach also used a blocky measurement matrix $\mathbf{\Phi}$.

In contrast to both [11] and [12], DeepInverse works with arbitrary (and not just blocky) measurement matrices $\mathbf{\Phi}$.

3. DEEP CONVOLUTIONAL NETWORKS PRIMER

Deep Convolutional Networks (DCNs) consist of three major layers: first, convolutional layer that is the core of these networks. This layer consists of a set of learnable filters with a limited receptive field that are replicated across the entire visual field and form feature maps. Second, ReLU nonlinearity layer that causes nonlinearity in decision function of the overall network. Third, pooling layer that is a form of downsampling and provides translation invariance. Backpropagation algorithm is used to train the whole network and fine tune filters of convolutional layers. All three layers play important roles in DCNs' primary application which is image classification. In other words, in DCNs one reduces dimensionality of a given image by a series of convolutional and pooling layers in order to extract the label of that image. Authors in [13] have introduced a probabilistic framework that provides insights into the success of DCNs in image classification task.

DCNs have two distinctive features that make them uniquely applicable to sparse recovery problems. First, sparse connectivity of neurons. Second, having shared weights across the entire receptive fields of one layer which increases learning speed comparing to fully-connected networks.

4. CONVOLUTIONAL NETWORKS FOR SIGNAL RECOVERY

We now develop our new DeepInverse signal recovery framework that learns the inverse transformation from measurement vectors $\mathbf{y}$ to signals $\mathbf{x}$ using a special DCN. When trained on a set of representative images, the network learns both a representation for the signals $\mathbf{x}$ and an inverse map approximating a greedy or convex recovery algorithm.

DeepInverse takes an input (set of measurements $\mathbf{y}$) in $\mathbb{R}^M$ and produces an output (signal estimate $\widehat{\mathbf{x}}$) in $\mathbb{R}^N$, where typically $M < N$. Accomplishing this dimensionality increase requires several modifications to the conventional DCN architecture. (See Fig. 1.) First, to boost the dimensionality of the input from $\mathbb{R}^M$ to $\mathbb{R}^N$, we employ a fully connected linear layer. While, in general, one could learn the weights in this layer, in this paper, we set the weights to implement the adjoint operator $\mathbf{\Phi}^\mathsf{T}$. Second, to preserve the dimensionality of the processing in $\mathbb{R}^N$, we dispense with the downsampling max-pooling operations.

As in the usual CS paradigm, 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 $(1 \leq j \leq N)$. The training set $\mathcal{D}_{\text{train}} = \{(\mathbf{y}^{(1)}, \mathbf{x}^{(1)}), (\mathbf{y}^{(2)}, \mathbf{x}^{(2)}), \ldots, (\mathbf{y}^{(l)}, \mathbf{x}^{(l)})\}$ consists of l pairs of signals and their corresponding measurements. Similarly, test set $\mathcal{D}_{\text{test}} = \{(\mathbf{y}^{(1)}, \mathbf{x}^{(1)}), (\mathbf{y}^{(2)}, \mathbf{x}^{(2)}), \ldots, (\mathbf{y}^{(s)}, \mathbf{x}^{(s)})\}$ consists of s pairs including original signals and their corresponding measurements. By training a DCN, we learn a nonlinear mapping from a signal proxy $\tilde{\mathbf{x}}$ to its original signal $\mathbf{x}$.

In the experiments described below, we use one fixed fully connected layer (to implement $\mathbf{\Phi}^\mathsf{T}$) and three convolutional layers. As in a DCN, each convolutional layer applies a ReLU nonlinearity to its output.

A few details are in order. We denote signal proxy by $\tilde{\mathbf{x}}$ where $\tilde{\mathbf{x}} = \mathbf{\Phi}^\mathsf{T}\mathbf{y}$ We assume that $\tilde{\mathbf{x}}$ is $n_1 \times n_2$ (where $n_1 \times n_2 = N$). Then the (i,j)-th entry of the k-th feature map in the first convolutional layer receives $\tilde{\mathbf{x}}$ as its input; its output is given by

$$(\mathbf{x}_{c_1})^k_{i,j} = \mathcal{S}(\text{ReLU}((\mathbf{W}^{\mathbf{k}}_{\mathbf{1}} * \tilde{\mathbf{x}})_{i,j} + (\mathbf{b}^{\mathbf{k}}_{\mathbf{1}})_{i,j})), \qquad (1)$$

where $\mathbf{W}^{\mathbf{k}}_{\mathbf{1}} \in \mathbb{R}^{k_1 \times k_2}$ and $\mathbf{b}^{\mathbf{k}}_{\mathbf{1}} \in \mathbb{R}^{n_1+k_1-1 \times n_2+k_2-1}$ denote the filter and bias values corresponding to the k-th feature map of the first layer and $\text{ReLU}(x) = \max(0, x)$. Finally, the subsampling operator $\mathcal{S}(\cdot)$ takes the output of $\text{ReLU}(\cdot)$ to the original signal size by ignoring the borders created by zero-padding the input.

