A Learning Approach to Compressed Sensing

Phan Minh Nguyen
Stanford University

Abstract

Can we solve inferential problems using a learning approach? We explore the use of learning systems, as black boxes, in solving the compressed sensing problem. In particular, we derive a regression problem from the compressed sensing problem, and investigate how the convolutional neural networks and the variational auto-encoders perform under various settings.

1. Introduction

Can we solve inferential problems using a learning approach? We investigate this question with the specific instance of compressed sensing.

Compressed sensing is a linear inverse problem where the setting is underdetermined. More specifically, we are given an observation $y \in \mathbb{R}^m$ and a matrix $A \in \mathbb{R}^{m \times n}$ such that $y = Ax_0$, for some $x_0 \in \mathbb{R}^n$. Here $m < n$ (hence underdetermined setting), and we are asked to recover x_0 given the knowledge of y and A. Of course, if we know nothing more about x_0, there is no way to guarantee that we can recover exactly x_0, since $m < n$. The way is to assume that x_0 belongs to a class of signals, i.e. a subset of $\mathbb{R}^n$ of some special structure. It is a common assumption that x_0 is sparse.

In image processing, it is well-known that an image is approximately sparse when represented in an appropriate basis, e.g. some wavelet basis. Methods have been developed with guarantees for exact recovery of x_0 when the assumption of sparseness is correct. In general, by making assumptions on the structure of x_0 (i.e. restricting attention to a *known a-priori* model of x_0), inferential methods can be developed with certain optimality. Yet there are several issues. Firstly, the assumptions are usually highly idealized, and x_0 is never exactly sparse. More importantly, when dealing with specific objects like natural images, there can be more (or perhaps less) structural properties than (approximate) sparseness, and yet it is not a trivial task to model these structures. A natural question is, is there a way to go around developing model-based methods?

We explore a learning approach to this problem. More specifically, we ask the question: what if we can learn the structure? More broadly, what if we can learn to recover x_0, without explicitly finding the structure?

2. Literature Review

Compressed sensing emerged from the two papers [7, 4] in 2006, where it was shown that by solving a simple convex program

$$\min_{Ax=y} \|x\|_1 \tag{1}$$

it is possible to recover x_0, provided that x_0 is sufficiently sparse. The literature has since grown quickly. Notably the work [8] introduced an iterative algorithm, called the approximate message passing (AMP), to solve the compressed sensing problem. The work [16] modified the AMP to adapt to various (assumed) structures beyond sparseness, leveraging on the extensive literature of signal denoising and image processing. We emphasize that all these methods are inferential.

Recent progresses in machine learning propel interests in applying learning viewpoints to this problem. Three-layer stack denoising auto-encoders and convolutional neural networks (CNN) were explored in [19] and [18] respectively. These take a regression-based learning approach: given a training set of x_0's, one can generate training y's based on the model $y = Ax_0$ and train a learning system to output an approximation of x_0 from y. Notably the idea is not new and has appeared in e.g. the context of error-correction coding.

The third approach hybrids these two viewpoints [14, 17, 5]. These works built a learning system by retaining the overall structure of a well-understood inferential algorithm, while having its specific features (e.g. parameters or denoising functions) learned. Again this idea is not new (see e.g. [12]).

3. Approach

We investigate the use of CNN and variational auto-encoders (VAE) [15, 6], taking the regression-based learning approach. The choice of CNN follows [18], and is an intuitive choice since CNN accounts for spatial structures.

Figure 1. An image from CIFAR-10 Data: original, gray-scaled, and Haar-transformed.

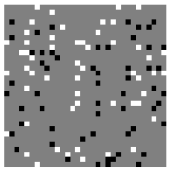

Figure 2. Image representation of a typical x_0 from Artificial Data.

We take note of the differences. Firstly, we use a deeper architecture (4 layers as opposed to 2 layers in [18]). Secondly, we also explore several issues not addressed in [18]. The choice of VAE, to the best of our knowledge, is not explored in the literature, and is also an intuitive choice, since the learning problem here is very similar to one that auto-encoders address.

