INVERTING LAYERS OF A LARGE GENERATOR

David Bau[1], Jun-Yan Zhu[1], Jonas Wulff[1], William Peebles[1],
Hendrik Strobelt[2], Bolei Zhou[3], Antonio Torralba[1]
[1]Massachusetts Institute of Technology,
[2]IBM Research, [3]The Chinese University of Hong Kong

1 INTRODUCTION

The remarkable realism achieved by state-of-the-art Generative Adversarial Networks for natural images leads us to ask: how can we see what a large GAN is *unable* to generate? The problem of mode-dropping — the tendendency of a GAN to give up on modeling significant parts of the target distribution — is seen as one of the most serious shortcomings of GANs (Goodfellow, 2016; Li & Malik, 2018), but generated samples provide little insight. A sample reveals what a GAN does, not what it is unable to do. Therefore, methods for characterizing GAN omissions are needed.

Given a trained generator $G : \mathbf{Z} \rightarrow \mathbf{X}$, we wish to understand whether any particular image $\mathbf{x}$ can be synthesized by G; that is, we want to understand whether $\mathbf{x} \in \text{range}(G)$, where $\text{range}(G)$ denotes the set of images that can be output by G. The direct way to solve this would be to find an encoder $E : \mathbf{X} \rightarrow \mathbf{Z}$ that can invert G so that $E(G(\mathbf{z})) = \mathbf{z}$; then we would know that $\mathbf{x} \in \text{range}(G)$ exactly when $\mathbf{x} = G(E(\mathbf{x}))$. Unfortunately, as we shall see, large generators with many layers are not easy to fully invert: traditional methods do not succeed at constructing an accurate E.

A tractable subproblem can be defined. Decompose $G = G_F(g_j(\cdots g_2(g_1(\mathbf{z}))))$, where $\{g_i\}$ represents a few initial layers of G, and G_F groups together all the final layers. The decomposition guarantees $\text{range}(G) \subset \text{range}(G_F)$, so if we can identify images $\mathbf{x} \notin \text{range}(G_F)$, we will know those images are also absent from $\text{range}(G)$. Therefore, in this paper we develop a method for inverting the final layers of a large generator. We seek an encoder $E_F : \mathbf{X} \rightarrow \mathbf{Z}_j$ such that

$$E_F(G_F(\mathbf{z}_j)) = \mathbf{z}_j \tag{1}$$

When $\mathbf{x} \neq G_F(E_F(\mathbf{x}))$, then $\mathbf{x} \notin \text{range}(G_F)$, and therefore $\mathbf{x} \notin \text{range}(G)$. Furthermore, the differences between $\mathbf{x}$ and $G_F(E_F(\mathbf{x}))$ can provide insight about the nature of the limitations of the GAN. For example, Figure 1 illustrates pairs of images of bedrooms and churches alongside

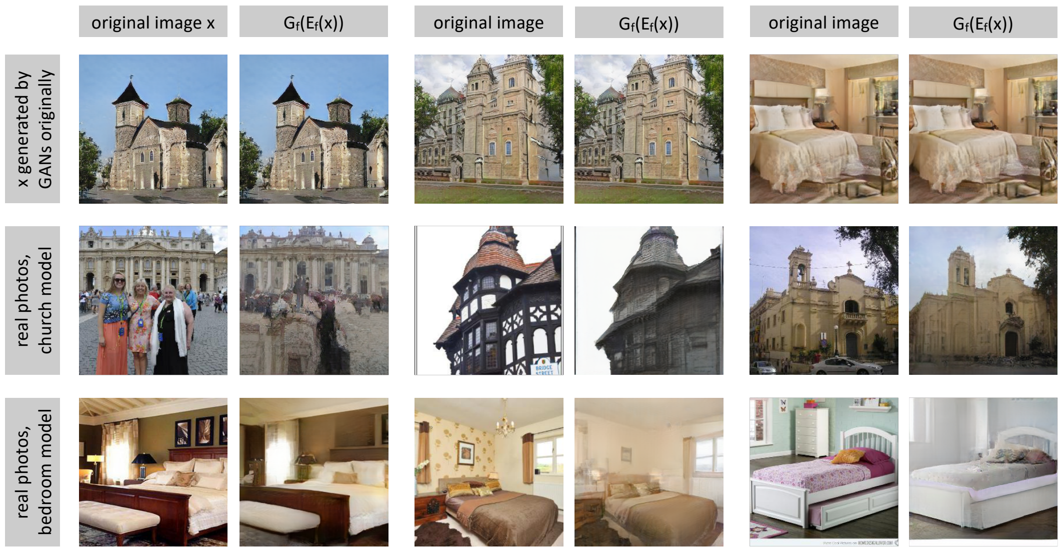
original image x
Gf(Ef(x))
original image
Gf(Ef(x))
original image
Gf(Ef(x))
x generated by GANs originally
real photos, church model
real photos, bedroom model

