Implicit bias of gradient descent on convolutional generators

Reinhard Heckel* and Paul Hand†

*Dept. of Electrical and Computer Engineering, Rice University
†Dept. of Mathematics and Khoury College of Computer Sciences, Northeastern University

May 5, 2019

Abstract

The majority of image generating neural networks have a convolutional architecture, and this architecture has empirically been proven to be efficient at representing natural images. It is widely acknowledged that convolutional neural networks incorporate strong prior assumptions about natural images, sometimes referred to as an inductive or implicit bias. An instantiation of this bias towards simple, natural images, is that an un-trained overparmeterized convolutional network optimized with gradient descent fits a single natural image significantly faster than complex images such as noise. In this paper, we isolate this effect and demonstrate that it is most pronounced if the network incorporates convolutions with a fixed and smooth upsampling kernel. This effect can be understood as the network architecture imposing constraints so that an natural image is closer in parameter space to a random initialization than a highly complex image such as noise.

1 Introduction

Convolutional neural networks are extremely popular for classification problems arising in computer vision. A convolutional network is designed with the two-dimensional structure of an image in mind and incorporates implicit assumptions about the structure of the objects to be classified, such as locality. Implicit assumptions in the form of the (convolutional) architecture are widely believed to contribute to the ability of neural networks to generalize to unseen data. In a classic paper, Tom Mitchell [Mit80] refers to this as a "bias for choosing one generalization over the other", and argues that such a bias is necessary for a method to make the "inductive leap to characterize new instances". Since then, prior modeling assumptions are often referred to as "inductive biases".

Convolutional neural networks are equally popular for image generation. Specifically, the majority of image generating networks are convolutional, ranging from the Deep Convolutional Generative Adversarial Networks (DCGANs) [Rad+15] to the U-Net [Ron+15], suggesting that those networks are efficient are representing images. A recent work, the deep image prior [Uly+18], demonstrated that convolutional image generating networks indeed have a bias towards "natural" or simple images by showing that an un-trained overparameterized convolutional auto-encoder network fits a single natural image faster when optimized with gradient descent (i.e., with significantly fewer iterations) than it fits pure noise. This observation enables image restoration performance on par with state of the art methods.

Building on those insights, in a recent work, we have proposed a deep image generating network [HH19], termed the deep decoder, that in contrast to the deep image prior is underparameterized, which besides having obvious computational advantages, enables compression in addition to state of the art image restoration performance. The deep decoder has an architecture similar

to convolutional neural network in that it consists of upsampling operations and learned 1×1 convolutions.

Both works [Uly+18; HH19] illustrate a bias of convolutional convolutional architectures towards natural images. The deep decoder is explicitly biased towards natural images in that with few parameters it can fit natural images (as shown empirically), but cannot fit noise (provably). The deep image prior is overparameterized and thus can fit noise in principle, but does so much slower compared to fitting a natural images, a phenomena in the intersection of architecture and optimization. The goal of this paper is to isolate and study this phenomena. We hasten to add that the phenomena that convolutional neural networks optimized with gradient descent fit structure faster than noise has also been observed in the context of classification, specifically [Zha+16, Fig. 1(a)] shows that true labels are fitted faster than the shuffled labels.

In this paper, we consider a simple untrained convolutional neural network that only consists of convolutional-like operations. We consider architectures that have a sufficient number of parameters to represent an arbitrary image and study the following phenomena in the intersection of the network's architecture and optimization:

Convolutional generators optimized with gradient descent
fit natural images faster than complex ones.

With complex images, we mean images that consist of a large number of 'edges', such an extremely fine checkerboard or images of noise.

The paper is organized as follows. We first introduce the architecture we consider, and then perform experiments demonstrate the effect, validating it on a large datset, and demonstrating that the distance between the final and initial networks weights is a key feature that determines the difference of fitting a simple and a complex image. In the appendix, we then identify a simple toy models that exhibits the effect.