The overall framework is as follows. For the x_0 data, we either extract it from CIFAR-10 images (CIFAR-10 Data), or generate it artificially (Artificial Data).

- **CIFAR-10 Data:** For each CIFAR-10 image u, we first transform it into gray-scaled image $\tilde{u}$ for simplicity, then further transform it into $x_0 = \Phi\tilde{u}$, where Φ is the Haar transform of level 3, so as to obtain an approximately sparse $x_0 \in \mathbb{R}^n$. Note that Φ is orthogonal and so preserves all information about $\tilde{u}$. An example image is shown in Fig. 1.

- **Artificial Data:** We generate $x_0 \in \mathbb{R}^n$ with entries independent and identically distributed (i.i.d.), with $\mathbb{P}(x_{0,i} = 0) = 0.9$ and $\mathbb{P}(x_{0,i} = -1) = \mathbb{P}(x_{0,i} = 1) = 0.05$. This generates very sparse x_0. Here $n = 32 \times 32 = 1024$, matching with the CIFAR-10 image dimension. Note that this is the type of x_0 data that the AMP and other inferential methods are shown to work well with. A typical x_0 in this data set is shown in Fig. 2.

We generate a matrix $A \in \mathbb{R}^{m\times n}$ with entries i.i.d. as $\mathcal{N}(0, 1/m)$. This is a common choice in the compressed sensing literature. We keep A fixed for all training and testing procedures. For each x_0, we obtain $y = Ax_0$. For different undersampling ratio, m varies.

The learning system takes y as input and outputs $\hat{x}$, an approximate of x_0. Here we impose that the first layer to be $A^T y$, which is then passed to a neural network. In doing so, we keep the dimension of the input fed into the neural net to be n fixed, instead of varying m, hence simplifying the design. Furthermore, it is very likely that A has full row-rank, hence $A^T y$ preserves complete information about y.

For CNN, we employ the following architecture:

$$\begin{aligned}\text{CNN}: y &\to A^T y \\ &\to \Big\{\text{CONV}(5,16) \to \text{ReLU} \to \text{CONV}(5,16) \\ &\to \text{BN} \to \text{ReLU} \to \text{POOL}\Big\} \times 2 \\ &\to \text{FULL} \to \hat{x}\,(y)\end{aligned}$$

where $\text{CONV}\,(5,16)$ denotes a convolutional layer with 16 filters of size 5×5, BN denotes the batch normalization layer, POOL denotes a max pooling layer with stride 2 for each dimension, and FULL denotes a fully connected layer. For VAE, we employ the following architecture:

$$\begin{aligned}\text{VAE}: y &\to A^T y \\ &\to \Big\{\text{FULL}_{400} \to \text{BN} \to \text{ReLU}\Big\} \times 2 \\ &\to \text{Z} \to \Big\{\text{FULL}_{400} \to \text{BN} \to \text{ReLU}\Big\} \times 2 \\ &\to \text{FULL} \to \hat{x}\,(y)\end{aligned}$$

where FULL_{400} denotes a fully connected layer with hidden dimension 400, and Z denotes the latent layer with latent dimension 100. We note a finding that the batch normalization layer is key to faster convergence, which is important for a regression problem that may take more epochs than a typical classification problem to obtain a reasonable performance. We also note that we do not use convolutional layers in the architecture of VAE, since it plays a complementary role to CNN.

For N training points $\left\{x_0^{(i)}, y^{(i)}\right\}_{i=1}^N$, we consider the following loss function for training:

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{x}\left(y^{(i)}\right) - x_0^{(i)}\right\|_2^2. \tag{2}$$

For most experiments, we use $N = 2000$ for the interest of time. The validate and test sets are chosen to have size 1000 each. To measure the quality of recovery, we use

$$\frac{1}{N}\sum_{i=1}^{N}\mathbf{1}\left\{\left\|\hat{x}\left(y^{(i)}\right) - x_0^{(i)}\right\|_2^2 \le 0.05\left\|x_0^{(i)}\right\|_2^2\right\} \tag{3}$$

which is referred to as the accuracy. We only use the test accuracy, not the training accuracy or validation accuracy, to measure the performance.

