Deep Decoder: Concise Image Representations from Untrained Non-convolutional Networks

Reinhard Heckel[*] and Paul Hand[†]

[*]Dept. of Electrical and Computer Engineering, Rice University
[†]Dept. of Mathematics and College of Computer and Information Science, Northeastern University

March 12, 2019

Abstract

Deep neural networks, in particular convolutional neural networks, have become highly effective tools for compressing images and solving inverse problems including denoising, inpainting, and reconstruction from few and noisy measurements. This success can be attributed in part to their ability to represent and generate natural images well. Contrary to classical tools such as wavelets, image-generating deep neural networks have a large number of parameters—typically a multiple of their output dimension—and need to be trained on large datasets. In this paper, we propose an untrained simple image model, called the deep decoder, which is a deep neural network that can generate natural images from very few weight parameters. The deep decoder has a simple architecture with no convolutions and fewer weight parameters than the output dimensionality. This underparameterization enables the deep decoder to compress images into a concise set of network weights, which we show is on par with wavelet-based thresholding. Further, underparameterization provides a barrier to overfitting, allowing the deep decoder to have state-of-the-art performance for denoising. The deep decoder is simple in the sense that each layer has an identical structure that consists of only one upsampling unit, pixel-wise linear combination of channels, ReLU activation, and channelwise normalization. This simplicity makes the network amenable to theoretical analysis, and it sheds light on the aspects of neural networks that enable them to form effective signal representations.

1 Introduction

Data models are central for signal and image processing and play a key role in compression and inverse problems such as denoising, super-resolution, and compressive sensing. These data models impose structural assumptions on the signal or image, which are traditionally based on expert knowledge. For example, imposing the assumption that an image can be represented with few non-zero wavelet coefficients enables modern (lossy) image compression [Ant+92] and efficient denoising [Don95].

In recent years, it has been demonstrated that for a wide range of imaging problems, from compression to denoising, deep neural networks trained on large datasets can often outperform methods based on traditional image models [Tod+16; Agu+17; The+17; Bur+12; Zha+17]. This success can largely be attributed to the ability of deep networks to represent realistic images when trained on large datasets. Examples include learned representations via autoencoders [HS06] and generative adversarial models [Goo+14]. Almost exclusively, three common features of the recent success stories of using deep neural network for imaging related tasks are that the corresponding networks are over-parameterized (i.e., they have much more parameters than the dimension of the image that they represent or generate), that the networks have a convolutional structure, and perhaps most importantly, that the networks are trained on large datasets.

An important exception that breaks with the latter feature is a recent work by Ulyanov et al. [Uly+18], which shows that a deep neural network—over-parameterized and convolutional—can solve inverse problems well without any training. Specifically, Ulyanov et al. [Uly+18] demonstrated that fitting the weights of an over-parameterized deep convolutional network to a single image, when done in combination with regularization by early stopping, can yield state-of-the-art performance for inverse problems like denoising, superresolution, and inpainting. This result is surprising as it suggests that the combination of the network structure and the early stopping regularization provides an effective model for natural signals without a large training dataset.

In this paper, we propose a simple image model in the form of a deep neural network that can generate natural images well, and thus enables image compression, denoising, and solving other inverse problems with close-to or state-of-the-art performance. We call the network a deep decoder, due to its resemblance to the decoder part of an autoencoder. The network does not require training, and contrary to previous approaches, the network itself incorporates all assumptions on the data, is under-parameterized, does not involve convolutions, and has a simplicity that makes it amenable to theoretical analysis. The key features of the approach, and key contributions of the paper are as follows:

- The network is not learned and itself incorporates all assumptions on the data. Instead of being trained with a large dataset, the parameters of the network are optimized to fit a single image. This has multiple benefits: the same network and code is usable for multiple applications; and the method is not sensitive to a potential misfit between training and test data, since there is no training. The method does not make assumptions on the class of natural images as part of additional regularization, such as by stopping optimization early.

- The network is under-parameterized in the sense that there are fewer weight parameters than the dimensionality of the output image. Thus, the network maps a lower-dimensional space to a higher-dimensional space, similar to classical image representations such as sparse wavelet representations. This feature enables image compression by storing the coefficients of the network after its weights are optimized to fit a single image. In Section 2, we demonstrate that the compression is on-par with wavelet thresholding [Ant+92], a strong baseline that underlies JPEG-2000. An additional benefit of underparameterization is that it provides a barrier to overfitting, which enables robustness to noise.

- The network does not have a convolutional structure. The majority of the networks for image compression, restoration, and recovery have a convolutional structure [Tod+16; Agu+17; The+17; Bur+12; Zha+17]. While convolutions are perhaps critical and certainly useful for a number of important image related problems, our work suggests that they are not critical for the specific problem of generating an image *without learning*.

- The network only consists of a simple combination of few building blocks, which makes it amenable to analysis and theory. For example, we prove that the deep decoder can only fit a small proportion of noise, which, combined with the empirical observation that it can represent natural images well, explains its denoising performance.

The remainder of the paper is organized as follows. In Section 2, we first demonstrate that the deep decoder enables concise image representations. We formally introduce the deep decoder in Section 3. In Section 4, we show the performance of the deep decoder on a number of inverse problems such as denoising. In Section 5 we discuss related work, and finally, in Section 6 we provide theory and explanations on what makes the deep decoder work.

2 Concise image representations with a deep image model

Intuitively, a model describes a class of signals well if it is able to represent or approximate a member of the class with few parameters. In this section, we demonstrate that the deep decoder, an untrained, non-convolutional neural network, defined in the next section, enables *concise* representation of an image—on par with state of the art wavelet thresholding.

The deep decoder is a deep image model $G\colon \mathbb{R}^N \to \mathbb{R}^n$, where N is the number of parameters of the model, and n is the output dimension, which is typically much larger than the number of parameters ($N \ll n$). The parameters of the model, which we denote by $\mathbf{C}$, are the weights of the network, and not the input of the network, which we will keep fixed. To demonstrate that the deep decoder enables concise image representations, we choose the number of parameters of the deep decoder, N, such that it is a small fraction of the output dimension of the deep decoder, i.e., the dimension of the images[1].

