Seeing What a GAN Cannot Generate

David Bau¹, Jun-Yan Zhu¹, Jonas Wulff¹, William Peebles¹
Hendrik Strobelt², Bolei Zhou³, Antonio Torralba¹
¹Massachusetts Institute of Technology, ²IBM Research, ³The Chinese University of Hong Kong

Abstract

Despite the success of Generative Adversarial Networks (GANs), mode collapse remains a serious issue during GAN training. Worse yet, little work has focused on understanding and quantifying which modes have been dropped by a model. In this work, we take a first step and present two analytic methods for systematically studying this phenomenon. First, we deploy a semantic segmentation network to compare the distribution of segmented objects in the generated images with the target distribution in the training set. Differences in segmentation statistics reveal object classes that are omitted by a GAN. Second, given the identified omitted object classes, we further visualize what the GAN is doing instead. In particular, we compare specific differences between individual photos and their approximate reconstructions by a GAN model. To this end, we propose a new image reconstruction method based on inverting the layers of a generator. Finally, we use our framework to analyze several state-of-the-art GANs trained on multiple datasets and identify the typical failure cases of existing models.

1. Introduction

The remarkable ability of a Generative Adversarial Network (GAN) to synthesize realistic images leads us to ask: how can we know what a GAN is unable to generate? Mode-dropping or mode collapse, where a GAN omits portions of the target distribution, is seen as one of the most serious challenges for GANs [10, 20], yet current analysis tools provide little insight into this phenomenon. Quantitative evaluation methods such as Inception scores [29] and Fréchet Inception Distance [13] measure the gap between the generated and the target distribution, but they do not provide explanations of what, specifically, the differences are. Samples of generated images can be used to directly visualize what a GAN is capable of doing. However, such samples do not reveal what a GAN cannot synthesize.

In particular, we wish to know: does a GAN deviate from the target distribution by ignoring difficult images altogether? Or are there specific, semantically meaningful parts and objects that a GAN decides not to learn about? And if so, how can we detect and visualize these missing concepts that a GAN does not generate?

To answer these questions, we propose a pair of new methods that allow us to directly understand the omissions...
of a generator. To analyze the distribution as a whole, we examine Fréchet Segmentation Statistics: we perform semantic segmentation on both generated and ground truth images and compare the distributions of segmentation classes. Any differences in segmentation statistics can be directly interpreted. For example, Figure 1a shows that in a church GAN model, object classes such as people, cars, and fences appear on fewer pixels of the generated distribution as compared to the training distribution.

Once omitted object classes are identified, we ask what the GAN is doing instead. For example, we want to know whether the GAN is distorting instances of missing objects in a specific way, or if it is skipping those objects entirely. To answer such questions, we must find instances where the GAN should generate an object class but is not. Therefore, we propose visualizing the limitations of a GAN using a new reconstruction method called Layer Inversion. This method yields the image that is closest to an arbitrary input image, while being generated by layers of the given GAN. Unlike existing methods to invert a generator [21], our method allows us to invert complex, state-of-the-art GANs.

Deviations between the original image and its reconstruction reveal image features and objects that the generator cannot draw faithfully. We apply our framework to analyze several recent GAN models trained on different scene datasets. Surprisingly, we find that dropped object classes are not distorted or rendered in a low quality or as noise. Instead, they are simply not rendered at all, as if the object was not part of the scene. For example, in Figure 1b, we observe that large human figures are skipped entirely, and the parallel lines in a fence cannot be drawn crisply. This allows a GAN to ignore classes that are too hard, while at the same time producing output of high average visual quality.

2. Related Work

Generative Adversarial Networks (GANs) [11] have enabled many computer vision and graphics applications such as image generation [17, 6, 18], image and video manipulation [16, 30, 45, 15, 34, 26], object recognition [35, 5], and text-to-image [28, 42, 38]. One important issue in this emerging topic is how to evaluate and compare different methods [33, 36]. For example, many evaluation metrics have been proposed to evaluate unconditional GANs such as Inception score [29], Fréchet Inception Distance [13], and Wasserstein Sliced Distance [17]. Though the above metrics can quantify the model performance from different aspects, they cannot explain what visual content the models fail to synthesize. In this work, we aim to provide explanations of a common failure case of GANs: mode collapse. Our analysis complements previous metrics and can provide additional insights into a model’s limitations.