In all experiments, we train CNN for 100 epochs, and VAE for 300 epochs. We also apply ℓ_2 regularization.

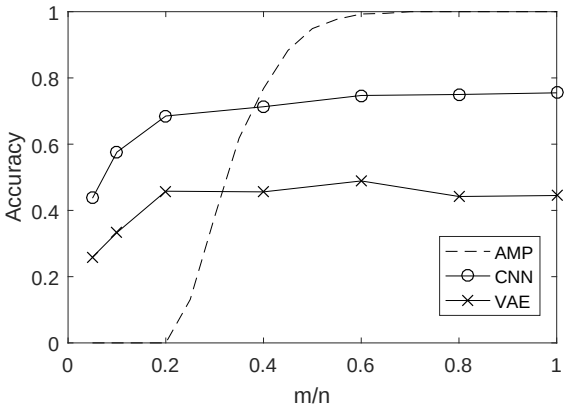

Figure 3. Performance on CIFAR-10 Data.

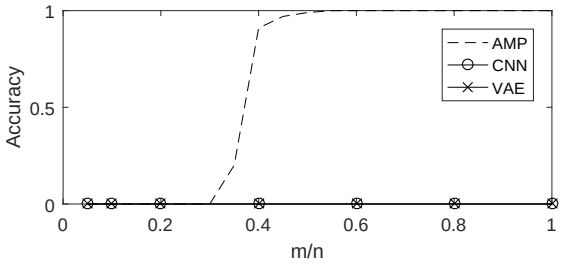

Figure 4. Performance on Artificial Data.

4. Experiments and Discussions

We perform several experiments in an exploratory manner. Each is described in a section below. In some experiments, we also present the performance of the original AMP [8] for comparison, although we note that there are variants of the AMP that may work better.

4.1. Experiment 1: CIFAR-10 Data

We experiment with the CIFAR-10 Data. The result, plotted against the underdetermined ratio m/n, is shown in Fig. 3. It is quite disappointing that the learning systems do not seem to attain accuracy close to 1 as $m/n \to 1$, unlike the AMP which does so already at $m/n \approx 0.7$. This may be due to inadequate hyper-parameter tuning, insufficient number of training epochs, or insufficient training data size. We note that the training accuracy of both CNN and VAE also does not reach anywhere close to 100% accuracy, and so while more careful regularization may yield an improvement, we do not expect to see near-perfect accuracy with better regularization.

The fact that the performance quickly saturates at some accuracy as m/n increases raises the question whether there is a fundamental issue with the learning approach, aside from the aforementioned engineering factors. Let us consider two hypotheses.

- **Hypothesis 1:** Intuitively when $m/n = 1$, A is a full-rank square matrix, and so y should contain all information about x_0. However even in that case, the performance is not great. The hypothesis is that the learning systems might have to spend enormous efforts to infer the matrix A, since we do not feed the knowledge of A into the neural networks. Note that the size of A is mn, and so even for $m = 0.05n$, this accounts for 52×10^3 unknowns!

- **Hypothesis 2:** We observe that, curiously, CNN and VAE still get non-trivial accuracy when m/n is low, beating the AMP. We hypothesize that our learning systems are able to infer some structure about the CIFAR-10 images and use that knowledge to reconstruct test images. This is a feature distinct from inferential methods like the AMP. However learning might not offer optimality that inferential methods can attain, and so perfect performance (i.e. close to 100% accuracy performance) is not guaranteed.

At this point, we have an odd situation. On one hand, one may argue that having to infer A is the key obstacle. Note that if we know A exactly, in the case $m/n = 1$, one can solve the linear system $y = Az$ for z and deduce the solution $z = x_0$. This is a simple inferential procedure. On the other hand, we see that the learning systems do not seem to mimic an inferential procedure. Which is true?

In the rest of the report, we will see that empirical evidence supports Hypothesis 2. In Section 4.6, we will argue that Hypothesis 1 can be false.

As another observation, CNN works better than VAE. This is unsurprising, given that x_0 is derived from images.