We draw 100 images from the ImageNet validation set uniformly at random and crop the center to obtain a 512x512 pixel color image. For each image $\mathbf{x}^*$, we fit a deep decoder model $G(\mathbf{C})$ by minimizing the loss

$$L(\mathbf{C}) = \|G(\mathbf{C}) - \mathbf{x}^*\|_2^2$$

with respect to the network parameters $\mathbf{C}$ using the Adam optimizer. We then compute for each image the corresponding peak-signal-to-noise ratio, defined as $10\log_{10}(1/\text{MSE})$, where $\text{MSE} = \frac{1}{3\cdot 512^2}\|\mathbf{x}^* - G(\mathbf{C})\|_2^2$, $G(\mathbf{C})$ is the image generated by the network, and $\mathbf{x}^*$ is the original image.

We compare the compression performance to wavelet compression [Ant+92] by representing each image with the N-largest wavelet coefficients. Wavelets—which underly JPEG 2000, a standard for image compression—are one of the best methods to approximate images with few coefficients. In Figure 1 we depict the results. It can be seen that for large compression ratios ($3 \cdot 512^2/N = 32.3$), the representation by the deep decoder is slightly better for most images (i.e., is above the red line), while for larger compression ratios ($3 \cdot 512^2/N = 8$), the wavelet representation is slightly better. This experiment shows that deep neural networks can represent natural images well with very few parameters and without any learning.

The observation that, for small compression ratios, wavelets enable more concise representations than the deep decoder is intuitive because any image can be represented exactly with sufficiently many wavelet coefficients. In contrast, there is no reason to believe a priori that the deep decoder has zero representation error because it is underparameterized.

The main point of this experiment is to demonstrate that the deep decoder is a good image model, which enables applications like solving inverse problems, as in Section 4. However, it also suggest that the deep decoder can be used for lossy image compression, by quantizing the coefficients $\mathbf{C}$ and saving the quantized coefficients. In the appendix, we show that image representations of the deep decoder are not sensitive to perturbations of its coefficients, thus quantization does not have a detrimental effect on the image quality. Deep networks were used successfully before for the compression of images [Tod+16; Agu+17; The+17]. In contrast to our work, which is capable of

[1]Specifically, we took a deep decoder G with $d = 6$ layers and output dimension $512 \times 512 \times 3$, and choose $k = 64$ and $k = 128$ for the large and small compression ratios, respectively.

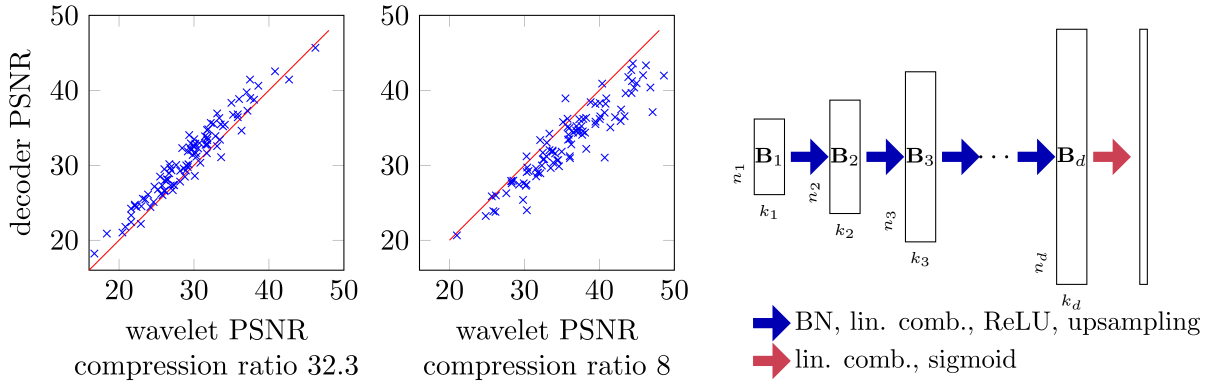

Figure 1: The deep decoder (depicted on the right) enables concise image representations, on-par with state-of-the-art wavelet based compression. The crosses on the left depict the PSNRs for 100 randomly chosen ImageNet-images represented with few wavelet coefficients and with a deep decoder with an equal number of parameters. A cross above the red line means the corresponding image has a smaller representation error when represented with the deep decoder. The deep decoder is particularly simple, as each layer has the same structure, consisting of one channelwise normalization (BN), a pixel-wise linear combination of channels, ReLU nonlinearity, and upsampling.

compressing images without any learning, the aforementioned works *learn* an encoder and decoder using convolutional recurrent neural networks [Tod+16] and convolutional autoencoders [The+17] based on training data.

3 The deep decoder

We consider a decoder architecture that transforms a randomly chosen and fixed tensor $\mathbf{B}_1 \in \mathbb{R}^{n_1 \times k_1}$ consisting of k_1 many n_1-dimensional channels to an $n_d \times k_{\text{out}}$ dimensional image, where $k_{\text{out}} = 1$ for a grayscale image, and $k_{\text{out}} = 3$ for an RGB image with three color channels. Throughout, n_i has two dimensions; for example our default configuration has $n_1 = 16 \times 16$ and $n_d = 512 \times 512$. The network transforms the tensor $\mathbf{B}_1$ to an image using batch-normalization and upsampling operations, pixel-wise linearly combining the channels, and applying rectified linear units (ReLUs). Specifically, the channels in the $(i+1)$-th layer are given by

$$\mathbf{B}_{i+1} = \mathbf{U}_i \text{relu}(\text{bn}(\mathbf{B}_i \mathbf{C}_i)), \quad i = 1, \ldots, d.$$

Here, the coefficient matrices $\mathbf{C}_i \in \mathbb{R}^{k_i \times k_{i+1}}$ contain the weights of the network. Each column of the tensor $\mathbf{B}_i \mathbf{C}_i \in \mathbb{R}^{n_i \times k_{i+1}}$ is formed by taking linear combinations of the channels of the tensor $\mathbf{B}_i$ in a way that is consistent across all pixels.

Then, $\text{bn}(\cdot)$ performs the batch norm operation [IS15], which is equivalent to normalizing each channel individually. Specifically, let $\mathbf{Z}_i = \mathbf{B}_i \mathbf{C}_i$ be the channels in the i-th layer, and let $\mathbf{z}_{ij}$ be the j-th channel in the i-th layer. Then batch normalization performs the following transformation: $\mathbf{z}'_{ij} = \frac{\mathbf{z}_{ij} - \text{mean}(\mathbf{z}_{ij})}{\sqrt{\text{var}(\mathbf{z}_{ij}) + \epsilon}} \gamma_{ij} + \beta_{ij}$, where mean and var compute the empirical mean and variance, and γ_{ij}

and β_{ij} are parameters, learned independently for each channel, and ϵ is a fixed small constant. Learning the parameters γ and β helps the optimization but is not critical.