2 Convolutional generators

Throughout, we consider convolutional generators G, that map a randomly chosen and fixed tensor $\mathbf{B}_0 = [\mathbf{b}_{01}, \ldots, \mathbf{b}_{0k}] \in \mathbb{R}^{n_0 \times k}$ consisting of k many n_0-dimensional channels or images 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. The network transforms the fixed input tensor $\mathbf{B}_0$ to an image only using upsampling/no-upsampling and convolutional operations, followed by channel normalization (a special case of batch normalization) and an application of a ReLU non-linearity, see Figure 1 for an illustration. We initially consider the following four closely related architectural choices, which differ in the upsampling/no-upsampling and convolutional operations which generate the activations in the $(i+1)$-st layer, $\mathbf{B}_{i+1}$, from the activations in the i-th layer, $\mathbf{B}_i$:

i) **Bilinear upsampling and linear combinations.** Layer $i+1$ is obtained by linearly combining the channels of layer i with learnable coefficients, followed by bi-linear upsampling. Note that linearly combining the channels is equivalent to performing one-times-one convolutions. This is the deep decoder architecture proposed in [HH19],

ii) **Fixed interpolation kernels and linear combinations.** Layer $i+1$ is obtained by linearly combining the channels of layer i with learnable coefficients followed by convolving each channel with the same 4x4 interpolation kernel that is used in the linear upsampling.

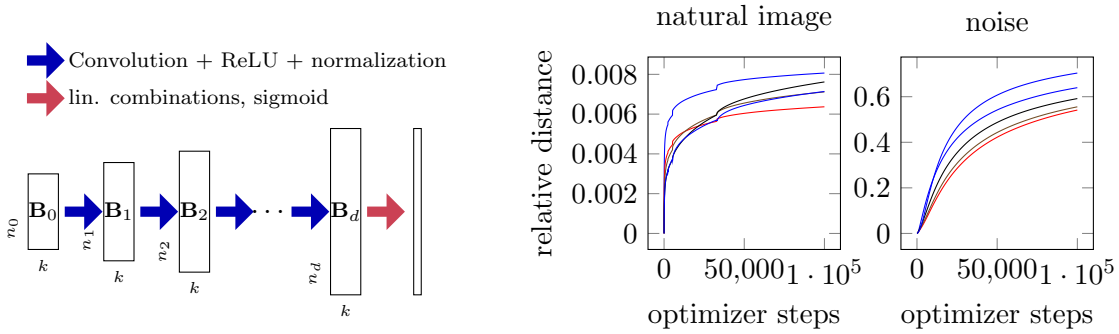

Figure 1: **Left panel:** In the generators we study, the output is generated through repeated convolutional layers, ReLU activations, and channel normalization. **Right panel:** The relative distances of the weights in each layer from its random initialization. The weights need to change significantly more to fit the noise, compared to an image, thus a natural image lies closer to a random initialization than noise.

iii) **Parameterized convolutions:** Layer $i+1$ is obtained from layer i though a convolutional layer with 4x4 convolutions.

iv) **Deconvolutional network:** Layer $i+1$ is obtained from layer i though a deconvolution layer with 4x4 convolutions.

To emphasize that i)-iv) are structurally extremely similar operations, we recall that each operation consists only of upsampling and convolutional operations. Let $\mathbf{T}(\mathbf{c})\colon \mathbb{R}^n \to \mathbb{R}^n$ be the convolutional operator with kernel $\mathbf{c}$, let $\mathbf{u}$ the linear upsampling kernel (equal to $\mathbf{u} = [0.5, 1, 0.5]$ in the one-dimensional case), and let $\mathbf{U}\colon \mathbb{R}^n \to \mathbb{R}^{2n}$ be an upsampling operator, that in the one dimensional case transforms $[x_1, x_2, \ldots, x_n]$ to $[x_1, 0, x_2, 0, \ldots, x_n, 0]$. In each of the architectures i)-iv), the ℓ-th channel of layer $i+1$ is obtained from the channels in the i-th layer as: $\mathbf{b}_{i+1,\ell} = \text{relu}\left(\sum_{j=1}^{k} \mathbf{M}(\mathbf{c}_{ij\ell})\mathbf{b}_i\right)$, where the linear operator $\mathbf{M}$ is defined as follows for the four architectures

$$\text{i) } \mathbf{M}(c) = c\mathbf{T}(\mathbf{u})\mathbf{U}, \quad \text{ii) } \mathbf{M}(c) = c\mathbf{T}(\mathbf{u}), \quad \text{iii) } \mathbf{M}(\mathbf{c}) = \mathbf{T}(\mathbf{c}), \quad \text{iv) } \mathbf{M}(\mathbf{c}) = \mathbf{T}(\mathbf{c})\mathbf{U}\ .$$

The coefficients associated with the i-th layer are given by $\mathbf{C}_i = \{\mathbf{c}_{ij\ell}\}$, and all coefficients of the networks are $\mathbf{C} = \{\mathbf{c}_{ij\ell}\}$. Note that here, the coefficients or parameters of the networks are the weights and not the input to the network.