4.2. Experiment 2: Artificial Data

We experiment with the Artificial Data. As mentioned in Section 3, this is the regime where the AMP works very well, so a good performance of the learning systems here would imply that we can do inference by learning. The result is shown in Fig. 4. Surprisingly, as a stark contrast to the AMP, both CNN and VAE achieve zero accuracy for any m/n! In fact, test accuracy, training accuracy and validation accuracy are all zero. What's wrong? Given that CNN and VAE are able to learn with the CIFAR-10 Data, the cause should lie in the structure of x_0 in the Artificial Data.

First, note that the sparsity level $\mathbb{P}(x_{0,i} = 0) = 0.9$ is high. Intuitively, the sparser x_0 is, the less information we need to infer. Yet the learning systems could not extract anything at such high sparsity level, which the AMP does not seem to have a trouble with. We thus hypothesize that the cause is in that the entries in x_0 are generated independently.

Delving deeper into this hypothesis, we recall that the neural networks are universal approximators, i.e. they can approximate any reasonable function. The AMP, or inferential methods such as (1), admit very simple descriptions. That is, there exists a function with a simple procedural description that works well. Then why can't our learning systems learn, at the very least, to approximate this function? In other words, what do they have troubles learning? Note that it is hypothesized in Section 4.1 that the systems have to learn to infer A. However, if this hypothesis is true, then the systems also have to infer A with the CIFAR-10 Data, yet still achieve a reasonable accuracy.

We conclude that the independence structure in the x_0 data set is what renders CNN and VAE unable to learn. This is complementary to the conclusion in Section 4.1: our learning systems are in favor of data sets with certain structures (which, in this case, are image-derived), and in a disadvantage when learning from data sets with other structures.

We do another experiment to further validate this conclusion. We train on the CIFAR-10 Data, but test on the Artificial Data. As expected, the training accuracy is reasonably above zero, but the test accuracy is zero. We see that the learning systems do not try to approximate an inferential procedure, but rather learn the structures in the data.

4.3. Experiment 3: CIFAR-10 Data with Different Classes

The discussion in Section 4.2, which concludes that the nature of a data set strongly affects the performance of a learning system, leads us to another question. What would happen if we work on CIFAR-10 Data, but the training images and the test images belong to classes of different natures? We perform training on images drawn from the classes "plane", "car", "ship", "truck" (vehicle class), and testing on images from "bird", "cat", "deer", "dog", "flog", "horse" (animal class).

The result is shown in Fig. 5. We see that CNN and VAE still perform reasonably well, but suffer a noticeable loss. That is, the neural networks are able to generalize, but insufficient diversity in the training data degrades the performance.

This is not surprising in light of the discussion in Section 4.2, as well as the intuition that images from the animal class are not of drastically different nature from images from the vehicle class.

4.4. Experiment 4: CIFAR-10 Data without Haar transform

One question is whether it offers any advantage if we set $x_0 = \tilde{u}$ (gray-scaled image), instead of $\Phi\tilde{u}$ (Haar transform of the image), i.e. we do not perform Haar transformation. Of course, a natural image should be of different structure

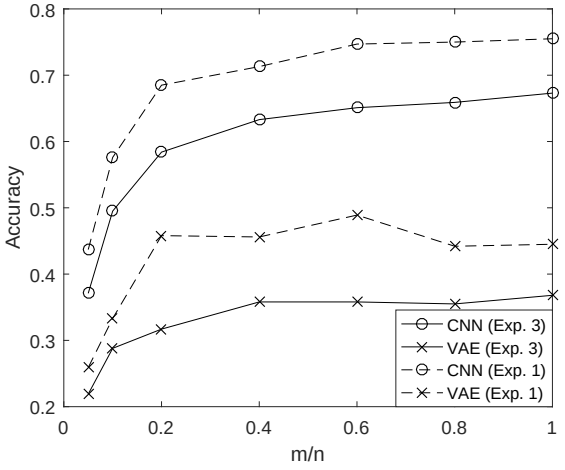

Figure 5. Performance on CIFAR-10 Data, with training and testing on different classes. Exp. 3 refers to the experiment in Section 4.3, and Exp. 1 refers to Section 4.1.