Network Inversion Previous explorations of inversions of GAN generators have found that inversions can be used to explore the space of a GAN [44] or unsupervised feature learning [8]. Later work found that DCGAN left-inverses can be computed to high precision [21, 39], and that inversions of a GAN for glyphs can reveal specific strokes that the generator is unable to generate [7]. 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 much more difficult to compute. In the current work, we develop a layer-wise inversion method that is more effective for these large-scale GANs. We apply a classic layer-wise training approach [14, 4] to the problem of training a deep GAN encoder and further introduce layer-wise instance optimization. Our work is also loosely related to inversion methods for understanding CNN features and classifiers [23, 9, 24, 25]. However, we focus on understanding generative models rather than classifiers.

Understanding and Visualizing Networks Most prior work on network visualization concerns discriminative classifiers [41, 43, 32, 1, 22, 31, 2, 19]. GANs have been visualized by examining the discriminator [27] and by examining the semantics of internal features [3]. Different from recent work [3] that aims to understand what a GAN has learned, our work provides a complementary perspective and focuses on what a GAN cannot generate and what semantic concepts a GAN fails to capture.

3. Method

Our goal is to understand the semantic concepts that a GAN generator cannot generate, in both the entire distribution and in each image instance. To achieve this, we will first measure Fréchet Segmentation Statistics, by segmenting both generated and target images and identifying types of objects that a generator omits when compared to the distribution of real images. Once omitted object classes are identified, we then visualize how they are omitted for individual images. In particular, we find real images that contain the omitted classes, and then project them to the best reconstruction given an intermediate layer of the generator. We call our second step layer-wise inversion.

3.1. Quantifying Distribution-level Mode Collapse

The deviation between a GAN generator and the true distribution of images is often measured using metrics such as Fréchet Inception Distance (FID) [13], which calculate the distance between two distributions in the feature space of a pre-trained network. These metrics quantify differences but cannot explain how the distributions are different.

To gain insight on the differences, we compare the average statistics of the semantic segmentations of generated images with those of ground truth images. The advantage
of measuring differences in terms of segmentation statistics is that individual features are semantically meaningful. Furthermore, segmentations of specific images can be directly visualized to understand the presence or absence of each object class in the generated and target distributions.

To implement this idea, we segment all images using the Unified Perceptual Parsing network [37], which labels each pixel of an image with one of 336 object classes. Within each distribution, we measure the mean area within each image of each object class. Since most classes do not appear on most images, when visualizing segmentation statistics, we sort classes in descending order. We start with the classes with the largest mean area in the ground truth distribution, and we focus our attention on the most common classes.

It is also possible to summarize differences in segmentation in a single number. To do this, we define the Fréchet Segmentation Statistics (FSS), which is an interpretable analog to the popular Fréchet Inception Distance (FID) metric [13]. Like FID, FSS is calculated by first modeling generated and target distributions as Gaussians by collecting means and covariances of sampled segmentation areas; then the Fréchet distance formula provides the distance between the modeled Gaussians:

\[
\text{FSS} \equiv \|\mu_g - \mu_t\|^2 + \text{Tr}(\Sigma_g + \Sigma_t - 2(\Sigma_g \Sigma_t)^{1/2}) \tag{1}
\]

In our FSS calculation, \(\mu_t\) is based on the mean pixel count for each object class over a sample of training images, and \(\Sigma_t\) is based on the covariance of these pixel counts. Similarly \(\mu_g\) and \(\Sigma_g\) reflect segmentation statistics for the generative model. In our experiments, we compare statistics between samples of 10,000 generated and natural images.

Generated segmentation statistics measure the entire distribution: for example, they reveal when a generator omits a particular object class. However, they do not single out specific images where an object should have been generated but was not. To gain further insight, we need a method to visualize omissions of the generator for each image.

### 3.2. Quantifying Instance-level Mode Collapse

To address the above issue, we compare image pairs \((x, x')\) where \(x\) is a real image, in which a particular object class is dropped by a GAN generator \(G\), and \(x'\) is a projection on to the space of all the images that can be generated by layers of the GAN model.

