Out-of-Sample Testing for GANs

Pablo Sánchez-Martín psanch@tsc.uc3m.es
University Carlos III in Madrid
Madrid Spain

Pablo M. Olmos olmos@tsc.uc3m.es
University Carlos III in Madrid
Madrid, Spain

Fernando Perez-Cruz fernando.perezcruz@sdsc.ethz.ch
Swiss Data Science Center
Zürich/ Laussane, Switzerland

Abstract

We propose a new method to evaluate GANs, namely EvalGAN. EvalGAN relies on a test set to directly measure the reconstruction quality in the original sample space (no auxiliary networks are necessary), and it also computes the (log)likelihood for the reconstructed samples in the test set. Further, EvalGAN is agnostic to the GAN algorithm and the dataset. We decided to test it on three state-of-the-art GANs over the well-known CIFAR-10 and CelebA datasets.

1. Introduction

Implicit generative modeling, in general, and Generative Adversarial Networks (GANs), in particular, promise to solve the universal simulator problem in an end-to-end fashion (Goodfellow et al., 2014a; Kingma and Welling, 2014; Mohamed and Lakshminarayanan, 2016). GANs have been successfully applied to a variety of tasks, such as image-to-image translation (Isola et al., 2017), image super-resolution (Ledig et al., 2017), image in-painting (Pathak et al., 2016), domain adaptation (Zhu et al., 2017), text-to-image synthesis (Zhang et al., 2017), dark matter estimation (Rodriguez et al., 2018), and breaking federated learning systems (Hitaj et al., 2017), among many others.

Progress in GANs has been quite remarkable and fast in the past four years. Most of the work has concentrated on improving its training to make it more stable, robust and generalizable to numerous architectures and datasets (Nowozin et al., 2016; Gulrajani et al., 2017; Arjovsky et al., 2017; Li et al., 2017; Miyato et al., 2018) to name a few. There has also been significant progress on theoretical aspects of GAN convergence to the underlying density (Mescheder et al., 2017; Tolstikhin et al., 2017; Arora et al., 2017; Liu et al., 2017), and on their quantitative evaluation (Lucic et al., 2018; Borji, 2018; Sajjadi et al., 2018). This is the topic that occupies us on this paper.

Generating realistic looking natural images is a challenging unsolved problem and it has the advantage that it can be visually demonstrated (i.e. look at the pictures that I can generate), which explains why GANs research has zeroed in their generation. But, in order to evaluate quantitatively if the images generated by any GAN have the same properties

than the images from our training set, we have moved to Inception-based metrics: Inception Score (Salimans et al., 2016), Fréchet Inception Distance (Heusel et al., 2017) or Precision and Recall for Distributions (Sajjadi et al., 2018), which can only be used for evaluating natural images and limits the evaluation of GANs for other problems, in which there might not be a general accepted tool like Inception (Szegedy et al., 2017) to evaluate the quality of the generated samples. Furthermore, for natural images, Inception-based metrics are being criticized because it seems that most GAN algorithms achieve similar performance with proper hyperparameter optimization and random restarts (Lucic et al., 2018). Finally, GANs are solely validated by using iid samples from the generator network without using an out-of-sample test set because direct likelihood evaluation for that test set is not possible and, even argued, that it might not be the right metric because quality and likelihood might not be related (Theis et al., 2016).

In this paper, we argue that we should still be interested in the likelihood of test samples even when it is not correlated with image quality, because it will inform us if the samples cannot be generated at all (i.e. mode dropping). We propose a procedure to directly evaluate GANs, namely EvalGAN, using a test set, as it is customary in most machine learning algorithms, and without relying on Inception (Szegedy et al., 2017) or any other auxiliary network. EvalGAN measures two different and relevant metrics for understanding the quality of a trained GAN: reconstruction quality and marginal likelihood for the reconstructed test sample.

First, we measure how good we can reconstruct any given sample. Since GANs typically map a lower dimensional random input to higher dimensional space, there might be some reconstruction error that we want to account for, e.g. not every image might be reconstructed equally well or at all. Second, and irrespectively of the sample quality, we measure the marginal likelihood of each reconstructed sample, because it provides us with an indication of the regions in the sample space that we are over-repressing or fully ignoring. One key aspect of EvalGAN is the need to define a metric in the sample space that captures the complexity of each problem and that we can rely on to define quality and marginal likelihood for any sample.

In this paper, we are agnostic about what GAN to use. Our evaluation method is demonstrated using Wasserstein GANs (Arjovsky et al., 2017), WGAN with gradient penalties (WGAN-GP) (Gulrajani et al., 2017), and Spectral-normalized DCGANs (Miyato et al., 2018) trained over both CIFAR10 and CELEBA datasets. Our code can be accessed at https://github.com/psanch21/EvalGAN and can be used over any GAN.

2. Literature Review

Measuring GAN performance and quality is proving to be elusive, because, in high dimensional spaces, there are many ways in which the generated samples are different from true samples. When we compare samples in the original sample space those differences are more significant than the striking similarities (Lopez-Paz and Oquab, 2017; Im et al., 2018).

Given that GAN advances are driven by natural image generation and that we have a general tool for classifying them, i.e. Inception, we have settled for comparing images with it. The well-known IS (Salimans et al., 2016) and FID (Heusel et al., 2017) are the prime example for this evaluation trend. Recently, to improve on FID, (Sajjadi et al., 2018)

proposes two metrics that resemble precision and recall for understanding how good the generated samples cover the training samples and vice versa, allowing to understand the different failure modes of GANs. Also, in (Jitkrittum et al., 2018), the authors have proposed a goodness-of-fit that inform us in linear time about the regions in which each GAN might perform best. Even when both of these procedures are explained in general terms, they are tested on features from the last pooling layer from Inception, as for FID. The main criticism for these metrics is the need for Inception, as it is unclear how such a solution can be extended to GANs for other samples spaces.

EvalGAN first computes the noise input that generates the GAN sample with the lowest distortion w.r.t. the original image, leading to a direct comparison between the test image and its best GAN reconstruction. This reconstruction has been previously applied to explore the visual manifold of GANs in (Zhu et al., 2016) and briefly introduced in the experimental section of (Metz et al., 2017) for illustrating their GAN performance for a few training examples. However, those authors do not advocate for this error measure to be used as the main tool for evaluating GANs. On the contrary, we see this measure as the central measure to understand the quality of the samples being generated by the GAN.

Finally, (Wu et al., 2017) proposes to used Annealed importance sampling to compute a lower bound to log-likelihood of a test set and showed it was two-orders of magnitude better than KDE. The authors only use low dimensional noise input and test with MNIST. They assume the reconstruction error does not affect the likelihood of the generated samples and they do not noticed that for more challenging datasets and higher dimensional input spaces, the generated test samples would lie outside the typical set for the given input noise distribution. Hence, their estimated likelihood would be biased by the sample's reconstruction quality. In this paper we measure both of them independently.

3. EvalGAN

To illustrate the two different types of evaluations that we want to address with EvalGAN and why they are both different and relevant, we show a cartoon representation in Figure 1. For this cartoon, we assume the input to the GAN is a one-dimensional uniform distribution between 0 and 1 and the output is a two-dimensional vector. In this example and throughout the paper, we take $\mathbf{z}$ to be input noise to the generative deep neural network $G(\cdot)$ and $\mathbf{x}$ denotes the output space.

The five triangles in the plot represent five test samples and the continuous line represents the manifold of all the points in the 2D space that the GAN can produce. This line is divided in 10 segments (note that one of them, the green dot, has a point mass of 0.1) and each one of them has equal probability. If we assume a Euclidean metric is valid for the 2D space, we can easily see that the points in the longer segments are less probable than those in the shorter segments.

Note that the cyan and purple test sample are reconstructed with very low error, and the cyan triangle has higher probability than the purple triangle, because it lies on a shorter segment. The orange triangle is generated with some non-negligible error (represented by the dotted line), but its reconstruction is generated 10% of the time. The red triangle represents a sample that it is reconstructed poorly and with low probability.

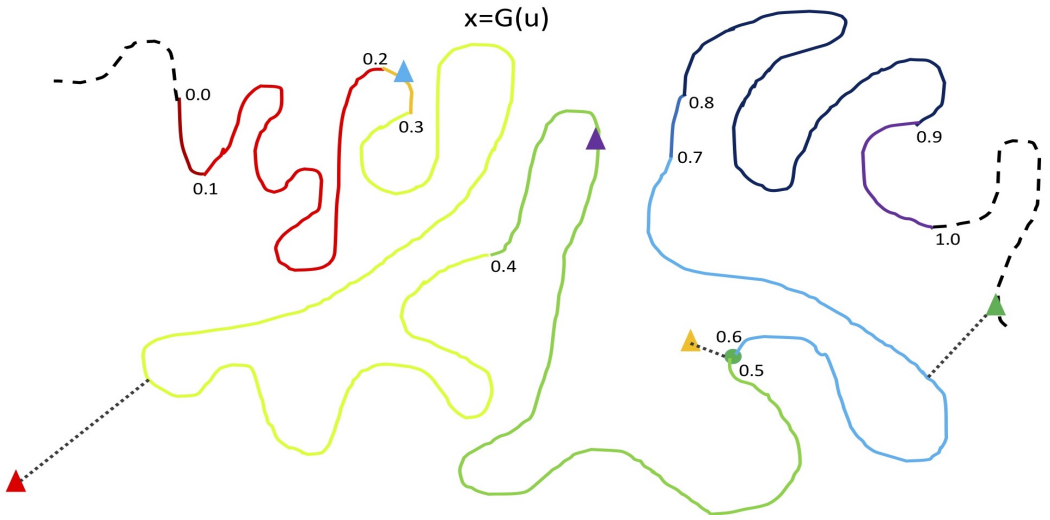


Figure 1: In this figure we show a cartoon that illustrates the need to evaluate GANs in two dimensions: quality and probability of being sampled. Details about the image meaning are described in the motivation of Section 3.

Finally, we have extended the manifold for values less than 0 and greater than 1 with dashed lines. During training we are not generating samples from this part of the manifold, so we are not controlling what points on the manifold they express but nevertheless we could generate those samples by changing the input distribution. The green triangle shows a sample that can be reconstructed with low error if we do not limit the input z to be between 0 and 1, but presents a high error otherwise. Even though this might look like a fringe example, our results demonstrate that we see this case repeatedly in practice.