The operator $\mathbf{U}_i \in \mathbb{R}^{n_{i+1} \times n_i}$ is an upsampling tensor, which we choose throughout so that it performs bi-linear 2x upsampling. For example, if the channels in the first layer have dimensions $n_1 = 16 \times 16$, then the upsampling operator $\mathbf{U}_1$ upsamples each channel to dimensions 32×32. In the last layer, we do not upsample, which is to say that we choose the corresponding upsampling operator as the identity. Finally, the output of the d-layer network is formed as

$$\mathbf{x} = \text{sigmoid}(\mathbf{B}_d \mathbf{C}_{d+1}),$$

where $\mathbf{C}_{d+1} \in \mathbb{R}^{k_d \times k_{\text{out}}}$. Throughout, our default architecture is a $d = 6$ layer network with $k_i = k$ for all i, and we focus on output images of dimensions $n_d = 512 \times 512$. See Figure 1 for an illustration. Recall that the parameters of the network are given by $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_d, \mathbf{C}_{d+1}\}$, and the output of the network is only a function of $\mathbf{C}$, since we choose the tensor $\mathbf{B}_1$ at random and fix it. Therefore, we write $\mathbf{x} = G(\mathbf{C})$. Note that the number of parameters is given by $\sum_{i=1}^{d}(k_i k_{i+1} + 2k_i) + k_{d+1}k_d$, where the term $2k_i$ corresponds to the two free parameters associated with the batch norm. We choose the number of output (i.e. color) channels throughout as $k_{\text{out}} = 3$ and the other parameters as $k_i = k$, for all i. Thus, the number of parameters is $dk^2 + 2dk + 3k$.

We finally note that taking pixelwise linear combinations of channels (with a consistent transformation across all pixels) is equivalent to performing 1x1 convolutions. To stress the fact that 1x1 convolutions are degenerate convolutions that do not exploit locality within images by extracting features from local image patches, we refrain from naming this operation a convolution throughout the paper. The deep decoder is *not* a convolutional neural network.

4 Solving inverse problems with the deep decoder

In this section, we use the deep decoder as a structure-enforcing model for solving standard inverse problems: denoising, super-resolution, and inpainting. In all of those inverse problems, the goal is to recover an image $\mathbf{x}$ from a noisy observation $\mathbf{y} = f(\mathbf{x}) + \eta$. Here, f is a known forward operator (possibly equal to identity), and η is structured or unstructured noise.

We recover the image $\mathbf{x}$ with the deep decoder as follows. Motivated by the finding from the previous section that a natural image $\mathbf{x}$ can (approximately) be represented with the deep decoder as $G(\mathbf{C})$, we estimate the unknown image from the noisy observation $\mathbf{y}$ by minimizing the loss

$$L(\mathbf{C}) = \|f(G(\mathbf{C})) - \mathbf{y}\|_2^2$$

with respect to the model parameters $\mathbf{C}$. Let $\hat{\mathbf{C}}$ be the result of the optimization procedure. We estimate the image as $\hat{\mathbf{x}} = G(\hat{\mathbf{C}})$.

We use the Adam optimizer for minimizing the loss, but have obtained comparable results with gradient descent. Note that this optimization problem is non-convex and we might not reach a global minimum. Throughout, we consider the least-squares loss (i.e., we take $\|\cdot\|_2$ to be the ℓ_2 norm), but the loss function can be adapted to account for structure of the noise.

We remark that fitting an image model to observations in order to solve an inverse problem is a standard approach and is not specific to the deep decoder or deep-network-based models in general. Specifically, a number of classical signal recovery approaches fit into this framework; for example solving a compressive sensing problem with ℓ_1-norm minimization amounts to choosing the forward operator as $f(\mathbf{x}) = \mathbf{A}\mathbf{x}$ and minimizing over $\mathbf{x}$ in a ℓ_1-norm ball.

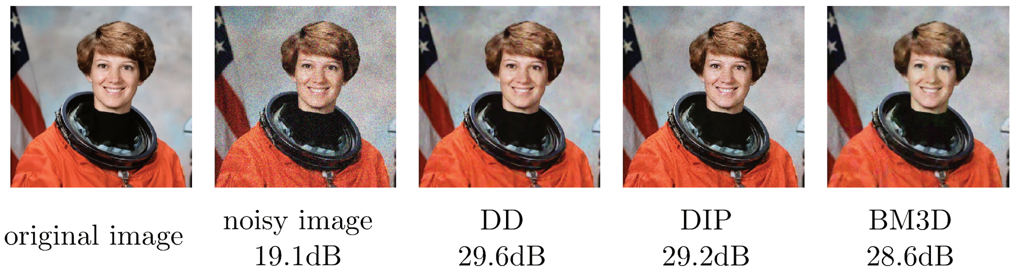

Figure 2: An application of the deep decoder for denoising the astronaut test image. The deep decoder has performance on-par with state of the art untrained denoising methods, such as the DIP method [Uly+18] and the BM3D algorithm [Dab+07].

4.1 Denoising

We start with the perhaps most basic inverse problem, denoising. The motivation to study denoising is at least threefold: First, denoising is an important problem in practice, second, many inverse problem can be solved as a chain of denoising steps [Rom+17], and third, the denoising problem is simple to model mathematically, and thus a common entry point for gaining intuition on a new method. Given a noisy observation $\mathbf{y} = \mathbf{x} + \eta$, where η is additive noise, we estimate an image with the deep decoder by minimizing the least squares loss $\|G(\mathbf{C}) - \mathbf{y}\|_2^2$, as described above.

The results in Figure 2 and Table 1 demonstrate that the deep decoder has denoising performance on-par with state of the art untrained denoising methods, such as the related Deep Image Prior (DIP) method [Uly+18] (discussed in more detail later) and the BM3D algorithm [Dab+07]. Since the deep decoder is an untrained method, we only compared to other state-of-the-art untrained methods (as opposed to learned methods such as [Zha+17]).

Why does the deep decoder denoise well? In a nutshell, from Section 2 we know that the deep decoder can represent natural images well even when highly underparametrized. In addition, as a consequence of being under-parameterized, the deep decoder can only represent a small proportion of the noise, as we show analytically in Section 6, and as demonstrated experimentally in Figure 4. Thus, the deep decoder "filters out" a significant proportion of the noise, and retains most of the signal.