**Defining a tractable inversion problem.** In the ideal case, we would like to find an image that can be perfectly generated by the generator \(G\) and stay close to the real image \(x\). Formally, we seek \(x' = G(z^*)\), where \(z^* = \arg\min_{z} \ell(G(z), x)\) and \(\ell\) is a distance metric in image space. Unfortunately, as shown in Section 4.4, previous methods [44, 8] fail to solve this full inversion problem for recent generators due to the large number of layers in \(G\).

Therefore, we instead solve a tractable subproblem of the full inversion problem. We decompose the generator \(G\) into layers

\[
G = G_f(g_f(\cdots(g_1(z)))) \tag{2}
\]

where \(g_1, \ldots, g_f\) are several early layers of the generator, and \(G_f\) groups all the later layers of the \(G\) together.

Any image that can be generated by \(G\) can also be generated by \(G_f\). That is, if we denote by \(\text{range}(G)\) the set of all images \(x\) that can be output by \(G\), then we have \(\text{range}(G) \subseteq \text{range}(G_f)\). That implies, conversely, that any image that cannot be generated by \(G_f\) cannot be generated by \(G\) either. So any omissions we can identify in \(\text{range}(G_f)\) will also be omissions of \(\text{range}(G)\).

Thus for layer-wise inversion, we visualize omissions by solving the easier problem of inverting the later layers \(G_f\):

\[
x' = G_f(r^*) \quad \tag{3}
\]

where \(r^* = \arg\min_{r} \ell(G_f(r), x)\)

Although we ultimately seek an intermediate representation \(r\), it will be helpful to begin with an estimated \(z\), because an initial guess for \(z\) will allow us to regularize our search to favor values of \(r\) that are more likely to be generated by \(z\). Therefore, we shall solve the inversion problem in two steps: first we construct a neural network \(E\) that approximately inverts all of \(G\), computing estimates \(z_0 = E(x)\); and then we solve an optimization problem to identify an \(r^* \approx r_0 = g_f(\cdots(g_1(z_0)))\) that generates a reconstructed image \(G_f(r^*)\) to closely recover \(x\).

**Layer-wise Network Inversion** A deep network can be trained more easily by pretraining individual layers on smaller problems [14]. Therefore, to learn the inverting
neural network, we also proceed layer-wise. For each layer, we train a network $e_i$ to approximately invert each of the layers $g_1, \ldots, g_j$ as well as $G_f$. That is, for each layer

$$r_i = g_i(r_{i-1})$$

(4)

our goal is to learn a network $e_i$ that approximates the computation $r_{i-1} \approx e_i(r_i)$. We also want the predictions of the network to preserve the output of the layer well, so we want $r_i \approx g_i(e_i(r_i))$. We only care about the quality of the inversions of $e_i$ near the manifold of actual representations $r_i$ generated by the generator, so we train the network by sampling $z$ and then using the previous layers to compute $r_i$. Precisely, we train $e_i$ to minimize the following losses:

$$\mathcal{L}_L \equiv \mathbb{E}_z[||r_{i-1} - e_i(r_i)||_1]$$

$$\mathcal{L}_R \equiv \mathbb{E}_z[||r_i - g_i(e_i(r_i))||_1]$$

$$e_i = \arg \min_e \mathcal{L}_L + \lambda_R \mathcal{L}_R,$$

(5)

where $r_{i-1} = g_{i-1}(\cdots g_1(z) \cdots)$ and $r_i = g_i(r_{i-1})$. Here $|| \cdot ||_1$ denotes an L1 loss, and we set $\lambda_R = 0.01$ to emphasize the reconstruction of $r_{i-1}$.

Once all layers are inverted, we can compose an inversion network for all of $G$:

$$E^* = e_1(e_2(\cdots (e_j(e_f(x))))))$$

(6)

The results can be further improved by jointly fine-tuning this composed network $E^*$ to invert $G$ as a whole. Denote this fine-tuned result as $E$.

Layer-wise Image Optimization  To avoid overfitting, the capacity of the network $E(x)$ is roughly balanced with the size of $G$, but the simplicity of $E$ means that it will learn only an approximate inverse: it will not be able to learn a function that inverts $G$ exactly. The network $E$ does obtain a good initial guess for $r$ that can be improved by direct optimization of image reconstruction losses.