3.1 EvalGAN: reconstruction quality

Given a test set sample, $\mathbf{x}_{\text{test}}$, we find the best approximation the GAN can generate by solving the following optimization problem:

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} d\left(\mathbf{x}_{\text{test}}, G(\mathbf{z})\right), \tag{1}$$

where $\mathbf{z}^*$ represents the input noise to the GAN to generate $\mathbf{x}^* = G(\mathbf{z}^*)$ as the sample that it is closest to $\mathbf{x}_{\text{test}}$, as defined by the suitable metric $d(\cdot, \cdot)$. The solution to this problem can be easily found by standard back-propagation, as it is done for generating adversarial training examples (Goodfellow et al., 2014b; Szegedy et al., 2015).

We have found that when solving (1) the values of $\mathbf{z}^*$ end up being far from the examples that can be generated by the input distribution[1]. For example, if $\mathbf{z}$ is uniformly distributed the values of $\mathbf{z}^*$ found after solving (1) are outside the valid range. If $\mathbf{z}$ is a zero-mean unit-covariance Gaussian, the squared norm of $\mathbf{z}$ tends to be much larger than the dimension of $\mathbf{z}$,

1. This issue was not reported in (Zhu et al., 2016; Metz et al., 2017), where this optimization was previously proposed.

i.e. the values of $\mathbf{z}^*$ are (far) outside the typical set for a (high-dimensional) Gaussian (Cover and Thomas, 1991). Furthermore, these deviations are more significant as the dimension of $\mathbf{z}$ increases. Hence, we also propose solving the following constraint optimization problem:

$$\mathbf{z}_c^* = \arg\min_{\mathbf{z}} d\left(\mathbf{x}_{\text{test}}, G(\mathbf{z})\right) \quad \text{s.t. } ||\mathbf{z}||^2 \leq \dim(\mathbf{z}) + \delta, \tag{2}$$

when $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and we denote $\mathbf{x}_c^* = G(\mathbf{z}_c^*)$. In our experiments, we set δ to zero because most $\mathbf{z}_c^*$ tend to be in the upper bound ($||\mathbf{z}||^2 = \dim(\mathbf{z})$) and for high-dimensional input spaces it should not matter, as the norm of any randomly generated sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ concentrates around $\sqrt{\dim(\mathbf{z})}$ (Cover and Thomas, 1991). For uniformly distributed $\mathbf{z}$, the necessary constraints are straightforward. In the experimental section, we show examples when the optimization is carried out with and without constraints and for some GANs and some samples the difference are quite significant.

3.2 EvalGAN: marginal likelihood

The likelihood of the test samples can be computed as follows:

$$p(\mathbf{x}_{\text{test}}) = \int p(\mathbf{x}_{\text{test}}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}, \tag{3}$$

In (Wu et al., 2017), the authors proposed an isotropic Gaussian likelihood for GANs, i.e:

$$p(\mathbf{x}_{\text{test}}|\mathbf{z}) \approx \frac{1}{(2\pi\sigma^2)^{\dim(\mathbf{z})}} \exp\left(\frac{||\mathbf{x}_{\text{test}} - G(\mathbf{z})||^2}{2\sigma^2}\right). \tag{4}$$

They solved the integral in (3) by annealed importance sampling. This is a fine choice if all samples in the test set could be matched to a $\mathbf{z}$ (i.e. there exist a $\mathbf{z}_{\text{test}}$ for which $\mathbf{x}_{\text{test}} = G(\mathbf{z}_{\text{test}})$) or the reconstruction error is similar (and small) for all test samples. But when the reconstruction can be uneven, best reconstructed images would seem more likely, which does not need to be the case, and setting the value of σ would be extremely hard.

This effect can be easily appreciated in the cartoon example in Figure 1, as a small σ would lead to the orange and red triangles presenting negligible likelihoods compared to the cyan and purple triangles, while a large σ would boost the likelihood of the orange triangle, because it is close to highly probable z. The value of σ would significantly affect the measured likelihood of the samples in ways that does not illustrate the quality or likelihood of any GAN.

In the previous subsection, we advocated for computing the quality of the reconstruction independently on how likely they could be generated. In this section, we now compute the likelihood of this reconstruction by counting all the $\mathbf{z}$ that can generate the same reconstruction with a negligible error:

$$p(\mathbf{x}_{\text{test}}) \approx \int_{d(\mathbf{x}_c^*, G(\mathbf{z})) < T} p(\mathbf{z}) d\mathbf{z} \approx \frac{1}{N} \sum_i \mathbb{I}_{[d(\mathbf{x}_c^*, G(\mathbf{z}_i)) < T]} \tag{5}$$

where T is a threshold to ensure that $G(\mathbf{z}_i)$ is close enough to $\mathbf{x}_c^*$, $\mathbf{z}_i$ are iid samples from $p(\mathbf{z})$, and $\mathbb{I}_{[d(\mathbf{x}_c^*, G(\mathbf{z}_i)) < T]}$ is an indicator function that it is one if the condition holds and zero otherwise. We can (and should) set T to be significantly smaller than $d(\mathbf{x}_{\text{test}}, \mathbf{x}_c^*)$, which

is the error of the best reconstruction of the test sample [2]. In this case, $\mathbf{z}_c^*$ generates $\mathbf{x}_c^*$ and we have decoupled measuring the reconstruction quality and how likely the generated sample can be.

We could also use $\mathbf{x}^* = G(\mathbf{z}^*)$ instead, where $\mathbf{z}^*$ is the solution to (1), but we show in the experimental section that those samples would not be generated when sampling from $p(\mathbf{z})$. The likelihood of $\mathbf{x}^*$ would be negligible compared to the likelihood of $\mathbf{x}_c^*$. When $\mathbf{x}^*$ is a better reconstruction than $\mathbf{x}_c^*$ emphasizes the need for separating both measures (quality and likelihood), because even if we could recover $\mathbf{x}^*$ by backpropagation, it would never be generated by sampling from $p(\mathbf{x})$. This also remarks that setting σ in (4) would be challenging, while setting T in our case is fairly straightforward.

Of course, for typical GANs, in which the dimension of $\mathbf{z}$ is the hundreds, the approximation in (5) is impractical at best. We now present three approximations that can be easily computed. We advocate for the last one, as it is the most computationally efficient and accurate of the three.

Isotropic samples. We can approximate the log likelihood as follows:

$$\log p(\mathbf{x}_{\text{test}}) \approx \dim(\mathbf{z}) \log \bar{\sigma}_\epsilon - \log Z, \tag{6}$$

where

$$\bar{\sigma}_\epsilon = \arg\max_{\sigma_\epsilon} \left(\frac{1}{N} \sum_{i=1}^{N} d(\mathbf{x}_c^*, \mathbf{x}_i^*) \right) \leq T, \tag{7}$$

$\mathbf{x}_i^* = G(\mathbf{z}_c^* + \epsilon_i)$, and $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2 \mathbf{I})$. The partition function Z only depends on $p(\mathbf{z})$ and it is independent of the GAN, because by construction all $p(\mathbf{z}_c^*)$ have the same probability.

If the curvature of $G(\cdot)$ changes considerably in different dimensions of $\mathbf{z}$ the previous measure benefits those samples that are in a more isotropic region of $G(\cdot)$, because it underestimates the probability of those sample in which $G(\cdot)$ changes significantly in different directions.

Non-isotropic samples. We can adapt the previous measure to account for differences in the curvature of $G(\cdot)$, by instead computing:

$$p(\mathbf{x}_{\text{test}}) \propto N_c^*/N, \tag{8}$$

where

$$N_c^* = \sum_i \mathbb{I}_{[d(\mathbf{x}_c^*, \mathbf{x}_i^*) < T]}, \tag{9}$$

and N_c^* counts how many $\mathbf{x}_i^*$ samples are sufficiently close to $\mathbf{x}_c^*$, when σ_ϵ is small and fixed.

Selecting a good σ_ϵ to ensure that N_c^* is nonzero for a given N and for all the test samples can be hard (and require a very large N), if the marginal likelihood for all the test samples vary substantially (which they do).

2. For a Euclidean metric our approximation is equivalent to changing $\mathbf{x}_{\text{test}}$ by $\mathbf{x}_c^*$ in the righthand side of (4).

Proposed measure. Finally, by combining the previous two approximations we get:

$$\log p(\mathbf{x}_{\text{test}}) \approx \log \frac{N_c^*}{N} + \dim(\mathbf{z}) \log \sigma_\epsilon - \log Z. \tag{10}$$

This approximation becomes more accurate as we increase σ_ϵ, because we are able to capture all the directions in $\mathbf{z}$-space in which the samples $\mathbf{x}_i^*$ are close enough to $\mathbf{x}_c^*$. This approximation can be computed accurately by gradually increasing σ_ϵ and N. In our simulations, we set the maximum N to 10,000 and we stop increasing σ_ϵ when N_c^* drops below 100.

3.3 EvalGAN: metric

One of the aspects that we have not investigated in this paper is the selection of the ideal metric, i.e. $d(\cdot, \cdot)$. Defining this metric correctly is crucial for EvalGAN to succeed at evaluating GANs and it should be carefully selected by each different problem. The different communities using GANs for creating universal simulators, should coalesce around the relevant metric for evaluating their GANs with EvalGAN.

In this paper, we illustrate three different GANs by generating natural images (CIFAR-10 and CelebA) and we have used the well-know Peak Signal-to-Noise Ratio (PSNR) typically used in image compression:

$$\text{PSNR}(\mathbf{x}_i, \mathbf{x}_j) = 10 \log_{10} \left(\frac{M^2}{\text{MSE}(\mathbf{x}_i, \mathbf{x}_j)} \right),$$

where M is the maximum possible pixel value of the images, i.e. 255 for 8-bit color images. The higher the PSNR (in dB) leads to higher image quality. The Mean Squared Error (MSE) of color images can be computed as follows:

$$\text{MSE}(\mathbf{x}_i, \mathbf{x}_j) = \frac{||\mathbf{x}_i - \mathbf{x}_j||_2^2}{3K}, \tag{11}$$

where K is the number of pixels in the images.

In this paper, we have opted for a simple metric. We understand that other metrics for images in which smoothness or other properties of the generated images are captured might be more relevant. We are not specially advocating for PSNR, except that it relates to image quality and it is easy to understand and compute.