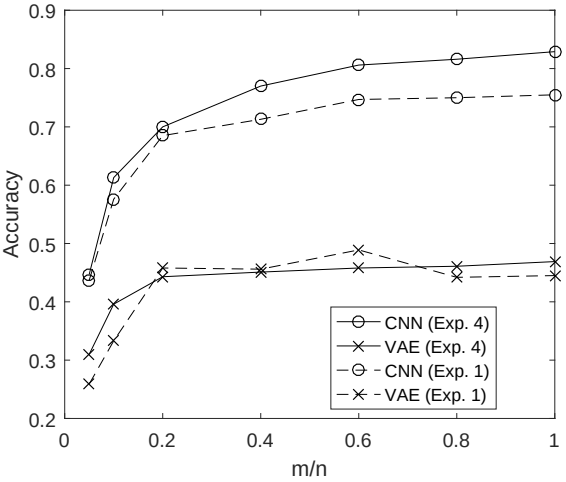

Figure 6. Performance on CIFAR-10 Data, without Haar transform. Exp. 4 refers to the experiment in Section 4.4, and Exp. 1 refers to Section 4.1.

from its Haar transform, and we expect to see a change in performance.

The result is shown in Fig. 6. Removing Haar transformation indeed improves the performance of CNN. This is not surprising, since the CNNs are known to be well suited for visual data. But curiously the performance of VAE neither improves nor degrades.

In fact, there is a simple reason behind this observation, argued as follows. We note that Φ is an orthogonal transformation. Notice that x_0 enters the system in only two ways: firstly, it is present in the ℓ_2 loss $\|\hat{x} - x_0\|_2^2$ that appears in both the loss function (2) and the accuracy measurement (3), and secondly, it appears in the input to the neural net-

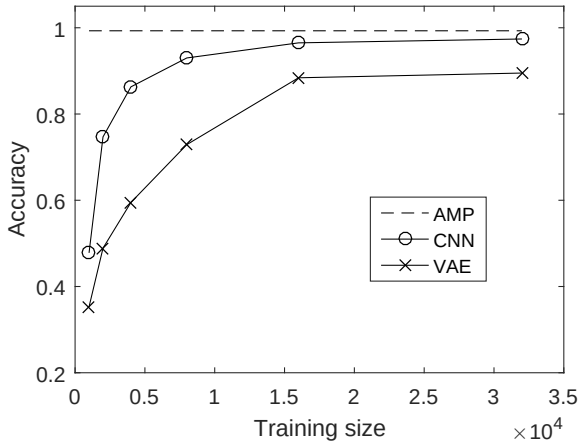

Figure 7. Performance on CIFAR-10 Data, plotted against the training size, at $m/n = 0.6$.

work.

To see that Φ does not affect $\|\hat{x} - x_0\|_2^2$, note that

$$\|\hat{x} - \tilde{u}\|_2^2 = \|\Phi\hat{x} - \Phi\tilde{u}\|_2^2$$

i.e. for the system without Haar transform, it would remain the same as one with Haar transform if we multiply the weight of the last fully connected layer with Φ. This is possible since we do not impose any structure on the weight matrix of the fully connected layer.

To see that Φ does not affect the input to the neural network, recall that this input is $A^T A\Phi\tilde{u}$ when using Haar transform, and $A^T A\tilde{u}$ when not using it. Let $B = A\Phi$ and hence $A^T A\Phi\tilde{u} = \Phi B^T B\tilde{u}$. Recall that A has entries i.i.d. Gaussian, and therefore rotationally invariant. That is, B has the same distribution as A. Therefore, $B^T B\tilde{u}$ has the same distribution as $A^T A\tilde{u}$. Finally, to see that Φ in $\Phi B^T B\tilde{u}$ does not affect the result, notice that in VAE, this input to the neural network is immediately passed through a fully connected layer. This completes our argument.