As described at the beginning of Section 3.2, inverting all of $G$ is difficult: $G$ is non-convex, and optimizations over $z$ are quickly trapped in local minima. Therefore, after obtaining an initial guess for $z$, we turn our attention to the more relaxed optimization problem of inverting the layers $G_f$: that is, starting from $r_0 = g_0(\cdots (g_1(z_0)))$, we seek an intermediate representation $r^*$ that generates a reconstructed image $G_f(r^*)$ to closely recover $x$.

To regularize our search to favor $r$ that are close to the representations computed by the early layers of the generator, we search for $r$ that can be computed by making small perturbations of the early layers of the generator:

$$z_0 \equiv E_d(x)$$

$$r \equiv \delta_j + g_j(\cdots (\delta_2 + g_2(\delta_1 + g_1(z_0))))$$

$$r^* = \arg \min_r \left(\ell(x, G_f(r)) + \sum \lambda_i ||\delta_i||^2\right)$$

(7)

(That is, we begin with a guess $z_0$ given by the neural network, and then we learn small perturbations of each layer before the $j$th, to obtain an $r$ that reconstructs the image $x$ well.)

The hyper-parameters $\lambda_i$ determine the balance between image reconstruction loss and the regularization of $r$. We set $\lambda_i = 1$ in our experiments.

4. Results

Implementation Details. We analyze three recent models: WGAN-GP [12], Progressive GAN [17], and StyleGAN [18]. Each architecture is trained to generate LSUN bedroom images [40]. In addition, for Progressive GAN we analyze a model trained to generate LSUN church images. To segment images, we use the Unified Perceptual Parsing network [37], which labels each pixel of an image with one of 336 object classes. Segmentation statistics are computed over samples of 10,000 images.

4.1. Generated segmentation statistics

We first measure whether segmentation statistics correctly reflect the output quality models across architectures. Table 1 shows numerical FSS values as computed over image samples for several models: Progressive GAN [17] trained on LSUN churches and bedrooms [40], as well as WGAN-GP [12] and StyleGAN [18] models of bedrooms. Similar to FID, FSS is able to distinguish between higher-quality and lower-quality models, and the ordering of methods is maintained. FSS provides the additional benefit of breaking down differences in terms of interpretable components.

4.2. Identifying Dropped Modes

Figure 1 and Figure 3 show the results of applying our method to analyze the generated segmentation statistics for Progressive GAN models of churches and bedrooms. Both the histograms and the instance visualizations provide insight into the limitations of the range of the generators. The histograms reveal that the generators partially skip difficult subtasks. For example, neither model renders as many people as appear in the target distribution.

Furthermore, when we use inversion to create models of natural images that include many pixels of people or other under-represented objects (Figure 1 and Figure 3 each shows two examples on the bottom), the instance visualizations provided by inversion reveal the way in which the models fail. The gaps are not due to low-quality drawings or
MC

Figure 4. Comparing different GAN models. Both models are trained to mimic LSUN-bedrooms. The FSS histograms on the left reveal that WGAN-GP (top) deviates from the true distribution much more than StyleGAN (bottom), identifying segmentation classes that are generated too little and others that are generated too much. On the right, we show generated images that the segmentation network has identified as containing many pixels of sky and exterior building pixels. Because the components of FSS have interpretable labels, we can directly visualize and understand the failures identified by the metric.

4.3. Identifying Overgenerated Patterns

Deviations between generated and ground-truth distributions can also appear as patterns of images that are overrepresented. Generated segmentation statistics can also be used to understand this phenomenon. Figure 4 compares segmentation statistics histograms for WGAN-GP and StyleGAN. The histograms reveal not only omissions but also segmentation classes such as (exterior) ‘building’ surfaces that are generated much more than they appear in the ground truth training set. Cases of these failures can be visualized by simply examining samples of output in which the segmentation classes are present. This reveals that WGAN-GP fails by generating formless white regions that are interpreted as sky, or walls that are textured in a way that they are interpreted as exterior building walls. By visualizing StyleGAN outputs of similar segmentation classes, we can see that this more powerful architecture produces fewer artifacts like this. The overall histogram of StyleGAN also reflects a significant improvement over previous architectures.

4.4. Layer-wise Inversion vs Other Methods

We compare our layer-wise inversion method to several previous methods; we also benchmark it against ablations of key steps of the method.