4. EvalGAN in practice

4.1 Experimental Setup

Three different state-of-the-art GANs have been considered: Wasserstein GAN (WGAN) (Arjovsky et al., 2017), Improved WGAN with gradient penalty (WGAN-GP) (Gulrajani et al., 2017) and deep convolutional GAN with spectral normalization (SN-DCGAN) (Miyato et al., 2018). Tensorflow implementation for the three of them are publicly available. To facilitate reproducibility of our results, in the Appendix we provide an exhaustive description of the parameters selected to construct both the generator and discriminator networks and those regarding the training process. To train all models, we consider as input a Gaussian

noise model: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, with $\dim(\mathbf{z}) \in [16, 32, 64, 128, 256]$. To solve the optimization problem in (1) and (2), we use Adam algorithm (Kingma and Ba, 2014) with parameters $\alpha = 0.005$ (learning rate), $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a stopping tolerance of 0.1 in 3000 iterations. For solving (2), we project the norm of $\mathbf{z}$ to the unit hypersphere if the norm of $\mathbf{z}$ is larger than $\sqrt{\dim(\mathbf{z})}$.

CIFAR10 is taken as the main running example in this section to illustrate our discussion and the quality metrics proposed. CIFAR10 contains 50,000 images for training and 10,000 images for test. Further experiments using the CelebA dataset are mainly included in the Appendix. In CelebA, 2,000 face images are used for test and 200,000 images for training. The results in this section refers to the SN-DCGAN algorithm, while WGAN and WGAN-GP are reported in the Appendix.

4.2 Assessing reconstruction quality in EvalGAN

We first analyze the influence of the generator input-dimension on the GAN reconstruction quality. We compute $\mathbf{z}^*$ for the images in the test set using the solution to the unconstraint problem in (1), once the GAN has been trained. The solid lines in Figure 2 show the evolution of the average PSNR with respect to $\dim(\mathbf{z})$, as expected the image quality improves with $\dim(\mathbf{z})$. Also, it is remarkable that the reconstruction quality of test samples is as good as those in the training set.

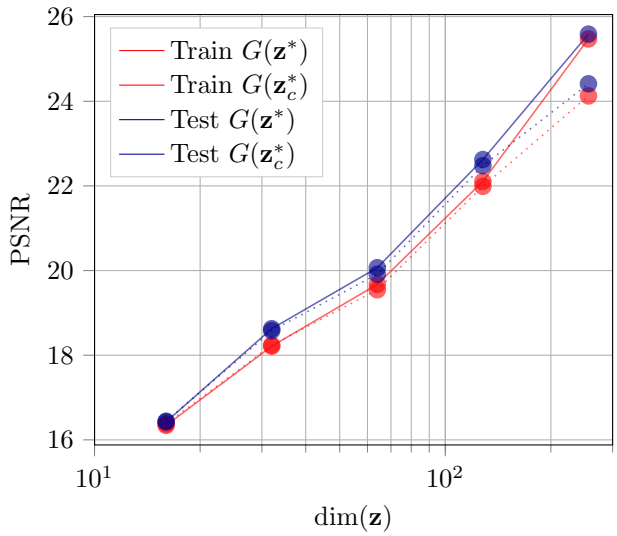


Figure 2: Evolution of the average PSNR between the original image and its reconstruction with the dimension of the latent space for SN-DCGAN trained using CIFAR10. Solid lines correspond to the reconstruction PSNR using the unconstrained projection found solving (1). Dashed lines correspond to the reconstruction PSNR using the constrained optimization in (2) with $\delta = 0$.

We also found that (almost) all $\mathbf{z}^*$ samples lie outside the typical set and hence the found images would never be generated when sampling from $p(\mathbf{z})$. This effect has not been previously reported in the literature and it shows that during the optimization of the GAN we are not controlling accurately the mapping from $\mathbf{z}$ to $\mathbf{x}$. This issue is illustrated in Figure 3(a), where we show the average $\log p(\mathbf{z})$ for the test and training samples and we compare it with the $\log p(\mathbf{z})$ of the samples from the typical set. In Figure 3(b), we show the histogram for $||\mathbf{z}^*||^2$ from the training and test samples, as well as the histogram of samples from $p(\mathbf{z})$ for $\dim(\mathbf{z}) = 256$. It is fairly obvious the values of $\mathbf{z}^*$ would never be sampled in practice.

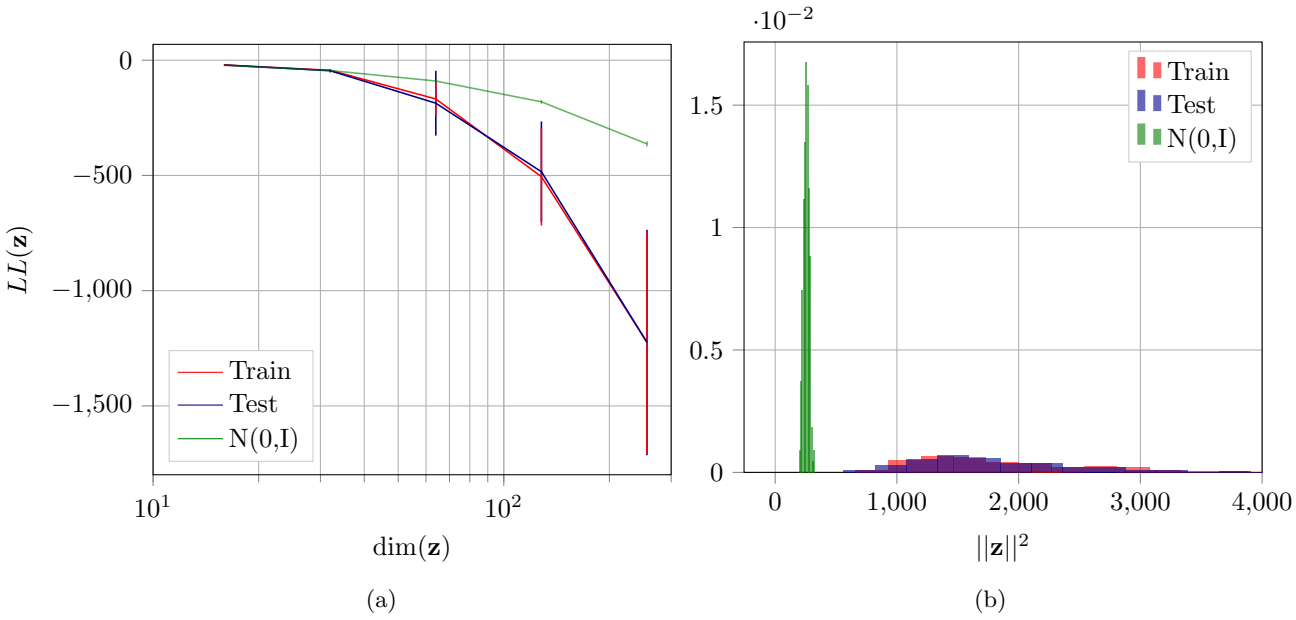
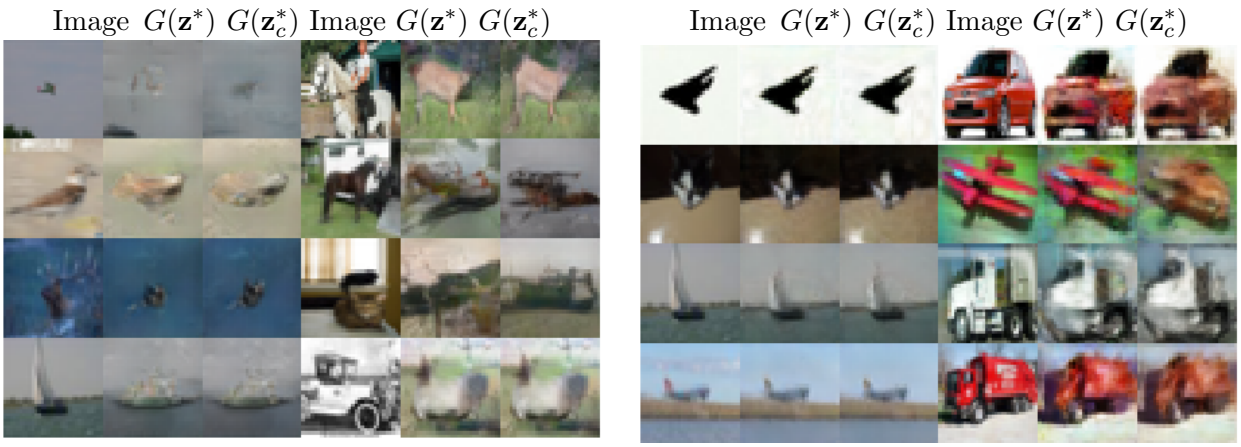


Figure 3: In (a), evolution of the average log-likelihood of the $\mathbf{z}^*$ solutions of the unconstrained problem in (1) computed over both the training and the test set for the SN-DCGAN trained with the CIFAR10 dataset. In (b), we show the histogram of $||\mathbf{z}^*||^2$ for the case $\dim(\mathbf{z}) = 256$.

As advanced in Section 3.1, we also advocate to constraint $\mathbf{z}^*$ to be in the typical set of $p(\mathbf{z})$. The dashed lines in Figure 2 represent the PSNR of the original image w.r.t. $G(\mathbf{z}_c^*)$, where $\mathbf{z}_c^*$ is found by solving (2). There is a noticeable degradation for high-dimension inputs in both train and test sets. In Figure 4 we show some test set examples reconstructed with $\mathbf{z}^*$ and $\mathbf{z}_c^*$. In (a) we use $\dim(\mathbf{z}) = 256$ and in (b) $\dim(\mathbf{z}) = 16$. In the lefthand side of each subplot, we report the images with largest PSNR and, in the righthand side, we show the images with the lowest PSNR values. For the high quality reconstructions, there is little visual difference between the constraint and unconstraint optimization and the input dimension does not seems to affect the reconstruction that much. For the lower quality reconstructions and $\dim(\mathbf{z}) = 256$, the differences are quite significant between the three images, but still the objects are recognizable in both reconstructions. For $\dim(\mathbf{z}) = 16$ neither reconstruction is meaningful, showing larger dimensions for $\mathbf{z}$ are really needed.