Figure 1: Using inverses to debug the limitations of a generator. The first row shows nearly perfect inversions of complex GAN-generated images. The second and third rows show that reconstructions of natural photos cannot perfectly replicate all the objects and styles in a real scene.

their reconstructed inversions, showing $\mathbf{x}$ next to $G_F(E_F(\mathbf{x}))$, applying method (f) described in this paper. The first row illustrates images that can be generated exactly (where $G_F(E_F(\mathbf{x}))$ is indistinguishable from $\mathbf{x}$), and the last two rows illustrate natural images that cannot be generated exactly by the generator. The inversions suggest that the analyzed church generator cannot represent people wearing colorful clothing, automobiles, or certain architectural styles. The bedroom generator cannot represent certain types of wall decorations, hanging lights, or dresser drawers.

Previous explorations of inversions of GAN generators have found that inversions can be used to explore the range of a GAN (Zhu et al., 2016); that for a DCGAN left-inverses can be computed to high precision (Lipton & Tripathi, 2017); and that inversions of a GAN for glyphs can reveal specific strokes that the generator is unable to generate (Creswell & Bharath, 2018). While previous work has investigated inversion of 5-layer DCGAN generators, we find that when moving to a 15-layer Progressive GAN, good inversions are more difficult to compute. In the current work, we develop a new inversion method that is effective for these larger GANs.

2 Methods

In our setting, we invert a Progressive GAN (Karras et al., 2018) trained to an output resolution of 256×256, which consists of 15 nonlinear convolution layers. The generator maps a normally distributed random 512-dimensional latent to an output RGB image. Within the generator, every pixel of every latent layer is normalized to lie on the unit hypersphere before being convolved.

We test the following inversion methods.

Method a: Direct optimization of $\mathbf{z}$. Lipton & Tripathi (2017) describe a direct optimization

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} d_I(\mathbf{x}, G(\mathbf{z})) \tag{2}$$

They found that this straightforward approach yields good results when inverting DCGAN. (They further achieved better results if boundaries are avoided, but that technique is unavailable to us since the latent space of a Progressive GAN has no boundaries.) We find slightly improved results by choosing an image similarity metric d_I that adds both L_1 distance in pixels and in a perceptual feature space given by a pretrained VGG; we report results using this perceptual d_I.

Method b: Direct learning of E. Another natural solution is to learn a deep network for E by minimizing expected reconstruction losses over generated images:

$$\mathcal{L}_L(E, G) \equiv \mathbb{E}_{\mathbf{z}}[d_L(\mathbf{z}, E(G(\mathbf{z})))] \tag{3}$$
$$\mathcal{L}_R(E, G) \equiv \mathbb{E}_{\mathbf{z}}[d_I(\mathbf{x}, G(E(\mathbf{x}))) \mid \mathbf{x} = G(\mathbf{z})] \tag{4}$$
$$E_{(b)} = \arg\min_E (\mathcal{L}_L(E, G) + \lambda \mathcal{L}_R(E, G)) \tag{5}$$

This minimizes a distance d_L between true and reconstructed latent vectors together with minimizing pixel reconstruction loss. (Since Progressive GAN normalizes away the magnitude of $\mathbf{z}$, we use cosine similarity $d_L(\mathbf{z}_1, \mathbf{z}_2) = 1 - (\mathbf{z}_1 \cdot \mathbf{z}_2)/(||\mathbf{z}_1|| \cdot ||\mathbf{z}_2||)$, which is invariant to $||\mathbf{z}||$.) The hyperparameter λ determines the balance between reconstructing latents and pixels (we set $\lambda = 1$).

Method c: Optimization of $\mathbf{z}$ after initializing with $E(\mathbf{x})$. This is the method used in Zhu et al. (2016). By initializing method (a) using a $E_{(b)}(\mathbf{x})$, we can achieve improved results. For smaller generators, Zhu et al. (2016) found that this method performs well.