The feature maps of the second and third convolutional layers are developed in a similar manner. While the filter shapes and and biases could be different in the second and third layers of the network, the principles in these layers are the same as first layer. Let ℓ_1, ℓ_2, and ℓ_3 denote the number of filters in the first, second, and third layers, respectively. If we denote the output of this convolutional network by $\hat{\mathbf{x}}$ and its set of parameters by $\Omega = \left\{\{\mathbf{W}^{\mathbf{k}}_{\mathbf{1}}, \mathbf{b}^{\mathbf{k}}_{\mathbf{1}}\}^{\ell_1}_{k=1}, \{\mathbf{W}^{\mathbf{k}}_{\mathbf{2}}, \mathbf{b}^{\mathbf{k}}_{\mathbf{2}}\}^{\ell_2}_{k=1}, \{\mathbf{W}^{\mathbf{k}}_{\mathbf{3}}, \mathbf{b}^{\mathbf{k}}_{\mathbf{3}}\}^{\ell_3}_{k=1}\right\}$, then we can define a nonlinear mapping from the measurements to original signal as $\hat{\mathbf{x}} = \mathcal{M}(\mathbf{y}, \Omega)$.

Using the mean squared error (MSE) as a loss function over the training data $\mathcal{D}_{\text{train}}$ defined as $\mathcal{L}(\Omega) = \frac{1}{l}\sum_{i=1}^{l} \|\mathcal{M}(\mathbf{y}^{(i)}, \Omega) - \mathbf{x}^{(i)}\|_2^2$, we can employ backpropagation [14] in order to minimize $\mathcal{L}(\Omega)$ and learn the parameters.

5. EXPERIMENTAL RESULTS

In this section we describe the implementation of DeepInverse and compare its performance to several other state-of-the-art CS recovery algorithms.

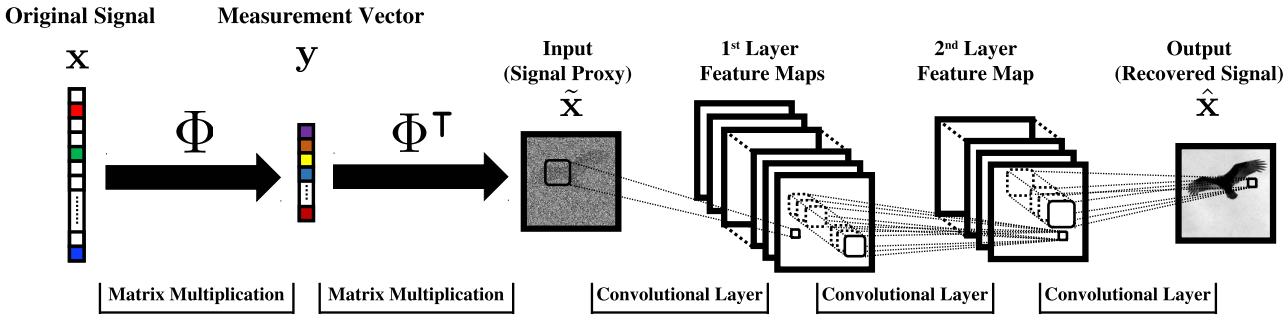


Fig. 1: *DeepInverse* learns the inverse transformation from measurement vectors $\mathbf{y}$ to signals $\mathbf{x}$ using a special deep convolutional network.

As we mentioned earlier, one of the main goals of this paper is to show that we can use deep learning framework to recover images from undersampled measurements without any need to divide images into small blocks and recover each block separately. For this purpose, we use DeepInverse that receives a signal proxy, i.e., $\tilde{\mathbf{x}} = \mathbf{\Phi}^{\intercal}\mathbf{y}$ (with same dimension as $\mathbf{x}$) as its input. In addition, it has 3 layers with the following specifications. The first layer has 64 filters, each having 1 channel of size 11×11. The second layer has 32 filters, each having 64 channels of size 11×11. The third layer has 1 filter with 32 channels of size 11×11. We trained DeepInverse using 64×64 cropped subimages of the natural images in the *ImageNet* dataset [15]. Test images were drawn from ImageNet images that were not used for training purposes.