(a) $\dim(\mathbf{z}) = 16$ (b) $\dim(\mathbf{z}) = 256$

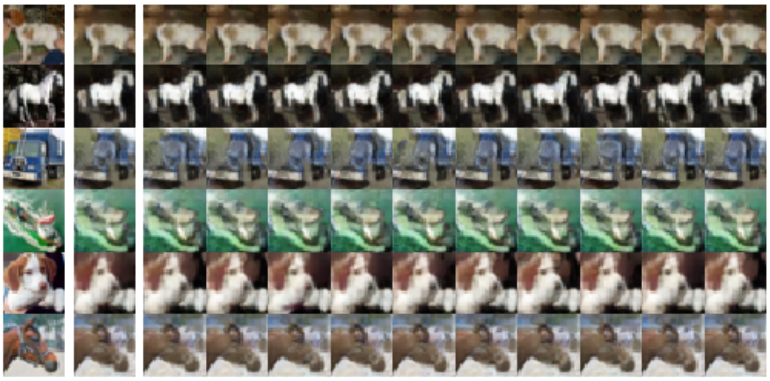
Figure 4: Each figure contains two groups with 3 columns each. The left group represents the test samples with largest PSNR, while the right group contain samples with the lowest PSNR values.

To obtain the results above, we also checked if different initializations for $\mathbf{z}$ in (2) lead to the same $\mathbf{x}_c^* = G(\mathbf{z}_c^*)$. In Figure 5 (a), we show 10 different reconstruction for the same image from 10 different initializations, as well as the sample from the mean input noise sample, i.e. $\mathbf{z}_{c,p}^* = \sum_m \mathbf{z}_{c,m}^*/10$, where $\mathbf{z}_{c,m}^*$ are each one of the 10 solutions to (2) with the same test image. The first column is the original image, the second column represents the image coming from $G(\mathbf{z}_{c,p}^*)$ and the last 10 columns shows each one of the individual reconstructions $G(\mathbf{z}_{c,m}^*)$. We also took the two $\mathbf{z}_{c,m}^*$ that were further apart and linearly interpolate their values to generate the images in between. These images are shown in Figure 5 (b) with similar behavior as the previous experiment. Similar conclusions can be drawn when we perform polar interpolation instead of linear interpolation. In short, even if the optimization problems are not convex and uni-modality is not enforced by GAN training, we did not find issues with either.

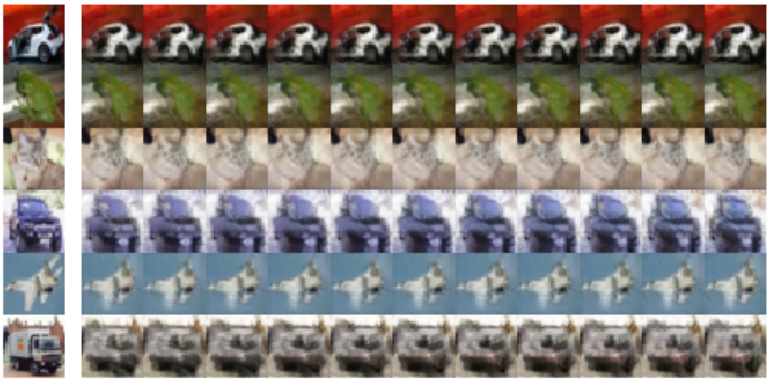
4.3 EvalGAN marginal likelihood

We now concentrate in evaluating the likelihood of the reconstructed images, independent of their reconstruction quality. First, in Figure 6 (a) we show the evolution of $d(\mathbf{x}_c^*, \mathbf{x}_i^*)$ as a function of σ_ϵ for 20 train and 20 test samples. We can see that the degradation of the samples varies considerably. For example, if we set the threshold for the PSNR at 40dB (much larger than the 25dB reconstruction error reported in Figure 2 the image with the largest σ_ϵ, for which this mean reconstruction quality is achieved, is above 0.04. For the image with lowest σ_ϵ, before the quality threshold is met, is below 0.01. This means that the most probable image in the set is at least $(0.04/0.01)^{256} \approx 10^{154}$ more probable than the least likely image and we are only comparing 20 random samples in this plot.

We now turn to computing the log-likelihood for 400 images using the approximation in (10), in Figure 6 (b) we show the histogram of $\log_{10} p(\mathbf{x}) - \log_{10} Z$. We use a threshold T



(a)

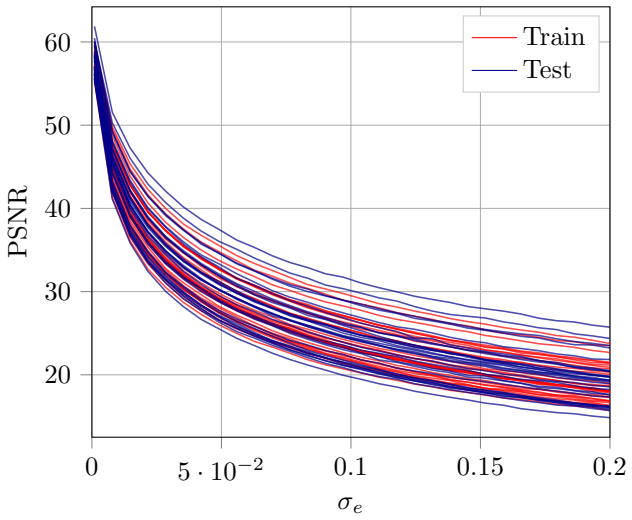


(b)

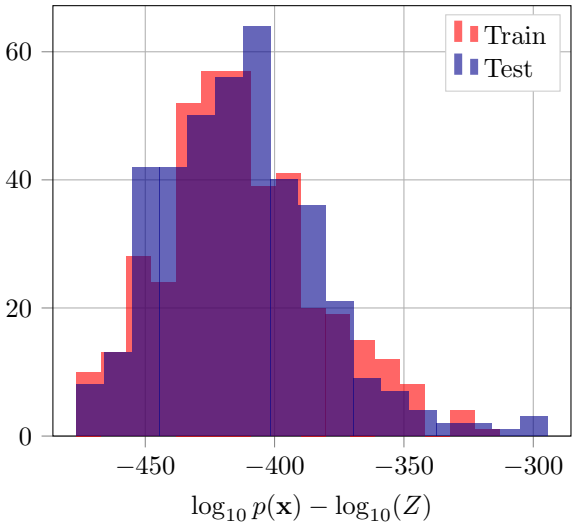
Figure 5: In (a) from left to right: real image, reconstruction using the latent mean $\sum_m \mathbf{z}^*_{c,m}/10$, and reconstruction using the solution to (2) for 10 different initializations. For this experiment we consider $\dim(\mathbf{z}) = 256$. In (b), the leftmost image is the real sample. The rest are the reconstructions from linearly interpolated $\mathbf{z}$ values using (2) for two different initializations.

in (9) corresponding to a PSNR w.r.t. to $G(\mathbf{z}^*_c)$ of 40 dB. Note that the few images in the right-most tale of the histogram are 10^{125} times more probable of being generated than those in the mode of the histogram, and are 10^{175} times more likely than those in the left tail of the histogram. Hence, at a sample level, we are able to point exactly where overrepresentation and mode dropping occurs. The log-likelihood distribution is similar for the training and test sets, it does not seem to be an over-representation of the samples in the training set.

In Figure 7, we compare the log unnormalized marginal likelihood with the reconstruction PSNR between the real image and $G(\mathbf{z}^*_c)$ for 400 test CIFAR10 images using SN-DCGAN. First, we can notice that the dynamic range of both likelihood and PSNR is quite large, especially the former. We can also observe that images with simpler textures and large uniform backgrounds are not only reconstructed with better quality, but also they are being overrepresented by the generator network. In Figure 8, we compare the reconstruction of



(a)



(b)

Figure 6: In (a), we show the evolution of the SN-DCGAN average PSNR for 20 CIFAR10 images between $G(\mathbf{z}_c^*)$ and $G(\mathbf{z}_c^* + \epsilon_i)$ as a function of σ_ϵ, where $\epsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma_\epsilon^2 \mathbf{I})$. In (b) we show the unnormalized marginal likelihood histogram for the SN-DCGAN using (10) for 400 CIFAR10 images and a 40 dB PSNR threshold.

some of the most likely and least likely images with the original image and we can easily see this effect too. In the plots, we have added the reconstruction with the unconstraint optimization problem for completeness.

We also include the results CelebA, in which the most likely images seem to contain plain faces with soft smiling gestures, while least likely samples in the set are associated to people that either have a weird posture or they are wearing glasses or hats. It is interesting

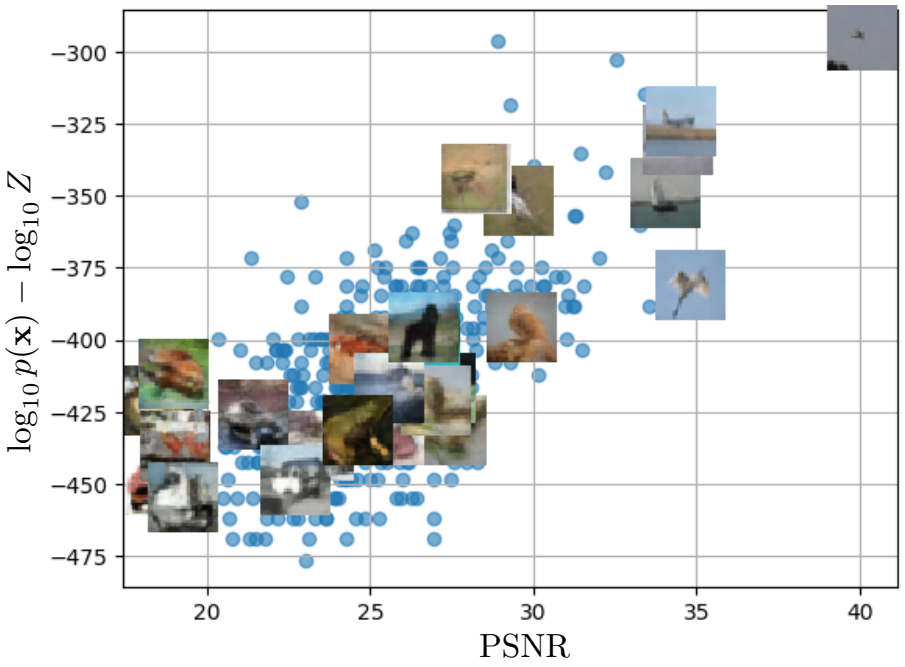


Figure 7: Scatter plot to compare the log unnormalized marginal likelihood in (10) with the PSNR between the real image and $G(\mathbf{z}_c^*)$ for 400 test CIFAR10 images using SN-DCGAN with $\dim(\mathbf{z}) = 256$.