4.5. Experiment 5: More CIFAR-10 Training Data

We ask the question whether we can address the problem of unsatisfactory performance with more training data. The result is shown in Fig. 7, where we experiment at the underdetermined ratio $m/n = 0.6$. Indeed, increasing the training size from 2000 significantly improves the performance. Yet the gained advantage diminishes when the training size becomes very large. There is still an observable gap to the AMP performance.

4.6. Experiment 6: The role of A

We depart from the underdetermined setting, set $m = n$, and ask whether the presence of A has any influence on the performance. When the system is without A, one can simply think of that as $A = I$ the identity matrix. We experiment this with the settings in Sections 4.1, 4.3 and 4.4. The result is shown in Table 1.

	Exp. 1	Exp. 3	Exp. 4
CNN with A	**0.755**	**0.673**	0.829
CNN w/o A	0.628	0.572	**0.931**
VAE with A	0.445	0.368	0.469
VAE w/o A	0.448	0.374	0.472

Table 1. Performance with and without A, for $m/n = 1$. Exp. 1, 3 and 4 refers to the setting in Sections 4.1, 4.3 and 4.4 respectively.

First observe that for VAE, the presence of A does not affect the performance at all. Again, one can explain this mathematically. Note that A is almost surely full-rank. Furthermore, for two different rows (or columns) A_1 and A_2 of A, we have $\langle A_1, A_2\rangle = \sum_{i=1}^n A_{1,i}A_{2,i}$, which converges to 0 in probability as $n \to \infty$ by the law of large numbers, since A has entries i.i.d. as $\mathcal{N}(0, 1/n)$. In addition, $\|A_1\|_2^2$ also converges to 1 in probability as $n \to \infty$. Therefore, for large n, A is approximately orthogonal. Recall that in VAE, the input to the neural network $A^T Ax_0$ is passed through to a fully connected layer. Since A is almost orthogonal, it does not affect the performance.

We observe that when with A, CNN performs better in the settings of experiments 1 and 3, but worse in experiment 4. Recall that in experiment 4, we set x_0 to the gray-scaled image. As such, CNN is expected to work better without A. This is in agreement with the result in Section 4.4, where applying a transformation on the gray-scaled image would yield worse results.

The fact that CNN works better with A in experiments 1 and 3 is intriguing. We return to Hypothesis 1 in Section 4.1, which states that the learning systems have to infer A, which hinders good performance. We see that depending on the nature of the data, it is possible that having A in the system is beneficial. As such, Hypothesis 1 is likely false.

There is yet another intriguing implication. Suppose we want to build a learning system which takes x_0 as input and tries to output an approximation of it (which is essentially the task of an auto-encoder). Here the system employs the convolutional architecture. Then depending on the nature of x_0, it is possible to improve the performance by adding an extra layer before the neural network. This layer simply multiplies x_0 with $A^T A$, where $A \in \mathbb{R}^{n\times n}$ with entries i.i.d. as $\mathcal{N}(0, 1/n)$.

Furthermore, as seen in Fig. 3 and 5, for $A \in \mathbb{R}^{m\times n}$ with $m/n < 1$ sufficiently high, the performance loss is negligible. Hence, in the above interpretation, one can reduce computational cost by, instead of generating A a square matrix, generating A with dimension $m \times n$.

This suggests a very cheap way to gain some performance, as one builds a convolutional neural network for

auto-encoding tasks, depending on the nature of the data!

5. Concluding Remarks

Throughout several experiments, an important insight we have gained is that learning systems inherently learn the structure within data, and use that knowledge to perform inference. The performance is highly dependent on the nature of the data, in particular, whether this nature is matching with the architecture of the neural network.

What does this imply? On one hand, the learning system can adapt to the data, whose structure might be poorly understood, a situation where model-based inferential methods may not work well. On the other hand, if the nature of the data is not matching with the underlying architecture, the result can be unsatisfactory or even atrocious. So when approaching an inferential problem with learning, we trade optimality with flexibility. That is, when the data's structure is completely understood, one can try to find an optimal inferential method. Yet when there is no good inferential method available at hands for a particular set of data, one can opt for engineering a learning system using well-known architectures.