3 Demonstrating implicit bias of convolutional generators

In this section, we demonstrate the phenomenon that convolutional generators, in particular those with *fixed* convolutional operations, fit natural or simple images significantly faster than noise and other complex unnatural images. Throughout this section, for each image or signal $\mathbf{x}^*$ we fit weights by minimizing the loss

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

with respect to the network parameters $\mathbf{C}$ using plain gradient descent with a fixed stepsize.

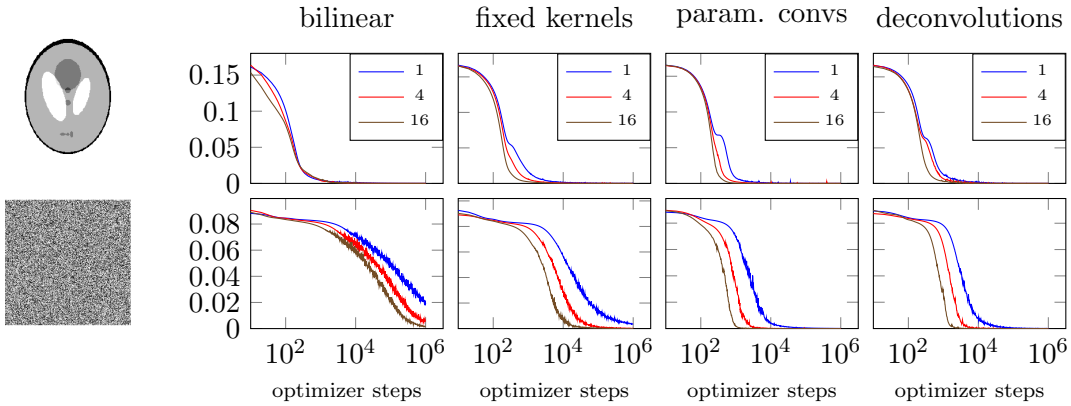

Figure 2: Fitting the phantom MRI and noise with different architectures, for different number of overparameterization factors (1,4, and 16). Gradient descent on convolutional generators involving fixed convolutional matrixes fit an image significantly faster than noise.

Evidence of phenomenon: In order to exemplify the effect, we fit the phantom MRI image as well as noise for each of the architectures above for a 5-layer network. We choose the number of channels, k, such that the overparameterization factor (i.e., the ratio of number of parameters of the network over the output dimensionality) is $1, 4$, and 16, respectively. The results show that for architectures i) and ii) involving fixed convolutional operations, gradient descent requires more than an order of magnitude fewer iterations to obtain a good fit of the phantom MRI image relative to noise. For architectures ii) and iii), we see a smaller effect, but the effect essentially vanishes when the network is highly overparameterized.

Replication: The effect above is replicable for natural images. Figure 3 in the appendix shows the average and standard deviation of the loss curves of 100 randomly chosen images from the ImageNet dataset for architecture i. The natural images are fit by gradient descent several orders of magnitude faster than noise is. We also note that the effect continues to exist for other unnatural images, such as checkerboard images in which each pixel alternates between 1 and 0.

Hint at an explanation: We highlight a strong contrast between fitting natural images and noise: with natural images, the distance between final and initial network weights is significantly smaller than it is when fitting images of noise. As demonstration, we again fit the phantom MRI image and noise for architecture i) and an overparameterization factor of 4 and record, for each layer i the relative distance $\|\mathbf{C}_i^{(t)} - \mathbf{C}_i^{(0)}\|/(\|\mathbf{C}_i^{(t)}\| + \|\mathbf{C}_i^{(0)}\|)$, where $\mathbf{C}_i^{(0)}$ are the weights at initialization (we initialize randomly), and $\mathbf{C}_i^{(t)}$ are the weights at the optimizer step t. Figure 1 shows that to fit noise, the weights have to change significantly, while for fitting a natural image they only have to change slightly. We hasten to add the the weight change should be understood relative to a scaling and bias introduced by the channel normalization. Specifically, the channel normalization operation, applied at each layer, normalizes each channel individually. Specifically, let $\mathbf{Z}_i$ be the channels in the i-th layer right before the ReLU operation, and let $\mathbf{z}_{ij}$ be the j-th channel in the i-th layer. Then channel 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, optimized for independently for each channel, and ϵ is a fixed small constant.