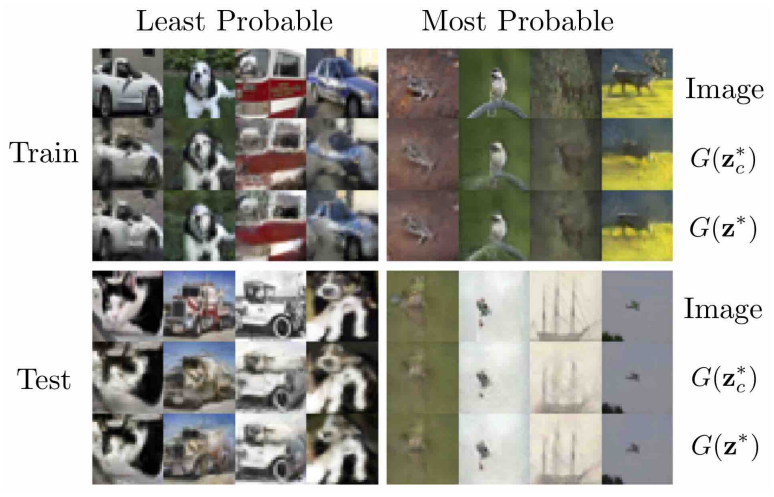
to note that in CelebA, reconstructed images using the solution to (2), i.e. $\mathbf{z}_c^*$. tend to simplify the original image including features common in the set of most probable images, e.g. inserting soft smiles instead of more complicated gestures, or even removing objects like glasses, hats, or even a microphone. The solution to the unconstrained problem in (1), i.e. $\mathbf{z}^*$, tend to partially keep those features.

Finally, in the Appendix we reproduce the previous experiments using WGANs, WGAN-GP and SN-DCGAN with CIFAR10 and CelebA datasets.

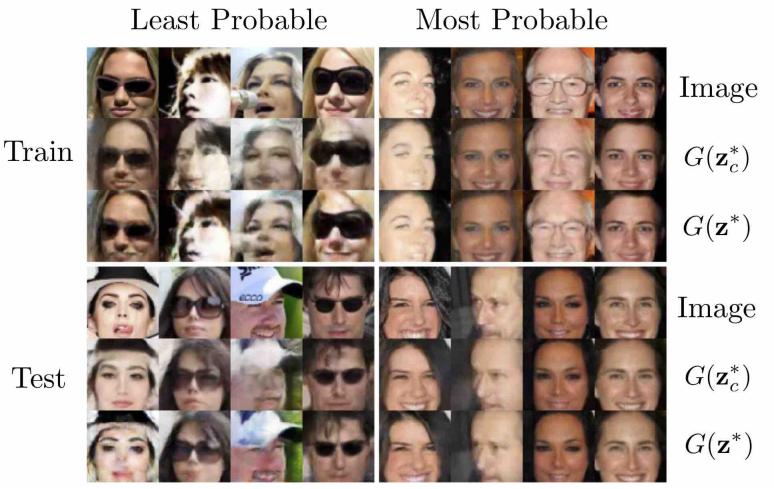
5. Discussion

The two measures that we have put forward in this paper, are very relevant when evaluating GANs and they have not been systematically used in the past. The reproduction quality tells us if a sample can be generated by the GAN and how good it matches the test sample [3]. The estimation of the log likelihood of the reconstruction (not the test sample) tell us how likely are we to see that reconstruction, which is the only image that the GAN can produce (This is a new metric proposed in this paper). Estimating the likelihood of the test sample directly is much harder and it mixes these two relevant metrics in one, making it useless to evaluate GANs, as already point it out in Theis et al. (2016).

3. This measure had been proposed previously in Zhu et al. (2016); Metz et al. (2017), but has not been advocated for systematically evaluating GANs.



(a) CIFAR10



(b) CelebA

Figure 8: In (a), we plot the most and least probable images for SN-DCGAN and CIFAR10 according to (10). In (b), we repeat the experiment for CelebA. In both cases $\dim(\mathbf{z}) = 256$.

The results in log likelihood estimation shows that training and test samples suffer significant over and under-representation issues that needs to be corrected when training GANs. We can use the mean log-likelihood to compare GANs, but we should also try to equalize the log-likelihoods for the training (and test) samples when training GANs. Because a difference in marginal likelihood of more than 10^{10} seems a bit extreme, in our opinion, and these differences happen for most pair of images (the largest difference are larger than 10^{150}).

We have also noticed that the samples that are more visually complex lead to lower reconstruction error and lower marginal likelihoods. For example, we can argue that the

samples that present lower marginal likelihood can be over-sampled when training GANs, as we should not expect that harder to generate samples need to be seen an equal number of times that those that are easier to generate. This will also improve the reconstruction quality of these samples.

In this paper, we have left open what the right metric for the different GANs would be. Is PSNR adequate or should we consider other distances for images? Also, what should be the right metric for generating text or speech? In general, for each problem, in which we want to evaluate GANs, we would need to design the right metric.

Finally, we have not been able to apply EvalGAN to Variational Autoencoders (VAE), as we had wished for. EvalGAN can be used to evaluate the decoding network of VAEs the same way we proposed to evaluate the generative networks of GANs. Additionally, EvalGAN, given a test data set, can help compare the $\mathbf{z}_c^*$ given by (2) with the $\mathbf{z}$ that is obtained from the encoding VAE network. Understanding if these two distributions are similar would tell us about how well the encoder and decoder have been trained and open a different way to further optimizing them. This has been left as further work.

5.1 The need for constraint optimization for evaluating the test samples

One of the main results from using EvalGAN is an ancillary result that we were not expecting when we embarked on this project. The values of $\mathbf{z}^*$ in (1) are well off the typical set of that would be generated from $p(\mathbf{z})$. When we constraint the result to be in the typical set the image quality degrades slightly, but still it does degrade and it is more apparent as $\dim(\mathbf{z})$ grows [4].

Expecting that the distribution of $\mathbf{z}^*$ matches that of $p(\mathbf{z})$ might be too much to ask for, because of biases in the available sets and the training of GANs and its architecture. But we should expect that $\mathbf{z}^*$ for both training and test samples should lie on the typical set of $p(\mathbf{z})$ without needing to constrain it, because otherwise we would not be controlling the samples that GANs will be generating as well as we could. We believe that GAN training should be modify to account for this problem. This is probably the most important conclusion of this study. We have not figure out a way forward (yet).

4. In the Appendix we show that this effect is less pronounced for WGAN-GP, but the samples are still outside the typical set.

Acknowledgments

The work of Pablo M. Olmos and Pablo Sánchez-Martín is supported by Spanish government MEC under grant TEC2016-78434-C3-3-R, by Comunidad de Madrid under grants IND2017/TIC-7618, IND2018/TIC-9649, and Y2018/TCS-4705, and by the European Union (FEDER). We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223, International Convention Centre, Sydney, Australia, 2017.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and Equilibrium in Generative Adversarial Nets (GANs). In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 224–232, 2017.

Ali Borji. Pros and Cons of GAN Evaluation Measures. *arXiv preprint arXiv:1802.03446*, 2018.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 1991. ISBN 0-471-06259-6.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014a.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2014b.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777. 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637. 2017.

Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the GAN: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

Daniel J. Im, He Ma, Graham Taylor, and Kristin Branson. Quantitatively Evaluating GANs With Divergences Proposed for Training. *International Conference on Learning Representations (ICLR)*, 2018.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

Wittawat Jitkrittum, Heishiro Kanagawa, Patsorn Sangkloy, James Hays, Bernhard Schölkopf, and Arthur Gretton. Informative Features for Model Comparison. In *Advances in Neural Information Processing Systems*, pages 816–827. 2018.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017.

Shuang Liu, Olivier Bousquet, and Kamalika Chaudhuri. Approximation and Convergence Properties of Generative Adversarial Learning. In *Advances in Neural Information Processing Systems*, pages 5545–5553. 2017.

David Lopez-Paz and Maxime Oquab. Revisiting Classifier Two-Sample Tests. *International Conference on Learning Representations (ICLR)*, 2017.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs Created Equal? A Large-Scale Study. In *Advances in Neural Information Processing Systems*, pages 698–707. 2018.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The Numerics of GANs. In *Advances in Neural Information Processing Systems*, pages 1825–1835. 2017.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled Generative Adversarial Networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for Generative Adversarial Networks. *International Conference on Learning Representations (ICLR)*, 2018.

Shakir Mohamed and Balaji Lakshminarayanan. Learning in Implicit Generative Models. *arXiv preprint arXiv:1610.03483*, 2016.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems*, pages 271–279. 2016.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context Encoders: Feature Learning by Inpainting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Andres C Rodriguez, Tomasz Kacprzak, Aurelien Lucchi, Adam Amara, Raphael Sgier, Janis Fluri, Thomas Hofmann, and Alexandre Réfrégier. Fast Cosmic Web Simulations with Generative Adversarial Networks. *arXiv preprint arXiv:1801.09070*, 2018.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing Generative Models via Precision and Recall. In *Advances in Neural Information Processing Systems*, pages 5234–5243. 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242. 2016.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *AAAI*, 2017.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *International Conference on Learning Representations (ICLR)*, 2016.

Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. AdaGAN: Boosting Generative Models. In *Advances in Neural Information Processing Systems*, pages 5424–5433. 2017.

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the Quantitative analysis of Decoder-Based Generative Models. *International Conference on Learning Representations (ICLR)*, 2017.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. StackGAN: Text to Photo-Realistic Image Synthesis With Stacked Generative Adversarial Networks. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative Visual Manipulation on the Natural Image Manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 597–613, 2016.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

Appendix A: Architecture of GANs

The structural parameters of both the discriminator and generator networks used to train the different GANs in our study (WGAN, WGAN-GP and SN-DCGAN) are as follows.

SNDCGAN: The discriminator is a 7 layer deep CNN with [64, 128, 128, 256, 256, 512, 512] filters each followed by a fully connected layer. We use Leaky ReLU as activation function of the intermediate layers. The generator starts with a fully connected layer followed by 4 deconvolutional layers with depths [512, 256, 128, 64]. We use batch normalization between the hidden layers and ReLU as activation function. This model is trained with the Adam optimizer with learning rate of 0.0001 and parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

WGANGP: For the discriminator, we use a CNN with 4 layers with [64, 128, 256, 512] filters each for CelebA and 3 layers with depths [128, 256, 512] for CIFAR10, followed by a single fully connected layer in both cases. We use Leaky ReLU as the activation function of the hidden layers. The generator starts with a fully connected layer and continues with a 4 layers CNN for CelebA and a 3 layer CNN for CIFAR10 with depths [512, 256, 128, 64] and [512, 256, 128] respectively. We use batch normalization in the hidden layers and ReLU as the activation function. We have used the Adam optimizer with learning rate of 0.0001 and parameters $\beta_1 = 0.5$ and $\beta_2 = 0.9$.