How to choose the parameters of the deep decoder? The larger k, the larger the number of latent parameters and thus the smaller the representation error, i.e., the error that the deep decoder makes when representing a noise-free image. On the other hand, the smaller k, the fewer parameters, and the smaller the range space of the deep decoder $G(\mathbf{C})$, and thus the more noise the method will remove. The optimal k trades off those two errors; larger noise levels will require smaller values of k. Throughout the denoising experiments, we choose $k = 128$.

4.2 Superresolution

We next super-resolve images with the deep denoiser. We define a forward model f that performs downsampling with the Lanczos filter by a factor of four. We then downsample a given image by a factor of four, and then reconstruct it with the deep decoder (with $k = 128$, as before). We compare performance to bi-cubic interpolation and to the deep image prior, and find that the deep

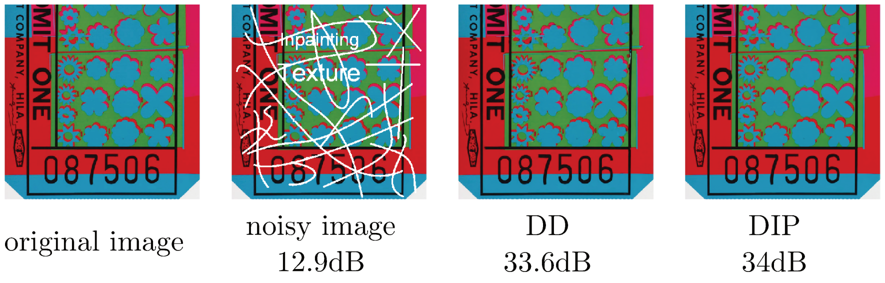

Figure 3: An application of the deep decoder for recovering an inpainted image. For this example, the deep decoder and the deep image perform almost equally well.

decoder outperforms bicubic interpolation, and is on-par with the deep image prior (see Table 1 in the appendix).

4.3 Inpainting

Finally, we use the deep decoder for inpainting, where we are given an inpainted image $\mathbf{y}$, and a forward model f mapping a clean image to an inpainted image. The forward model f is defined by a mask that describes the inpainted region, and simply maps that part of the image to zero. Figure 3 and Table 1 demonstrate that the deep decoder performs well on the inpainting problems; however, the deep image prior performs slightly better on average over the examples considered. For the impainting problem we choose a significantly more expressive prior, specifically $k = 320$.

5 Related work

Image compression, restoration, and recovery algorithms are either trained or untrained. Conceptually, the deep decoder image model is most related to untrained methods, such as sparse representations in overcomplete dictionaries (for example wavelets [Don95] and curvelets [Sta+02]). A number of highly successful image restoration and recovery schemes are not directly based on generative image models, but rely on structural assumptions about the image, such as exploiting self-similarity in images for denoising [Dab+07] and super-resolution [Gla+09].

Since the deep decoder is an image-generating deep network, it is also related to methods that rely on trained deep image models. Deep learning based methods are either trained end-to-end for tasks ranging from compression [Tod+16; Agu+17; The+17; Bur+12; Zha+17] to denoising [Bur+12; Zha+17], or are based on learning a generative image model (by training an autoencoder or GAN [HS06; Goo+14]) and then using the resulting model to solve inverse problems such as compressed sensing [Bor+17; HV18], denoising [Hec+18], phase retrieval [Han+18; SA18], and blind deconvolution [Asi+18], by minimizing an associated loss. In contrast to the deep decoder, where the optimization is over the weights of the network, in all the aforementioned methods, the weights are adjusted only during training and then are fixed upon solving the inverse problem.

Most related to our work is the Deep Image Prior (DIP), recently proposed by Ulyanov et al. [Uly+18]. The deep image prior is an untrained method that uses a network with an hourglass or encoder-decoder architecture, similar to the U-net and related architectures that work well as autoencoders. The key differences to the deep decoder are threefold: i) the DIP is over-parameterized,

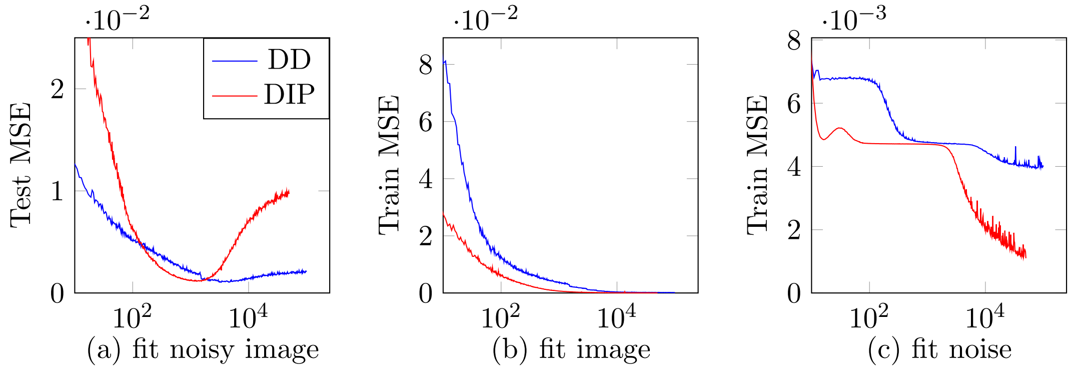

Figure 4: Denoising with the deep decoder and the deep image prior: Due to under-parameterization, the deep decoder can only fit a small proportion of the noise, and thus enables image denoising. Early stopping can enhance the performance. The deep image prior can fit noise very well, but fits an image faster than noise, thus early stopping is critical for denoising performance.

whereas the deep decoder is under-parameterized. ii) Since the DIP is highly over-parameterized, it critically relies on regularization through early stopping and adding noise to its input, whereas the deep decoder does not need to be regularized (however, regularization can enhance performance). iii) The DIP is a convolutional neural network, whereas the deep decoder is not.

We further illustrate point ii) comparing the DIP and deep decoder by denoising the astronaut image from Figure 2. In Figure 4(a) we plot the Mean Squared Error (MSE) over the number of iterations of the optimizer for fitting the noisy astronaut image $\mathbf{x} + \eta$ (i.e., $\left\|G(\mathbf{C}^t) - \mathbf{x}\right\|_2^2$ where $\mathbf{C}^t$ are the parameters of the deep decoder after t iterations). In Figure 4(b) and (c), we plot the loss or MSE associated with fitting the noiseless astronaut image, $\mathbf{x}$, ($\left\|G(\mathbf{C}^t) - \mathbf{x}\right\|_2^2$) and the noise itself, η, ($\left\|G(\mathbf{C}^t) - \eta\right\|_2^2$).