To achieve better performance with learning in the compressed sensing problem, one ought to have good data diversity, as well as abundant training samples.

We have also seen an intriguing instance (in particular, experiment 6 in Section 4.6) where one takes ideas from inferential problems to make improvements to a learning problem. This hints some connection between inferential problems, or particularly compressed sensing, and learning problems. Recent works [11, 10] have studied in this direction.

There are factors that we have not discussed here. As mentioned in [18], the deep neural network system has the advantage of speed: the test / inference time is fast, since it comprises of mainly matrix multiplications. We further comment on this point. In the case of the AMP, the computation also comprises of mainly matrix multiplications, and usually takes 30 iterations to run. So the inference time of a neural network would match the computation time of the AMP if its number of layers is (on the order of) about 30. Other variants of the AMP may take much longer, due to complex functions needed to compute. Yet we can hope that with that number of layers, the representation power of the neural network is sufficiently high to attain good performance for certain data sets.

Another factor is that the learning approach requires a lot of training data, with sufficient data diversity, whereas the inferential methods do not require any training data. Hence learning can be helpless in situations with data scarcity.

Finally, in this project, we have been using learning systems as black boxes. One drawback of the considered architectures is that they do not well incorporate knowledge of the matrix A. As mentioned, there are more principled approaches that hybrid inferential methods such as the AMP, and learning. Optimality of such approach is yet to be seen.

References

[1] M. Bayati, M. Lelarge, and A. Montanari. Universality in polytope phase transitions and iterative algorithms. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 1643–1647. IEEE, 2012.

[2] M. Bayati and A. Montanari. The dynamics of message passing on dense graphs, with applications to compressed sensing. *IEEE Transactions on Information Theory*, 57(2):764–785, 2011.

[3] M. Borgerding and P. Schniter. Onsager-corrected deep learning for sparse linear inverse problems. *arXiv preprint arXiv:1607.05966*, 2016.

[4] E. J. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52(12):5406–5425, 2006. 2

[5] J. Chang, C.-L. Li, B. Poczos, B. Kumar, and A. C. Sankaranarayanan. One network to solve them all—solving linear inverse problems using deep projection models. *arXiv preprint arXiv:1703.09912*, 2017. 2

[6] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016. 3

[7] D. L. Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006. 2

[8] D. L. Donoho, A. Maleki, and A. Montanari. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919, 2009. 2, 4

[9] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4829–4837, 2016.

[10] A. C. Gilbert, Y. Zhang, K. Lee, Y. Zhang, and H. Lee. Towards understanding the invertibility of convolutional neural networks. *arXiv preprint arXiv:1705.08664*, 2017. 5

[11] R. Giryes, G. Sapiro, and A. M. Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy. *CoRR, abs/1504.08291*, 2015. 5

[12] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 399–406, 2010. 2

[13] K. Jarrett, K. Kavukcuoglu, Y. LeCun, et al. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

[14] U. S. Kamilov and H. Mansour. Learning optimal nonlinearities for iterative thresholding algorithms. *IEEE Signal Processing Letters*, 23(5):747–751, 2016. 2

[15] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3

[16] C. A. Metzler, A. Maleki, and R. G. Baraniuk. From denoising to compressed sensing. *IEEE Transactions on Information Theory*, 62(9):5117–5144, 2016. 2

[17] C. A. Metzler, A. Mousavi, and R. G. Baraniuk. Learned d-amp: A principled cnn-based compressive image recovery algorithm. *arXiv preprint arXiv:1704.06625*, 2017. 2

[18] A. Mousavi and R. G. Baraniuk. Learning to invert: Signal recovery via deep convolutional networks. *arXiv preprint arXiv:1701.03891*, 2017. 2, 3, 5

[19] A. Mousavi, A. B. Patel, and R. G. Baraniuk. A deep learning approach to structured signal recovery. In *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*, pages 1336–1343. IEEE, 2015. 2

[20] Y. Wu and S. Verdú. Optimal phase transitions in compressed sensing. *IEEE Transactions on Information Theory*, 58(10):6241–6263, 2012.