WGAN: The discriminator is a 4 layer CNN with depths [32, 64, 128, 256] followed by a fully connected layer. All convolutional layers use Leaky ReLU as activation function and batch normalization.The generator contains a fully connected layer followed by 4 convolutional layers with depths [256, 128, 64, 32]. We use batch normalization between the hidden layers and Leaky ReLU activation function. For training we use the RMSProp optimizer with learning rate of 0.0001.

In Figure 9 we show samples of the three GANs when trained over CIFAR10 and CelebA dataset with $\dim(\mathbf{z}) = 256$.

Appendix B: Data Reconstruction

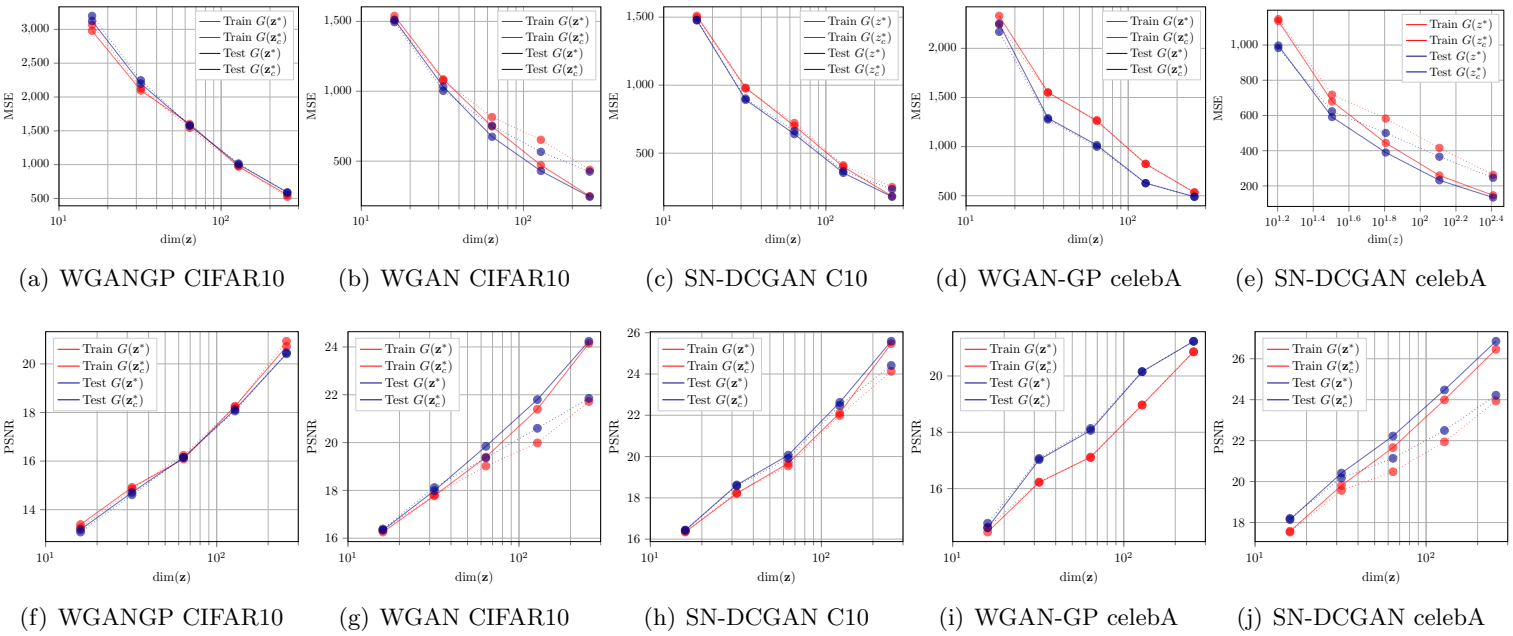
Figure 10 shows the average MSE between real test/training images and their reconstruction using $\mathbf{z}^*$ in (1) or $\mathbf{z}_c^*$ in (2), as $\dim(\mathbf{z})$ grows. SN-DCGAN stands out in terms of reconstruction error, achieving PSNR values above 26 dB for $\dim(\mathbf{z}) = 256$. In the top row of Figure 11 we show the average log-likelihood $LL(\mathbf{z}^*)$ as a function of $\dim(\mathbf{z})$. For high dimensions, in all cases it is significantly smaller than the typical LL of samples from the input distribution $p(\mathbf{z})$, indicating that the sampling from the input distribution so that the best reconstructed image is obtained is extremely unlikely. In the bottom row, we show the histogram of $||\mathbf{z}^*||^2$ for $\dim(\mathbf{z}) = 256$.

In Figure 12 we compare test images (first column) with $G(\mathbf{z}^*)$ (central column) and $G(\mathbf{z}^*)$ (right column). The left group of images represents the test samples with largest $\text{PSNR}(\mathbf{x}, G(\mathbf{z}_c^*))$ while the right group contains the samples with the worst PSNR values. The top row corresponds to $\dim(\mathbf{z}) = 256$, and the bottom row to $\dim(\mathbf{z}) = 16$. While for

(a) WGANGP (b) WGAN (c) SNDCGAN

(d) WGAN-GP (e) SNDCGAN

Figure 9: Samples drawn from WGAN, WGAN-GP and SN-DCGAN when trained over CIFAR and CelebA dataset with $\dim(\mathbf{z}) = 256$.



(a) WGANGP CIFAR10 (b) WGAN CIFAR10 (c) SN-DCGAN C10 (d) WGAN-GP celebA (e) SN-DCGAN celebA

(f) WGANGP CIFAR10 (g) WGAN CIFAR10 (h) SN-DCGAN C10 (i) WGAN-GP celebA (j) SN-DCGAN celebA

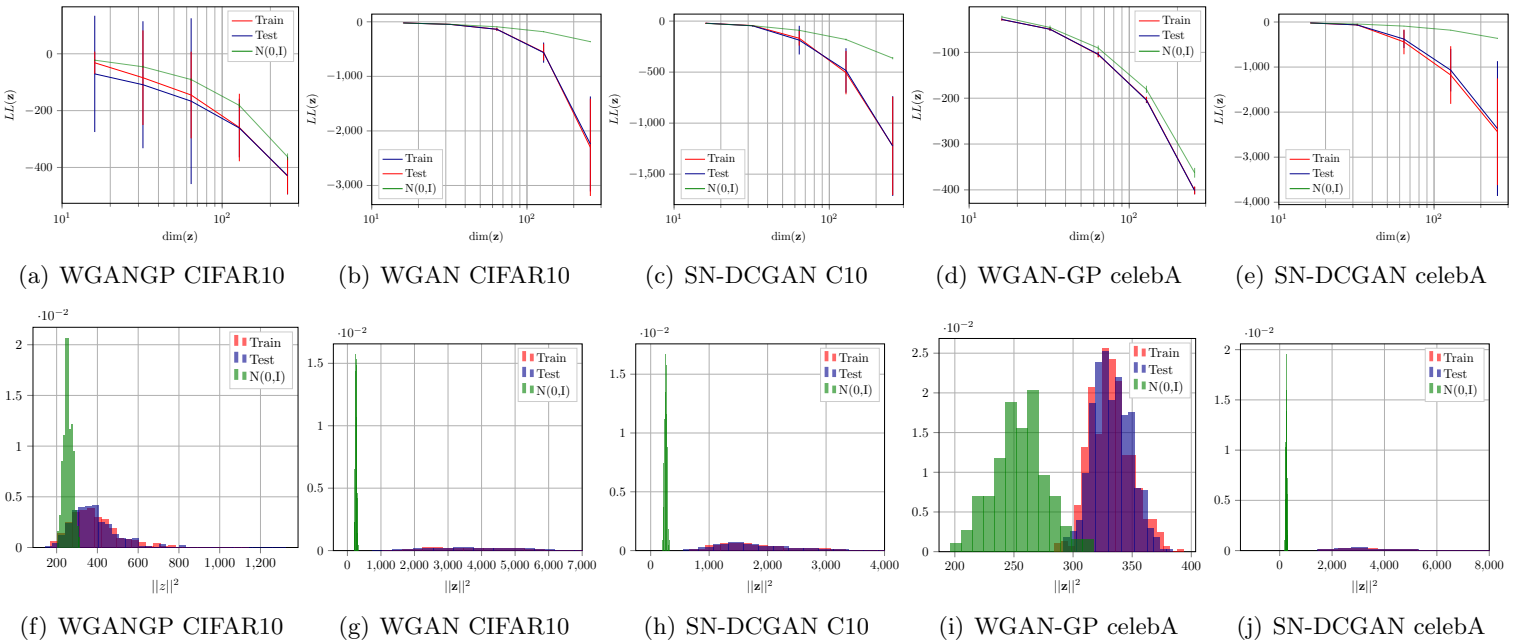
Figure 10: Average MSE and PSRN between real test/training images and their reconstruction using $\mathbf{z}^*$ in (1) or $\mathbf{z}_c^*$ in (2), as dim($\mathbf{z}$) grows.

dim($\mathbf{z}$) = 16 the reconstruction error is in general large for all images, for the high quality reconstructions in the case dim($\mathbf{z}$) = 256 there is little difference between the constraint and unconstraint optimizations, while for the lower quality reconstructions the differences are quite significant.

In Figure 13 we show the reconstructed image $G(\mathbf{z}_c^*)$ for 5 different test images using 10 different initializations. We also show the reconstruction mean input noise sample, i.e. $\mathbf{z}_{c,p}^* = \sum_m \mathbf{z}_{c,m}^*/10$, where $\mathbf{z}_{c,m}^*$, $m = 1, \ldots, 10$ are each one of the 10 solutions. In Figure 14 we also took the two $\mathbf{z}_{c,m}^*$ that were further apart and linearly interpolate their values to generate the images in between. These images are shown in with similar behavior as the experiment in Figure 13. Similar conclusions can be drawn when we perform polar interpolation instead of linear interpolation. In short, even if the optimization problems are not convex and uni-modality is not enforced by GAN training, we did not find issues with either.