References

[HH19] R. Heckel and P. Hand. "Deep Decoder: Concise image representations from untrained non-convolutional networks". In: *International Conference on Learning Representations*. 2019.

[Mit80] T. Mitchell. "The need for biases in learning generalizations". In: *Rutgers Computer Science Tech. Rept. CBM-TR-117* (1980).

[Rad+15] A. Radford, L. Metz, and S. Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[Ron+15] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[Uly+18] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Deep image prior". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9446–9454.

[Zha+16] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. "Understanding deep learning requires rethinking generalization". In: *arXiv preprint arXiv:1611.03530* (2016).

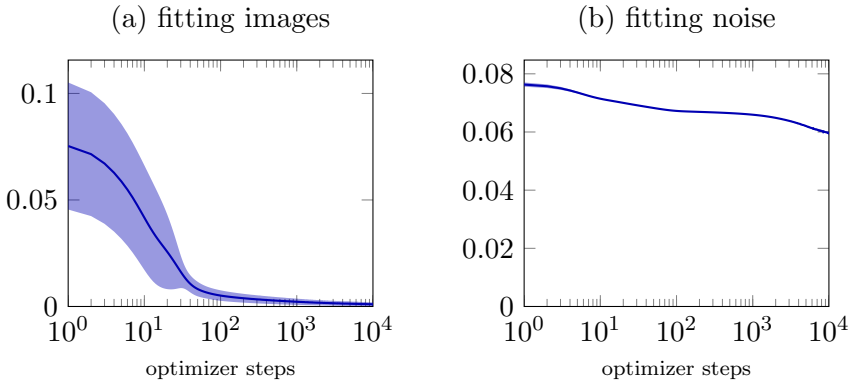

Figure 3: The loss curves for architecture i), a convolutional generator with linear upsampling operations, averaged over 100 $3 \times 512 \times 512$ (color) images from the Imagenet dataset. The error bars in the right panel are present but are very small. Convolutional generators fit natural images significantly faster than noise.

A toy example isolating the effect: We end this section with a minimal example where the effect is still visible. Towards this goal, we consider a one-dimensional convolutional generator with architecture i). Note that this generator can be written as $G(\mathbf{C}) = \text{relu}(\mathbf{T}(\mathbf{u})\mathbf{U}\mathbf{B}_1\mathbf{C}_1)\mathbf{c}_2$, where $\mathbf{C}_1 \in \mathbb{R}^{k \times k}$ and $\mathbf{c}_2 \in \mathbb{R}^k$ are the coefficients of the generator. Figure 4 show that even for this simple model, the convolutional generator fits the simple image (i.e., the step function) faster than the complex one (the noise). This is the simplest model that we could find for which this

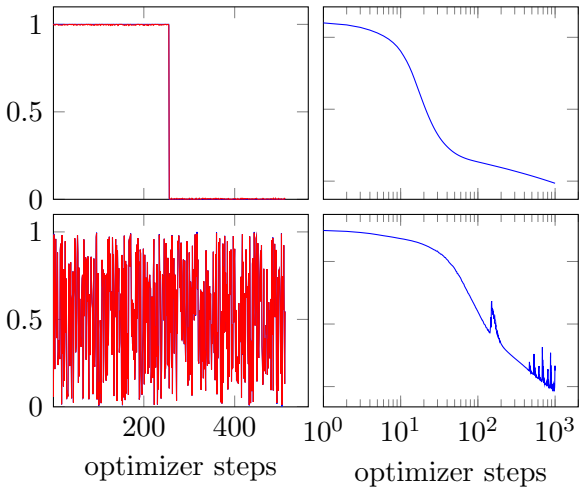

Figure 4: The toy model can accurately fit the 1d step funtion and noise functions in the left panenls. The right panels show the loss versus iteration number for fitting these functions. The step function is fit by the toy model almost an entire order of magnitude earlier than the noisy image.

effect can reliably observed, thus suggesting that fixed convolutional operations (for example due to upsampling) are at the heart of the effect.