The first three columns of Figure 5 compare our method to prior inversion methods. Each method is tested on a sample of 100 images in the range of $G$, where the ground truth $z$ is known, and the reconstruction of an example image is shown. An ideal inversion should be able to perfectly reconstruct $x' = x$ in this case. In addition, a reconstruction of an image that is not in the range of $G$ is shown at the bottom. There is no ground truth for inverting this image, but the qualitative comparisons are informative.

(a) Direct optimization of $z$. Smaller generators such as 5-layer DCGAN can be inverted by applying gradient descent directly on $z$ to minimize reconstruction loss of the image [21]. In column (a) we test this method on 15-layer Progressive GAN and find that neither $z$ nor $x$ can be constructed accurately. The results are visually unsatisfying.
(b): Direct learning of $E$. Another natural solution is to learn a deep network $E$ that inverts $G$ directly, without the complexity of layer-wise decomposition. Here, we learn an inversion network that has the same free parameters and architecture as the network $E$ used in our method, but we learn it all at once by directly minimizing expected reconstruction losses over generated images, rather than learning it by layers. The method does benefit from the power of a deep network to learn generalized rules, and the results are marginally better than the direct optimization of $z$. However, both qualitative and quantitative results remain poor.

(c): Optimization of $z$ after initializing with $E(x)$. This is the method used in [44]. By initializing method (a) using an initial guess from method (b), results can be improved slightly. For smaller generators, [44] found that this method performs well. However, when applied to a Progressive GAN, the reconstructions remain poor.

**Ablation experiments.** The last three columns of Figure 5 compare our method (f) to two ablations of our method.
(d): **Layer-wise network inversion only.** Instead of optimizing $r$ after the initial guess $z_0 = E(x)$, we can simply use the layer-wise-trained inversion network $E$ as the full inverse, and set $x' = G(z_0)$. This method has the advantage that it requires only a single forward pass through the inverter network $E$, so it is very fast. Reconstructions of $z$ and final images are better than the baseline methods but far short of our full method.

Nevertheless, it is notable that, despite the inaccuracy of $z_0$, the intermediate layer features are highly correlated with their true values; this method achieves 95.5% correlation versus the true $r_4$. Furthermore, the qualitative results show that when reconstructing images outside the range of $G$, this method obtains images that appear realistic despite being
Inverting \( G \) to a failure of \( G \) is to precisely identify images that are outside the range of nearly perfect reconstructions of latents and pixels. Wherever reconstruction fails, it is almost certainly due to failures of \( G \) rather than \( \mathbb{E} \)

**(e)**: Inverting \( G \) without relaxation to \( G_f \). We can improve the initial guess \( z_0 = \mathbb{E}(x) \) by directly optimizing \( z \) to minimize image reconstruction loss. This marginally improves upon \( z_0 \), however, there are still significant differences between the reconstructed images and the true images, and recovery of \( z \) remains poor. Although the qualitative results are good, the remaining error means that we cannot know if any reconstruction errors are due to failures of \( G \) to generate an image, or if those reconstruction errors are merely due to the inaccuracy of the inversion.

**(f)**: Our full method. By relaxing the problem and regularizing optimization of \( r \) rather than \( z \), our method achieves nearly perfect reconstructions of latents and pixels.

The nearly-perfect reconstructions of our method allow us to precisely identify images that are outside the range of \( G \). Wherever reconstruction fails, it is almost certainly due to a failure of \( G \) to generate the image rather than a failure of the inversion. Therefore, although these methods are trained only on images that are generated by \( G \), our ability to solve that problem to greater than 99% gives us a reliable tool for understanding the limitations of \( G \) outside its range.

### 4.5. Layer-wise Inversion Across Domains

Next, we apply the inversion tool to test the ability of generators to generate images outside their training sets. Figure 6 shows qualitative results of applying method (f) to invert and reconstruct natural photographs of different types using a Progressive GAN trained to generate LSUN bedrooms. Reconstructions of LSUN training set and LSUN holdout set are shown; these are compared to newly collected unrelated (non-bedroom) images, both indoor and outdoor. Objects that disappear from the reconstructions reveal visual concepts that cannot be represented by the model. Some indoor non-bedroom images are rendered in a bedroom style: for example, a dining room table with a white tablecloth is rendered to resemble a bed with a white bedsheet. Outdoor images are not reconstructed well.