Appendix C: EvalGAN sample marginal likelihood

The proposed metric to estimate the marginal likelihood of generating a given sample $p(\mathbf{x}_{\text{test}}) \propto \frac{N_c^*}{N} \bar{\sigma}_\epsilon^{\dim(\mathbf{z})}$ is based on evaluating the distortion between the generator output with inputs $\mathbf{z}_c^*$ and $\mathbf{z}_c^*$ corrupted by additive Gaussian noise of a certain variance σ_ϵ^2. For 20 test and train images, in Figure 15 we plot the evolution of the average PSNR between $G(\mathbf{z}_c^*)$ and $G(\mathbf{z}_c^* + \epsilon_i)$ as σ_ϵ grows. In all cases dim($\mathbf{z}$) = 256. Observe that there exists a significant variability in the degradation that each image suffers as samples are further apart from $\mathbf{z}_c^*$.



(a) WGANGP CIFAR10 (b) WGAN CIFAR10 (c) SN-DCGAN C10 (d) WGAN-GP celebA (e) SN-DCGAN celebA

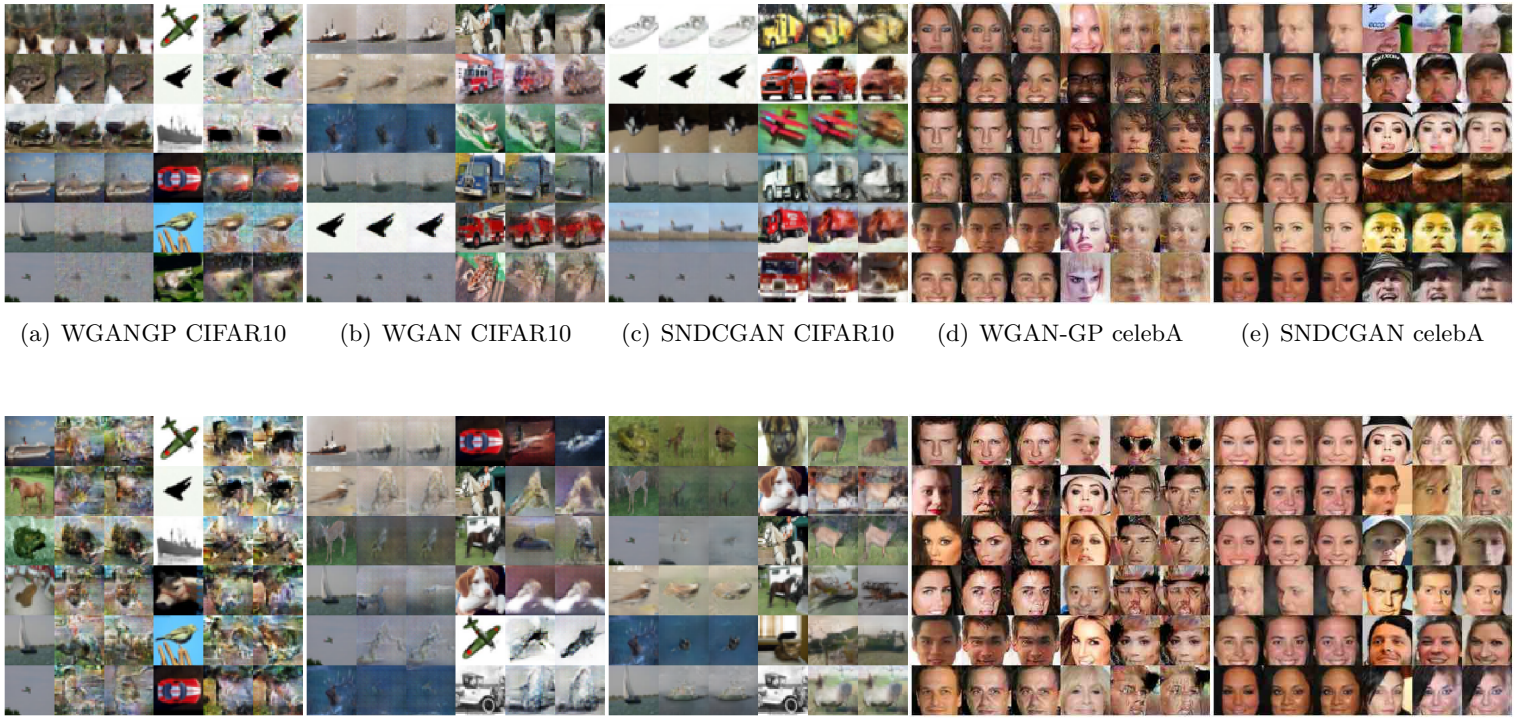
(f) WGANGP CIFAR10 (g) WGAN CIFAR10 (h) SN-DCGAN C10 (i) WGAN-GP celebA (j) SN-DCGAN celebA

Figure 11: In the top row we show the average log-likelihood $LL(\mathbf{z}^*)$ as a function of $\dim(\mathbf{z})$. In the bottom row, we show the histogram of $||\mathbf{z}^*||^2$ for $\dim(\mathbf{z}) = 256$.

This is better illustrated in Figure 16, where in the top row we show the histogram of the maximum value of σ_ϵ for which the average PSNR w.r.t. to $G(\mathbf{z}_c^*)$ is less than 40 dB. In the bottom row, we reproduce this experiment for a maximum PSNR value of 30 dB. In all cases we have used 400 test/train images. For the same set of images, in Figure 5.1 we show the unnormalized log marginal likelihood histogram using the. In all cases, results indicate an extreme overrepresentation of some samples in the test set, which corresponds to simple images with smooth textures and uniform backgrounds in CIFAR10 and plain smiling faces in SN-DCGAN, as it can be observed in Figure 18. It is interesting to note that, particularly for SN-DCGAN with CelebA, reconstructed images using the solution $\mathbf{z}_c^*$ to the constrained problem tend to simplify the original image including features common in the set of most probable images, e.g. inserting soft smiles instead of more complicated gestures, or even removing the glasses. This effect is less severe when we visualize the reconstructed image from the solution $\mathbf{z}^*$ to the unconstraint problem. Figure 19 shows scatter plots comparing the PSNR w.r.t. the original image versus the estimated log marginal likelihood obtained using EvalGAN. Observe that simpler images tend to be in regions with higher marginal likelihoods and better reconstructions, according to PSNR. We believe this effect must be certainly introducing a bias during the training of the GANs, as we sample minibatches of images from the generator at every training step.

In Figure 20 we show a comparison of different GANs using EvalGAN. SN-DCGAN performs better than WGAN-GP both in terms of reconstruction capabilities and in sample

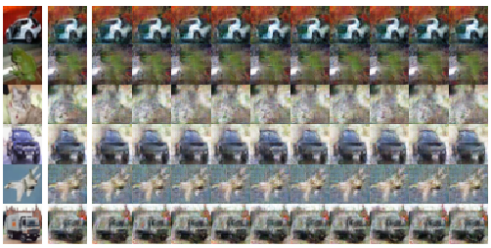
marginal likelihood. Also, WGAN on CIFAR10 provides much higher marginal likelihoods than SN-DCGAN, at the cost of worse average PSRN reconstruction quality.



(a) WGANGP CIFAR10 (b) WGAN CIFAR10 (c) SNDCGAN CIFAR10 (d) WGAN-GP celebA (e) SNDCGAN celebA

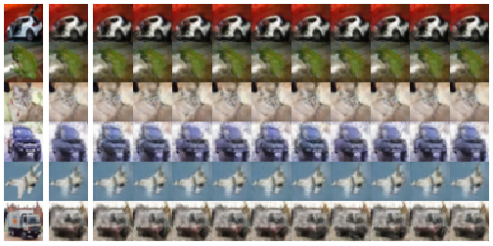
(f) WGANGP CIFAR10 (g) WGAN CIFAR10 (h) SNDCGAN CIFAR10 (i) WGAN-GP celebA (j) SNDCGAN celebA

Figure 12: We compare test images (first column) with $G(\mathbf{z}^*)$ (central column) and $G(\mathbf{z}^*)$ (right column). The left group of images represent the test samples with largest $\text{PSNR}(\mathbf{x}, G(\mathbf{z}_c^*))$ while the right group contains the samples with the worst PSNR values. The top row corresponds to $\dim(\mathbf{z}) = 256$, and the bottom row to $\dim(\mathbf{z}) = 16$.

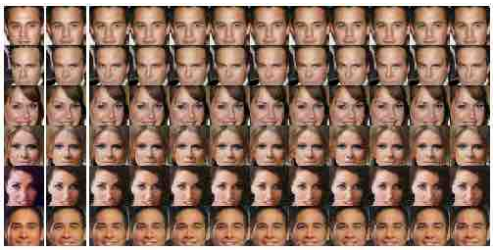


(a) WGANGP CIFAR10

(b) WGAN CIFAR10



(c) SNDCGAN CIFAR10



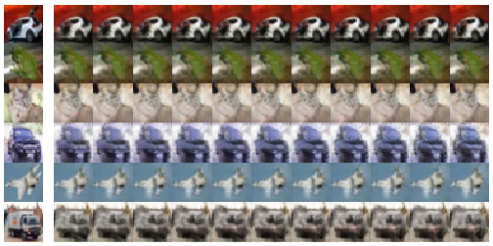
(d) WGAN-GP celebA

(e) SNDCGAN celebA

Figure 13: Reconstructed image $G(\mathbf{z}_c^*)$ for 5 different test images using 10 different initalizations. Left most column is the original image. In the second column we also show the reconstruction mean input noise sample, i.e. $\mathbf{z}_{c,p}^* = \sum_m \mathbf{z}_{c,m}^*/10$, where $\mathbf{z}_{c,m}^*$, $m = 1, \ldots, 10$ are each one of the 10 solutions.