The plots in Figure 4 show that with sufficiently many iterations, both the DIP and the DD can fit the image well. However, even with a large number of iterations, the deep decoder can not fit the noise well, whereas the DIP can. This is not surprising, given that the DIP is over-parameterized and the deep decoder is under-parameterized. In fact, in Section 6 we formally show that due to the underparameterization, the deep decoder can only fit a small proportion of the noise, no matter how and how long we optimize. As a consequence, it filters out much of the noise when applied to a natural image. In contrast, the DIP relies on the empirical observation that the DIP fits a structured image faster than it fits noise, and thus critically relies on early stopping.

6 Discussion on what makes the decoder work

In the previous sections we empirically showed that the deep decoder can represent images well and at the same time cannot fit noise well. In this section, we formally show that the deep decoder can only fit a small proportion of the noise, relative to the degree of underparameterization. In addition, we provide insights into how the components of the deep decoder contribute to representing natural images well, and we provide empirical observations on the sensitivity of the parameters and their distribution.

6.1 The deep decoder can only fit little noise

We start by showing that an under-parameterized deep decoder can only fit a proportion of the noise relative to the degree of underparameterization. At the heart of our argument is the intuition that a method mapping from a low- to a high-dimensional space can only fit a proportion of the noise relative to the number of free parameters. For simplicity, we consider a one-layer network, and ignore the batch normalization operation. Then, the networks output is given by

$$G(\mathbf{C}) = \mathbf{U}_1 \mathrm{relu}(\mathbf{B}_1 \mathbf{C}_1)\mathbf{c}_2 \in \mathbb{R}^n.$$

Here, we take $\mathbf{C} = (\mathbf{C}_1, \mathbf{c}_2)$, where $\mathbf{C}_1$ is a $k \times k$ matrix and $\mathbf{c}_2$ is a k-dimensional vector, assuming that the number of output channels is 1. While for the performance of the deep decoder the choice of upsampling matrix is important, it is not relevant for showing that the deep decoder cannot represent noise well. Therefore, the following statement makes no assumptions about the upsampling matrix $\mathbf{U}_1$.

Proposition 1. *Consider a deep decoder with one layer and arbitrary upsampling and input matrices. That is, let $\mathbf{B}_1 \in \mathbb{R}^{n_1 \times k}$ and $\mathbf{U}_1 \in \mathbb{R}^{n \times n_1}$. Let $\eta \in \mathbb{R}^n$ be zero-mean Gaussian noise with covariance matrix $\mathbf{I}$. Assume that $k^2 \log(n_1)/n \leq 1/32$. Then, with probability at least $1 - 2n_1^{-k^2}$,*

$$\min_{\mathbf{C}} \|G(\mathbf{C}) - \eta\|_2^2 \geq \|\eta\|_2^2 \left(1 - 20\frac{k^2 \log(n_1)}{n}\right).$$

The proposition asserts that the deep decoder can only fit a small portion of the noise energy, precisely a proportion determined by its number of parameters relative to the output dimension, n. Our simulations and preliminary analytic results suggest that this statement extends to multiple layers in that the lower bound becomes $\left(1 - c\frac{k^2 \log(\prod_{i=1}^d n_i)}{n}\right)$, where c is a numerical constant.

6.2 Upsampling

Upsampling is a vital part of the deep decoder because it is the only way that the notion of locality enters the signal model. The choice of the upsampling method strongly affects the 'character' of the resulting signal estimates. We now discuss the impacts of a few choices of upsampling matrices $\mathbf{U}_i$, and their impact on the images the model can fit.

No upsampling: If there is no upsampling, or, equivalently, if $\mathbf{U}_i = \mathbf{I}$, then there is no notion of locality in the resulting image. All pixels become decoupled, and there is then no notion of which pixels are near to each other. Specifically, a permutation of the input pixels (the rows of $\mathbf{B}_1$) simply induces the identical permutation of the output pixels. Thus, if a deep decoder without upsampling could fit a given image, it would also be able to fit random permutations of the image equally well, which is practically equivalent to fitting random noise.

Nearest neighbor upsampling: If the upsampling operations perform nearest neighbor upsampling, then the output of the deep decoder consists of piecewise constant patches. If the upsampling doubles the image dimensions at each layer, this would result in patches of $2^d \times 2^d$ pixels that are constant. While this upsampling method does induce a notion of locality, it does so too strongly in the sense that squares of nearby pixels become identical and incapable of fitting local variation within natural images.

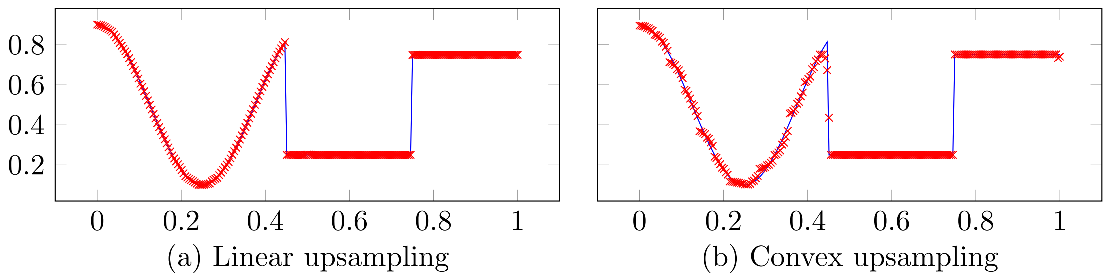

Figure 5: The blue curves show a one-dimensional piecewise smooth signal, and the red crosses show estimates of this signal by a one-dimensional deep decoder with either linear or convex upsampling. We see that linear upsampling acts as an indirect signal prior that promotes piecewise smoothness.