Figure 7 shows similar qualitative results using a Progressive GAN for LSUN outdoor church images. It can be seen that some architectural styles are dropped even in cases where large-scale geometry is preserved. The same set of unrelated (non-church) images as shown in Figure 6 are shown. When using the church model, the indoor reconstructions are lower quality and are rendered to resemble outdoor scenes; the outdoor image reconstructions recover more details.

### 5. Discussion

We have proposed a way to measure and visualize mode-dropping in state-of-the-art generative models. Generated segmentation statistics can be used to compare the quality of different models and architectures, and obtain insights into the semantic differences of the output spaces of different models.

Layer-wise inversions of the generator allow us to further probe the range of the generator using natural photographs, revealing specific objects and styles that cannot be represented. By comparing labeled distributions with one another, and by comparing natural photos with imperfect reconstructions, we can identify specific objects, parts, and styles that a generator is unable to produce.

The methods we propose here constitute a first step towards analyzing and understanding the limitations of a GAN and point to further questions. Why does a GAN decide to ignore classes that are more frequent than others in the target distribution (e.g., “person” vs. “fountain” in Figure 1)? How can we encourage a GAN to learn about a concept without skewing the training set? What is the impact of architectural choices? Finding ways to exploit and repair the mode-dropping phenomena identified by our methods are questions for future work.
References


S.1. Supplemental Materials

S.1.1. Sensitivity measure for FSS

FSS is computed using sample statistics, so the estimated statistics will vary when the data is resampled. Sampling error can be reduced by using larger samples. To estimate the sampling error of our FSS measurements at the 10,000 sample size used in our paper, Figure S.1 and Table S.1 use FSS to measure the difference between two different samples of the same data set. Measurements are done for the LSUN outdoor church and the LSUN bedroom data sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>FSS vs self</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSUN outdoor church</td>
<td>2.57</td>
</tr>
<tr>
<td>LSUN bedrooms</td>
<td>5.57</td>
</tr>
</tbody>
</table>

S.1.2. Analysis of unseen classes for additional GAN models

Here we present examples of analysis of differences between generated and target semantic classes for several different Progressive GAN models. Figure S.2 shows a Progressive GAN model trained on kitchens; Figure S.3 shows a model for living rooms, Figure S.4 shows a model for dining rooms.

S.1.3. Additional qualitative results on inversion

Figure 5 in the main paper compares inversion methods quantitatively and includes only one image of each type (generated and photograph) for qualitatively comparing our inversion method with baselines and ablations. In this section we present a larger number of images comparing the methods using reconstructions of church and bedroom models. Figure S.5 shows reconstructions for several generated images as well as images from the validation set for LSUN. Figure S.6 shows the same for a bedroom model.
Figure S.2. Analysis of the differences in semantic object distribution between target and generated images for a Progressive GAN trained on LSUN kitchens. Chairs, stove exhausts, and other objects are underrepresented.

Figure S.3. Analysis of the differences in semantic object distribution between target and generated images for a Progressive GAN trained on LSUN living rooms. Some categories of furniture such as coffee tables and ottomans are omitted.

Figure S.4. Analysis of the differences in semantic object distribution between target and generated images for a Progressive GAN trained on LSUN dining rooms. Dining rooms that include kitchens have lost many details.
Figure S.5. Examples of reconstructions of both GAN-generated and holdout images, using several inversion methods. The columns are the same as in Figure 5 in the main paper. (The target image is repeated to aid comparisons.) Above are GAN-generated images, that can be reconstructed nearly perfectly. Below are natural photographs. Dropped details reveal objects and styles that cannot be rendered by the GAN.
Reconstructions of generated images from Progressive GAN trained on bedrooms
Reconstructions of natural photos from LSUN validation set

Figure S.6. Examples of reconstructions of both GAN-generated and holdout images, using several inversion methods. The columns are the same as in Figure 5 in the main paper. (The target image is repeated to aid comparisons.) At top are GAN-generated images, that can be reconstructed nearly perfectly. Below are natural photographs. Dropped details reveal objects and styles that cannot be rendered by the GAN.