Figure 2 shows the plot of average probability of successful recovery for different undersampling ratios (M/N) and three different recovery algorithms: D-AMP [10], total variation (TV) minimization [16], and P-AMP [17]. Note that we do not include any results from [11, 12] in our simulation results, since these approaches are specifically designed for block-based recovery whereas in this paper we focus on recovering signals without subdivision.

Figure 2 compares the probability of successful recovery as measured by 2000 Monte Carlo samples. For each undersampling ratio and Monte Carlo sample, we define the success variable $\phi_{\delta,j} = \mathbb{I}\left(\frac{\|\hat{\mathbf{x}}^{(j)} - \mathbf{x}^{(j)}\|_2^2}{\|\mathbf{x}^{(j)}\|_2^2} \leq 0.1\right)$. For small values of undersampling ratio (e.g., 0.01) DeepInverse has better performance than state-of-the-art recovery methods. However, as the undersampling ratio increases, D-AMP outperforms DeepInverse. Although Figure 2 shows that for every undersampling ratio one method works better than others, there is not a clear winner in terms of reconstruction quality.

Figure 3 compares the average PSNR [1] of the Monte Carlo test samples for different undersampling ratios and algorithms. Figure 4 shows the histograms of the PSNRs of the recovered test images, indicating the DeepInverse outperforms D-AMP for some images in the test set.

[1] $\text{PSNR} = 10.\log_{10}\left(\frac{\max^2_{\text{Image}}}{\text{MSE}}\right)$

While Figs. 2 and 3 indicate that DeepInverse offers recovery probability and PSNR performance that is comparable to state-of-the-art CS recovery algorithms, Table 1 shows that DeepInverse has a run time that is a tiny fraction of current algorithms. This fact makes DeepInverse especially suitable for applications that need low-latency recovery.

Table 2 plots the images recovered by DeepInverse and D-AMP when they are on their best and worst behavior.

Table 3 shows the effect of adding input noise on recovery performance of D-AMP and DeepInverse. We can see that for undersampling ratio of 0.1 and 20 dB input noise, DeepInverse is more robust to noise comparing to D-AMP.

Finally, Figure 5 shows the convergence of the backpropagation training algorithm over different iterations for DeepInverse. It also shows the average PSNR of the images in the test dataset for different methods with $M/N = 0.1$. We can see that after several iterations DeepInverse starts to outperform TV minimization and P-AMP.

Although D-AMP has better performance than a 3-layer DeepInverse in general, we should consider two points. First, by training an DeepInverse with more layers the network will have a larger capacity and hence, we expect it to offer better recovery performance. We leave studying DeepInverses with larger capacities as a topic of our future work. Second, DeepInverse is specially useful for applications that need low-latency recovery and at the same time we are not able to divide images into smaller blocks, i.e., we need to apply a sensing matrix to the entire signal rather than smaller blocks of it.

Table 1: Average reconstruction time of test set images for different sampling rates and algorithms.

$\frac{M}{N}$	Reconstruction Time (s)			
	DeepInverse	D-AMP	TV	P-AMP
0.2	**0.01**	3.41	2.53	1.53
0.1	**0.01**	2.93	2.34	1.23
0.01	**0.01**	2.56	2.26	0.94

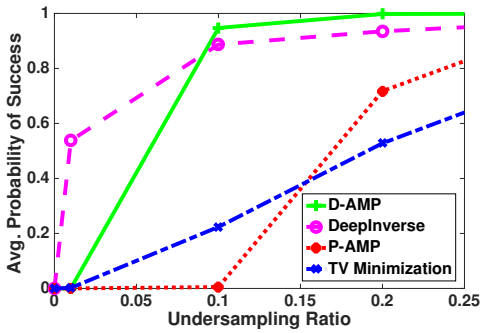


Fig. 2: Average probability of successful signal recovery for different sampling rates and algorithms.

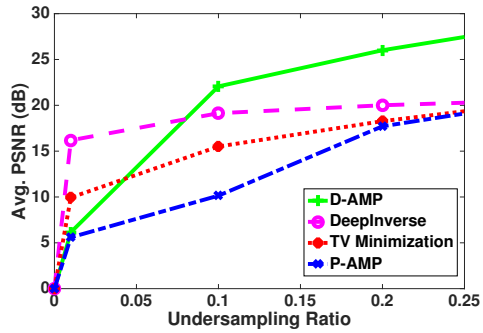


Fig. 3: Average PSNR (dB) for different sampling rates and algorithms.