Linear and convex, non-linear upsampling: The specific choice of upsampling matrix affects the multiscale 'character' of the signal estimates. To illustrate this, Figure 5 shows the signal estimate from a 1-dimensional deep decoder with upsampling operations given by linear upsampling $(x_0, x_1, x_2, \ldots) \mapsto (x_0, 0.5x_0 + 0.5x_1, x_1, 0.5x_1 + 0.5x_2, x_2, \ldots)$ and convex nonlinear upsampling given by $(x_0, x_1, x_2, \ldots) \mapsto (x_0, 0.75x_0 + 0.25x_1, x_1, 0.75x_1 + 0.25x_2, x_2, \ldots)$. Note that while both models are able to capture the coarse signal structure, the convex upsampling results in a multiscale fractal-like structure that impedes signal representation. In contrast, linear upsampling is better able to represent smoothly varying portions of the signal. Linear upsampling in a deep decoder indirectly encodes the prior that natural signals are piecewise smooth and in some sense have approximately linear behavior at multiple scales

6.3 Network input

Throughout, the network input is fixed. We choose the network input $\mathbf{B}_1$ by choosing its entries uniformly at random. The particular choice of the input is not very important; it is however desirable that the rows are incoherent. To see this, as an extreme case, if any two rows of $\mathbf{B}_1$ are equal and if the upsampling operation preserves the values of those pixels exactly (for example, as with the linear upsampling from the previous section), then the corresponding pixels of the output image is also exactly the same, which restricts the range space of the deep decoder unrealistically, since for any pair of pixels, the majority of natural images does not have exactly the same value at this pair of pixels.

6.4 Image generation by successive approximation

The deep decoder is tasked with coverting multiple noise channels into a structured signal primarily using pixelwise linear combinations, ReLU activation funcions, and upsampling. Using these tools, the deep decoder builds up an image through a series of successive approximations that gradually morph between random noise and signal. To illustrate that, we plot the activation maps (i.e., $\mathrm{relu}(\mathbf{B}_i\mathbf{C}_i)$) of a deep decoder fitted to the phantom MRI test image (see Figure 6). We choose a deep decoder with $d = 5$ layers and $k = 64$ channels. This image reconstruction approach is in contrast to being a semantically meaningful hierarchical representation (i.e., where edges get combined into corners, that get combined into simple shapes, and then into more complicated

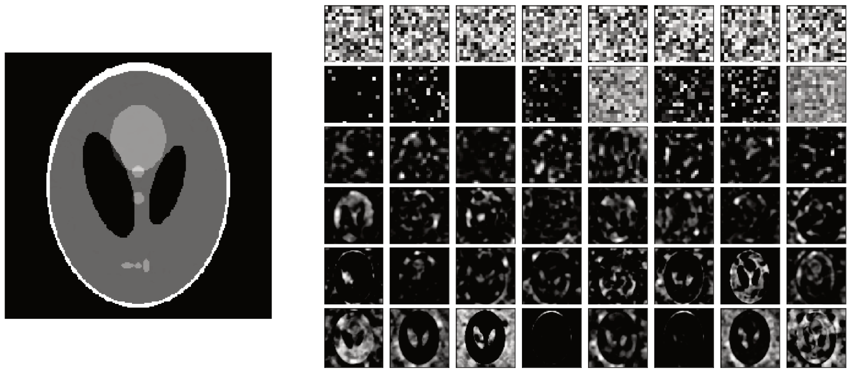

Figure 6: The left panel shows an image reconstruction after training a deep decoder on the MRI phantom image (PSNR is 51dB). The right panel shows how the deep decoder builds up an image starting from a random input. From top to bottom are the input to the network and the activation maps (i.e., relu($\mathbf{B}_i \mathbf{C}_i$)) for eight out of the 64 channels in layers one to six.

shapes), similar to what is common in discriminative networks.

Code

Code to reproduce the experiments is available at github.com/reinhardh/supplement_deep_decoder.

Acknowledgements

RH is partially supported by NSF award IIS-1816986 and would like to thank Ludwig Schmidt for helpful discussions on the deep decoder in general, and in particular for suggestions on the experiments in Section 2.

		barbara	lovett	mri	zebra	F16	baboon	fruit	astronaut	castle	saturn
DN	identity	20.3	20.9	22.1	21.3	20.3	20.3	20.5	20.6	20.4	20.2
	DD128	**26.8**	**27.9**	26.9	22.5	**29.1**	21.4	**29.2**	**29.8**	**27.7**	29.0
	DIP	24.4	25.3	26.6	**24.8**	25.0	**22.8**	25.7	26.1	25.0	25.0
	BM3D	24.7	25.1	**28.0**	22.8	25.2	22.6	26.3	26.2	25.6	**30.5**
SR	bicubic	26.3	26.0	24.5	18.2	26.4	**20.7**	27.1	29.3	25.8	27.9
	DD128	**26.4**	26.3	**26.4**	19.0	26.6	20.6	**28.6**	**30.2**	**26.1**	27.8
	DIP	26.4	**26.6**	25.6	**19.2**	**27.4**	20.6	28.3	29.6	26.0	**27.9**
IP	identity	14.9	14.4	18.3	13.0	11.7	14.0	12.4	14.0	14.2	13.4
	DD320	30.8	**32.9**	31.1	23.6	**34.7**	23.9	35.3	**36.1**	31.6	35.4
	DIP	**35.6**	26.9	**32.1**	**24.2**	34.7	**26.2**	**35.5**	35.3	**32.6**	**36.2**

Table 1: Performance comparison of the deep decoder for denoising (DN), superresolution (SR), and inpainting (IP).

References

[Agu+17] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool. “Soft-to-hard vector quantization for end-to-end learning compressible representations”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 1141–1151.

[Ant+92] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. “Image coding using wavelet transform”. In: *IEEE Transactions on Image Processing* 1.2 (1992), pp. 205–220.

[Asi+18] M. Asim, F. Shamshad, and A. Ahmed. “Solving bilinear inverse problems using deep generative priors”. In: *arXiv preprint arXiv:1802.04073* (2018).

[Bor+17] A. Bora, A. Jalal, E. Price, and A. G. Dimakis. “Compressed sensing using generative models”. In: *arXiv:1703.03208* (2017).

[Bur+12] H. C. Burger, C. J. Schuler, and S. Harmeling. “Image denoising: Can plain neural networks compete with BM3D?” In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 2392–2399.

[Dab+07] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. “Image denoising by sparse 3-D transform-domain collaborative filtering”. In: *IEEE Transactions on Image Processing* 16.8 (2007), pp. 2080–2095.

[Don95] D. L. Donoho. “De-noising by soft-thresholding”. In: *IEEE Transactions on Information Theory* 41.3 (1995), pp. 613–627.