Method d: Learning of E by layers. Instead of learning a network to invert G all at once, we can decompose $G(\mathbf{z}) = G_F(g_4(g_3(g_2(g_1(\mathbf{z})))))$ into layers; then we can apply method (b) on each of the smaller generators g_i separately. (For our 15-layer generator, g_1 through g_4 represent the first four nonlinear convolutions; and G_F contains layers 5-15 in one group.) That is, we learn e_i^* such that:

$$\mathcal{L}_L(e, g_i) \equiv \mathbb{E}_{\mathbf{z}}[d_L(\mathbf{r}_{i-1}, e(g_i(\mathbf{r}_{i-1})))] \quad \text{where } \mathbf{r}_{i-1} = g_{i-1}(\cdots g_1(\mathbf{z}) \cdots) \tag{6}$$
$$\mathcal{L}_R(e, g_i) \equiv \mathbb{E}_{\mathbf{z}}[d_L(\mathbf{r}_i, g_i(e(\mathbf{r}_i)))] \quad \text{where } \mathbf{r}_i = g_i(\cdots g_1(\mathbf{z}) \cdots) \tag{7}$$
$$e_i^* = \arg\min_e (\mathcal{L}_L(e, g_i) + \lambda \mathcal{L}_R(e, g_i)) \tag{8}$$
$$E = e_1^*(e_2^*(e_3^*(e_4^*(e_F^*(\mathbf{x}))))) \tag{9}$$