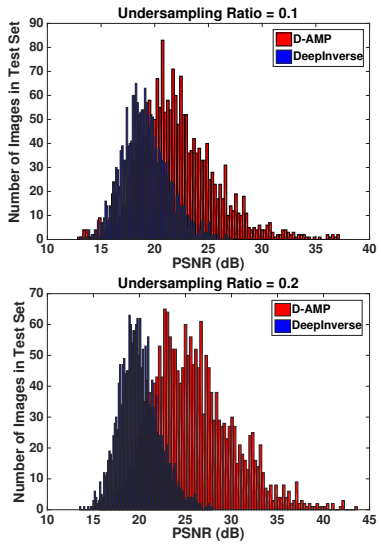


Fig. 4: Histograms of the PSNRs of the test images for two different sampling rates as recovered by D-AMP and DeepInverse.

Table 2: Quality of reconstruction (PSNR in dB) for best (denoted by ↑) and worst (denoted by ↓) reconstructed images by DeepInverse (DI) and D-AMP when undersampling ratio is 0.1.

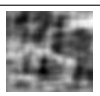
Condition	DAMP ↑	DAMP ↓	DI ↑	DI ↓
Image				
DeepInverse	 26.54	 **13.55**	 27.23	 13.10
DAMP	 **37.18**	 13.19	 **33.60**	 **13.40**

Table 3: Effect of noise on average PSNR (dB) of reconstructed test images for D-AMP and DeepInverse (undersampling ratio = 0.1). We added 20 dB noise to images of test set. Due to noise-folding [18] the variance of the noise that we observe after the reconstruction is larger than the input noise.

	Noiseless Measurements	Noisy Measurements
DAMP	22.06	21.14
DeepInverse	19.14	18.70

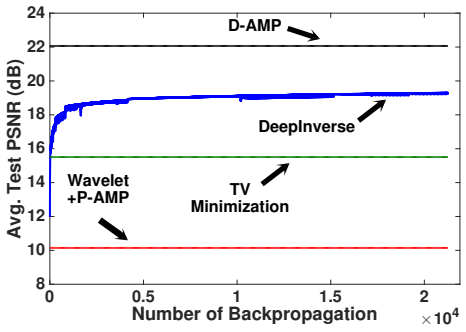


Fig. 5: Convergence of backpropagation training over different iterations for DeepInverse for the undersampling ratio $M/N = 0.1$.

6. CONCLUSIONS

In this paper, we have developed the DeepInverse framework for sensing and recovering signals. We have shown that this framework can learn a structured representation from training data and efficiently approximate a signal recovery at a small fraction of the cost of state-of-the-art recovery algorithms.

7. REFERENCES

[1] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] R. G. Baraniuk, "Compressive sensing," *IEEE Signal Processing Mag.*, vol. 24, no. 4, 2007.

[3] E. J. Candès, "Compressive sampling," in *Proceedings of the International Congress of Mathematicians*. Madrid, Spain, 2006, vol. 3, pp. 1433–1452.

[4] E. J. Candès and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. Inform. Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[5] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proc. Natl. Acad. Sci.*, vol. 106, no. 45, pp. 18914–18919, 2009.

[6] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 301–321, 2009.

[7] S. Mallat, *A wavelet tour of signal processing*, Academic Press, 1999.

[8] J. M. Duarte-Carvajalino and G. Sapiro, "Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization," Tech. Rep., DTIC Document, 2008.

[9] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[10] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," *IEEE Trans. Inform. Theory*, vol. 62, no. 9, pp. 5117–5144, 2016.

[11] A. Mousavi, A. B. Patel, and R. G. Baraniuk, "A deep learning approach to structured signal recovery," in *Proc. Allerton Conf. Communication, Control, and Computing*. IEEE, 2015, pp. 1336–1343.

[12] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, "Reconnet: Non-iterative reconstruction of images from compressively sensed random measurements," *Proc. IEEE Int. Conf. Comp. Vision, and Pattern Recognition*, 2016.

[13] A. B. Patel, T. Nguyen, and R. G. Baraniuk, "A probabilistic framework for deep learning," *Adv. in Neural Information Processing Systems (NIPS)*, 2016.

[14] D. E. Rumelhart, G. E Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive Modeling*, vol. 5, pp. 3, 1988.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "Imagenet large scale visual recognition challenge," *Int. J. Computer Vision*, pp. 1–42, 2014.

[16] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[17] A. Mousavi, A. Maleki, and R. G. Baraniuk, "Consistent parameter estimation for Lasso and approximate message passing," *arXiv preprint arXiv:1511.01017*, 2015.

[18] M. A. Davenport, J. N Laska, J. R Treichler, and R. G. Baraniuk, "The pros and cons of compressive sensing for wideband signal acquisition: Noise folding versus dynamic range," *IEEE Trans. Signal Processing*, vol. 60, no. 9, pp. 4628–4642, 2012.