[Gla+09] D. Glasner, S. Bagon, and M. Irani. “Super-resolution from a single image”. In: *IEEE International Conference on Computer Vision*. 2009, pp. 349–356.

[Goo+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A Courville, and Y. Bengio. “Generative adversarial nets”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[HV18] P. Hand and V. Voroninski. “Global guarantees for enforcing deep generative priors by empirical risk”. In: *Conference on Learning Theory*. arXiv:1705.07576. 2018.

[Han+18] P. Hand, O. Leong, and V. Voroninski. “Phase Retrieval Under a Generative Prior”. In: *arXiv preprint arXiv:1807.04261* (2018).

[Hec+18] R. Heckel, W. Huang, P. Hand, and V. Voroninski. “Deep denoising: Rate-optimal recovery of structured signals with a deep prior”. In: *arXiv:1805.08855* (2018).

[HS06] G. E. Hinton and R. R. Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *Science* 313.5786 (2006), pp. 504–507.

[IS15] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International Conference on Machine Learning*. 2015, pp. 448–456.

[LM00] B. Laurent and P. Massart. “Adaptive estimation of a quadratic functional by model selection”. In: *The Annals of Statistics* 28.5 (2000), pp. 1302–1338.

[Rom+17] Y. Romano, M. Elad, and P. Milanfar. “The little engine that could: Regularization by denoising (RED)”. In: *SIAM Journal on Imaging Sciences* 10.4 (2017), 18041844.

[SA18] F. Shamshad and A. Ahmed. “Robust Compressive Phase Retrieval via Deep Generative Priors”. In: *arXiv preprint arXiv:1808.05854* (2018).

[Sta+02] J.-L. Starck, E. J. Candes, and D. L. Donoho. "The curvelet transform for image denoising". In: *IEEE Transactions on Image Processing* 11.6 (2002), pp. 670–684.

[The+17] L. Theis, W. Shi, A. Cunningham, and F. Huszár. "Lossy image compression with compressive autoencoders". In: *arXiv:1703.00395* (2017).

[Tod+16] G. Toderici, S. M. OMalley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. "Variable rate image compression with recurrent neural networks". In: *International Conference on Learning Representations*. 2016.

[Uly+18] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Deep image prior". In: *Conference on Computer Vision and Pattern Recognition*. 2018.

[Win66] R. O. Winder. "Partitions of N-space by hyperplanes". In: *SIAM Journal on Applied Mathematics* 14.4 (1966), pp. 811–818.

[Zha+17] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising". In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.

Appendix

A Proof of Proposition 1

Suppose that the network has two layers, i.e., $G(\mathbf{C}) = \mathbf{U}_d \mathrm{relu}(\mathbf{B}_1\mathbf{C})\mathbf{c}_2$. We start by re-writing $\mathbf{B}_2 = \mathrm{relu}(\mathbf{B}_1\mathbf{C})$ in a convenient form. For a given vector $\mathbf{x} \in \mathbb{R}^n$, denote by $\mathrm{diag}(\mathbf{x} > 0)$ the matrix that contains one on its diagonal if the respective entry of $\mathbf{x}$ is positive and zero otherwise. Let $\mathbf{c}_{jci}$ denote the i-th column of $\mathbf{C}_j$, and denote by $\mathbf{W}_{ji} \in \{0,1\}^{k\times k}$ the corresponding diagonal matrix $\mathbf{W}_{ji} = \mathrm{diag}(\mathbf{B}_j\mathbf{c}_{jci} > 0)$. With this notation, we can write

$$\mathbf{B}_2 = \mathrm{relu}(\mathbf{B}_1\mathbf{C}_1) = [\mathbf{W}_{11}\mathbf{B}_1\mathbf{c}_{1c1}, \ldots, \mathbf{W}_{1k}\mathbf{B}_1\mathbf{c}_{1ck}].$$

Thus,

$$G(\mathbf{C}) = \mathbf{U}_1[\mathbf{W}_{11}\mathbf{B}_1, \ldots, \mathbf{W}_{1k}\mathbf{B}_1] \begin{bmatrix} \mathbf{c}_{1c1}[\mathbf{c}_2]_1 \\ \vdots \\ \mathbf{c}_{1c1}[\mathbf{c}_2]_k \end{bmatrix},$$

where $[\mathbf{c}_2]_i$ denotes the i-th entry of $\mathbf{c}_2$. Thus, $G(\mathbf{C})$ lies in the union of at-most-k^2-dimensional subspaces of $\mathbb{R}^n$, where each subspace is determined by the matrices $\{\mathbf{W}_{1j}\}_{j=1}^k$. The number of those subspaces is bounded by n^{k^2}. This follows from the fact that for each matrix $\mathbf{W}_{1j}$, by Lemma 1 below, the number of different matrices is bounded by n^k. Since there are k matrices, the number of different sets of matrices is bounded by n^{k^2}.

Lemma 1. *For any $\mathbf{W} \in \mathbb{R}^{n\times k}$ and $k \geq 5$,*

$$|\{\mathrm{diag}(\mathbf{W}\mathbf{v} > 0)\mathbf{W} | \mathbf{v} \in \mathbb{R}^k\}| \leq n^k.$$

Next, fix the matrixes $\{\mathbf{W}_{1j}\}_j$. As $G(\mathbf{C})$ lies in an at-most-k^2-dimensional subspace, let S be a k^2-dimensional subspace that contains the range of G for these fixed $\{\mathbf{W}_{1j}\}_j$. It follows that

$$\min_{\mathbf{C}} \|G(\mathbf{C}) - \eta\|_2^2 \geq \frac{\|P_{S^c}\eta\|_2^2}{\|\eta\|_2^2}. \tag{1}$$

Now, we make use of the following bound on the projection of the noise η onto a subspace.