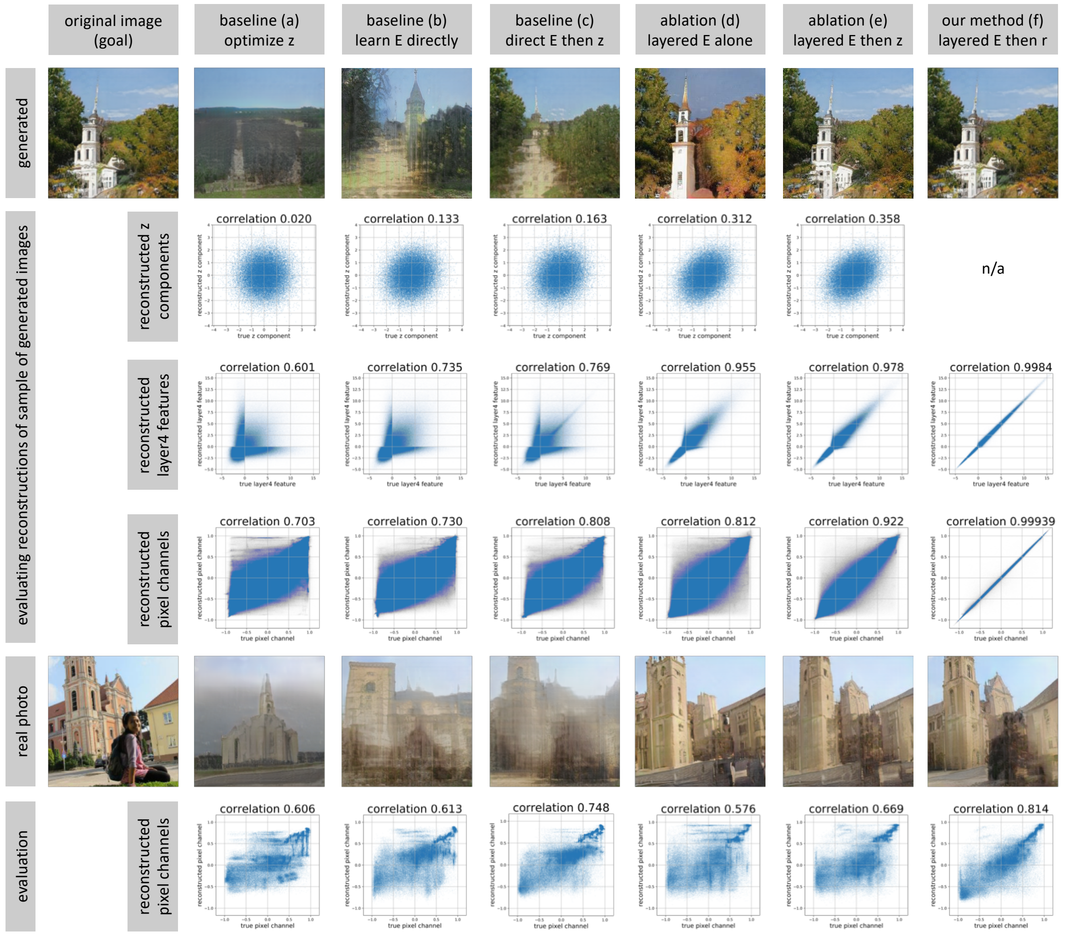

Figure 2: Comparison of methods to invert the generator of Progressive GAN trained to generate LSUN church images. Each of the five methods is described in the text; methods (a) (b) and (c) are baselines, and methods (d), (e), and (f) are variants of our method with different properties. At top, we evaluate behavior on GAN-generated images. Accuracy of reconstructed z, `layer4`, and reconstructed pixels based on a sample of 100 images are shown for each method. Note that method (f) achieves effectively perfect reconstructions of GAN-generated images. Below, we apply each of the methods on a natural non-GAN-generated image.

This method of inverting a network by layers is an application of the classic observation (Hinton & Salakhutdinov, 2006) that a network with many layers can often be trained more easily by pretraining the individual layers separately. The results can be further improved by using this composed network E as an initialization for method (b) and then fine-tuning this composed encoder together. Denote this fine-tuned result as $E_{(d)}$.

Method e: Optimization of z after applying (d). Following the idea of method (c) again, we can use $E_{(d)}(\mathbf{x})$ as an initialization for method (a); this direct optimization of **z** with smart initialization is reported as method (e).

Method f: Optimization of $\mathbf{r}_4$ after applying (d). Motivated by the observation that layers of a GAN generator contain nontrivial semantics (Bau et al., 2019), we now turn to the tractable subproblem of finding an E_F that inverts G_F. As discussed in the introduction, the range of G is covered by the range of G_F, so any inability of G_F to produce an output corresponds to an output G will also fail to produce. To compute an accurate E_F, We found best results initializing with $E_{(d)}(\mathbf{x})$

and optimizing $\mathbf{r}_4$ indirectly by learning residuals on other layers:

$$\mathbf{z}_0 \equiv E_{(d)}(\mathbf{x}) \tag{10}$$

$$\mathbf{z}_4 \equiv g_4(\delta_3 + g_3(\delta_2 + g_2(\delta_1 + g_1(\mathbf{z}_0)))) \tag{11}$$

$$\mathbf{z}_4^* = \arg\min_{\mathbf{z}_4} \left(d_I(\mathbf{x}, G_{(f)}(\mathbf{z}_4)) + \sum_i \lambda_i ||\delta_i||^2 \right) \tag{12}$$

That is, we search by optimizing over δ_i rather than following gradients of $\mathbf{z}_4$ directly. The hyperparameters λ_i prevent the residuals from growing too large (we set $\lambda_i = 1$). This optimization over residuals provides an implicit regularizer over $\mathbf{z}_4$ that favors values that are more similar to values produced by the initial layers g_i. This method of optimizing residuals produces very accurate recovery of the true `layer4` latent.

3 Results and Discussion

Figure 2 compares the six methods. Our new methods (d), (e), and (f) achieve better reconstructions of both latents and pixels than previous methods (a), (b), and (c). Note that method (f) achieves reconstructions of latents and pixels that are nearly perfect.

The nearly-perfect reconstructions of method (f) allow us to precisely identify images that are outside the range of G by using that method to reconstruct an arbitrary image. We can conclude that wherever reconstruction fails, it is almost certainly due to a failure of G to generate the image rather than a failure of E. Therefore, although these methods are trained only on images that are generated by G, our ability to solve that problem completely gives us a useful tool for understanding the limitations of G outside its range. By comparing natural photos with imperfect reconstructions as in Figure 1, we can identify specific objects, parts, and styles that a generator is unable to produce.

References

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Zhou Bolei, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. In *ICLR*, 2019. 3

Antonia Creswell and Anil Anthony Bharath. Inverting the generator of a generative adversarial network. *IEEE transactions on neural networks and learning systems*, 2018. 2

Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 1

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 3

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 2

Ke Li and Jitendra Malik. On the implicit assumptions of gans. *arXiv preprint arXiv:1811.12402*, 2018. 1

Zachary C Lipton and Subarna Tripathi. Precise recovery of latent vectors from generative adversarial networks. *arXiv preprint arXiv:1702.04782*, 2017. 2

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 2