Lemma 2. *Let $S \subset \mathbb{R}^n$ be a subspace with dimension ℓ. Let $\eta \sim \mathcal{N}(0, I_n)$ and $\beta \geq 1$. Then,*

$$\mathrm{P}\left[\frac{\|P_{S^c}\eta\|_2^2}{\|\eta\|_2^2} \geq 1 - \frac{10\beta\ell}{n}\right] \geq 1 - e^{-\beta\ell} - e^{-n/16}.$$

Proof of Lemma 2. From Laurent and Massart [LM00, Lem. 1], if $X \sim \chi_n^2$, then

$$\mathrm{P}\left[X - n \geq 2\sqrt{nx} + 2x\right] \leq e^{-x},$$
$$\mathrm{P}\left[X \leq n - 2\sqrt{nx}\right] \leq e^{-x}.$$

With these, we obtain

$$\mathrm{P}\left[X \geq 5\beta n\right] \leq e^{-\beta n} \text{ if } \beta \geq 1, \tag{2}$$
$$\mathrm{P}\left[X \leq n/2\right] \leq e^{-n/16}. \tag{3}$$

We have $\frac{\|P_{S^c}\eta\|_2^2}{\|\eta\|_2^2} = 1 - \frac{\|P_S\eta\|_2^2}{\|\eta\|_2^2}$. Note that $\|P_S\eta\|_2 \sim \chi_\ell^2$ and $\|\eta\|_2^2 \sim \chi_n^2$. Applying inequality (2) to bound $\|P_S\eta\|_2$ and inequality (3) to bound $\|\eta\|_2^2$, a union bound gives that claim. $\square$

Thus, by inequality (1) and Lemma 2 with $\ell = k^2$, for all $\beta \geq 1$,

$$\mathrm{P}\left[\frac{1}{\|\eta\|_2^2}\min_{\mathbf{C}} \|G(\mathbf{C}) - \eta\|_2^2 \geq 1 - \frac{10\beta k^2}{n} \middle| \{\mathbf{W}_{1j}\}_j\right] \geq 1 - e^{-k^2\beta} - e^{-n/16}. \tag{4}$$

Since the number of matrices $\{\mathbf{W}_{1j}\}_j$ is bounded by $n_1^{k^2}$, by a union bound,

$$\mathrm{P}\left[\frac{1}{\|\eta\|_2^2}\min_{\mathbf{C}} \|G(\mathbf{C}) - \eta\|_2^2 \leq 1 - \frac{10\beta k^2}{n}\right] \leq n_1^{k^2}(e^{-\beta k^2} + e^{-n/16}) \leq 2n_1^{-k^2}, \tag{5}$$

where the last inequality follows with choosing $\beta = 2\log(n_1)$ and by the assumption that $k^2 < \frac{n}{32\log n_1}$. This proves the claim in Proposition 1.

A.1 Proof of Lemma 1

Our goal is to count the number of sign patterns $(\mathbf{Wv} > 0) \in \{0, 1\}$. Note that this number is equal to the maximum number of partitions one can get when cutting a k-dimensional space with n many hyperplanes that all pass through the origin, and are perpendicular to the rows of $\mathbf{W}$. This number if well known (see for example Winder [Win66]) and is upper bounded by

$$2\sum_{i=0}^{n-1}\binom{n-1}{k}.$$

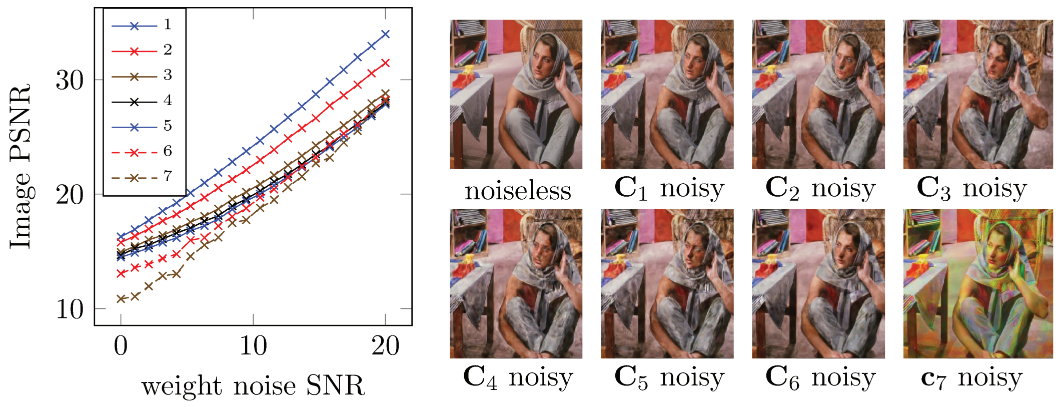

Figure 7: Sensitivity to parameter perturbations of the weights in each layer, and images generated by perturbing the weights in different layers, and keeping the weights in the other layers constant.

Thus,

$$|\{\text{diag}(\mathbf{W}\mathbf{v} > 0)W \colon \mathbf{v} \in \mathbb{R}^k\}| \leq 2\sum_{i=0}^{n-1} \binom{n-1}{k} \leq 2k \left(\frac{e(n-1)}{k}\right)^k \leq n^k,$$

where the last inequality holds for $k \geq 5$.

B Sensitivity to parameter perturbations and distribution of parameters

The deep decoder is not overly sensitive to perturbations of its coefficients. To demonstrate this, fit the standard test image Barbara with a deep decoder with 6 layers and $k = 128$, as before. We then perturb the weights in a given layer i (i.e., the matrix $\mathbf{C}_i$) with Gaussian noise of a certain signal-to-noise ratio relative to $\mathbf{C}_i$ and leave the other weights and the input untouched. We then measure the peak signal-to-noise ratio in the image domain, and plot the corresponding curve for each layer (see Figure 7). It can be seen that the representation provided by the deep decoder is relatively stable with respect to perturbations of its coefficients, and that it is more sensitive to perturbations in higher levels.

Finally, in Figure 8 we depict the distribution of the weights of the network after fitted to the Barbara test image, and note that the weights are approximately Gaussian distributed.

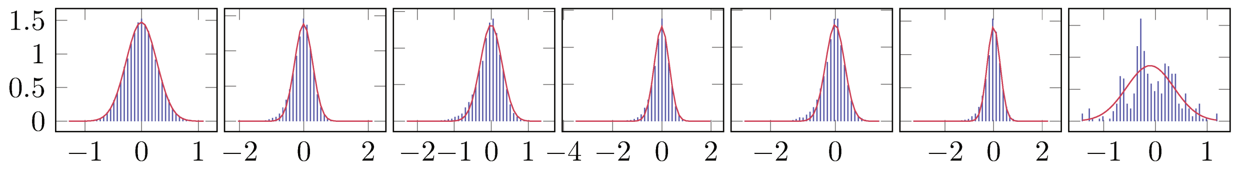

Figure 8: Distribution of the weights for fitting the test image Barbara along with a Gaussian fit: The distribution of the weighs is approximately Gaussian.