(a) WGANGP CIFAR10

(b) WGAN CIFAR10

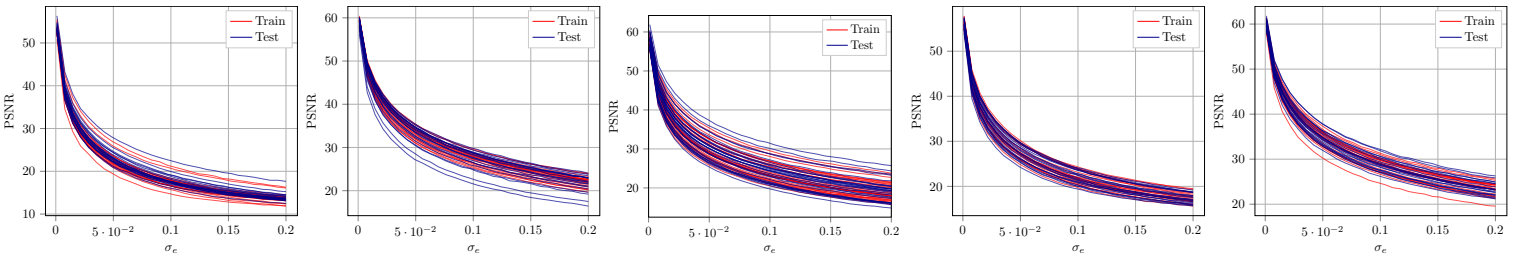


(c) SNDCGAN CIFAR10

(d) WGAN-GP celebA

(e) SNDCGAN celebA

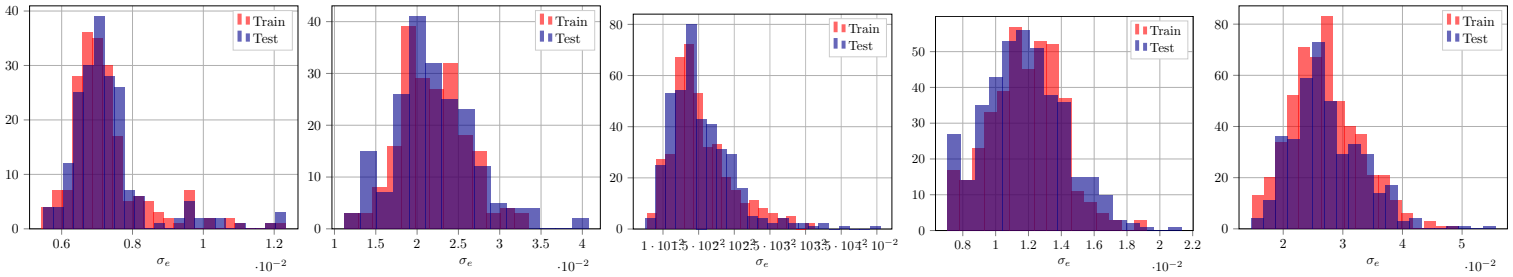
Figure 14: Reconstruction from linearly interpolated noise samples using the two noise samples $\mathbf{z}^*_{c,m}$ that are further apart among those found for 10 different initalizations of the constrained problem in (2). The left most column is the original image.



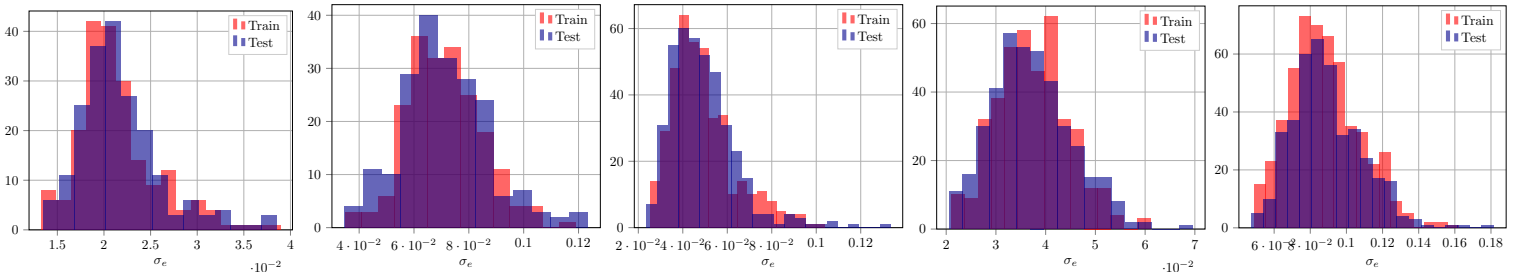
(a) WGANGP CIFAR10 (b) WGAN CIFAR10 (c) SNDCGAN CIFAR10 (d) WGAN-GP celebA (e) SNDCGAN celebA

Figure 15: Evolution of the average PSNR between $G(\mathbf{z}^*_c)$ and $G(\mathbf{z}^*_c + \epsilon_i)$ as σ_ϵ grows for 20 test and 20 train images, where $\epsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma^2_\epsilon \mathbf{I})$

.

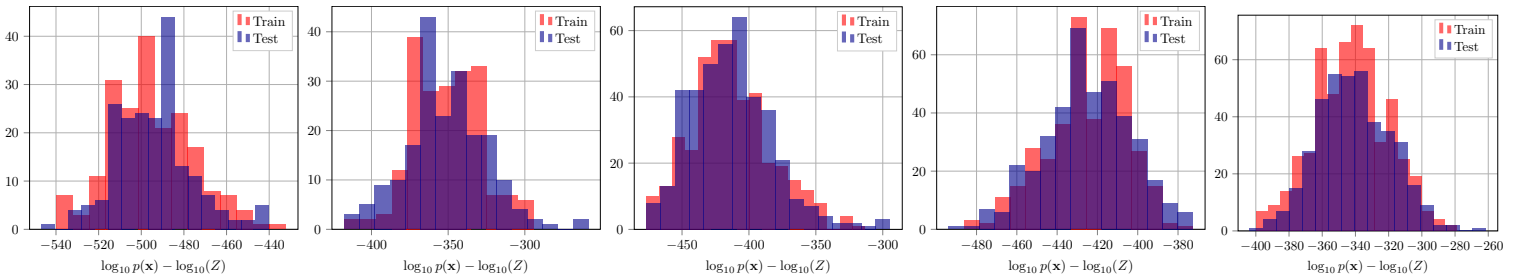


(a) WGANGP CIFAR10 (b) WGAN CIFAR10 (c) SNDCGAN CIFAR10 (d) WGAN-GP celebA (e) SNDCGAN celebA



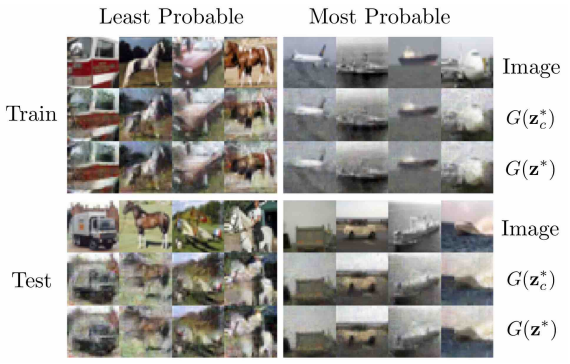
(f) WGANGP CIFAR10 (g) WGAN CIFAR10 (h) SNDCGAN CIFAR10 (i) WGAN-GP celebA (j) SNDCGAN celebA

Figure 16: In the top row we show the histogram of the maximum value of σ_ϵ for which the average PSNR w.r.t. to $G(\mathbf{z}_c^*)$ is less than 40 dB. In the bottom row, we reproduce this experiment for a maximum PSNR value of 30 dB.

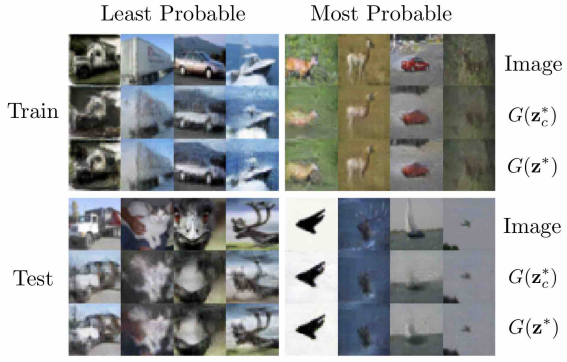


(a) WGANGP CIFAR10 (b) WGAN CIFAR10 (c) SNDCGAN CIFAR10 (d) WGAN-GP celebA (e) SNDCGAN celebA

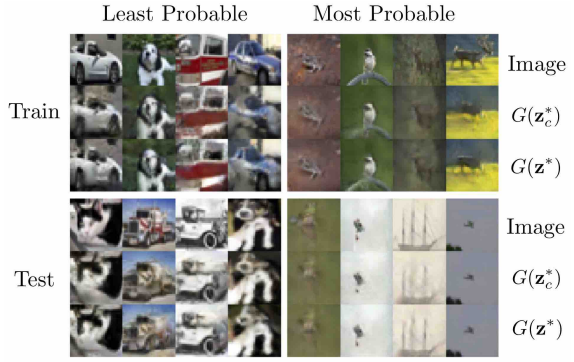
Figure 17: Histogram of the EvalGAN estimated log-probability (unnormalized) using 400 test˙train images.



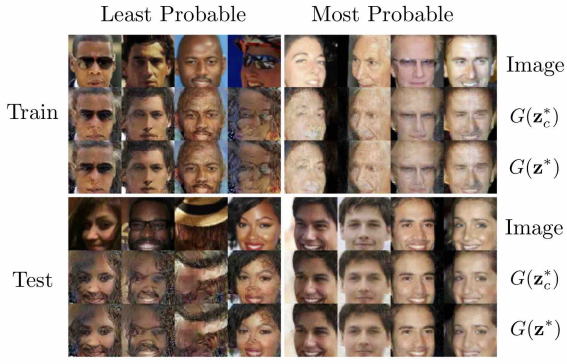
(a) WGANGP CIFAR10



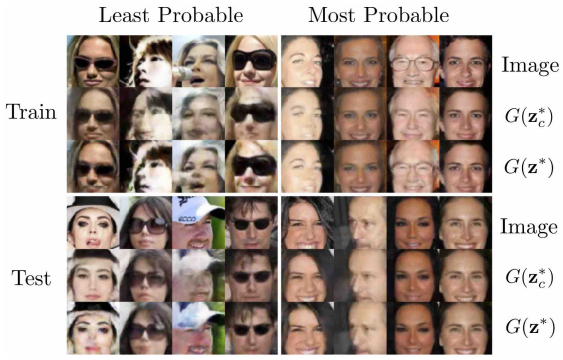
(b) WGAN CIFAR10



(c) SNDCGAN CIFAR10

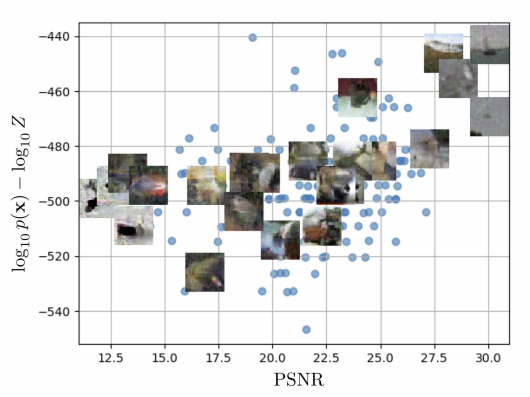


(d) WGAN-GP celebA

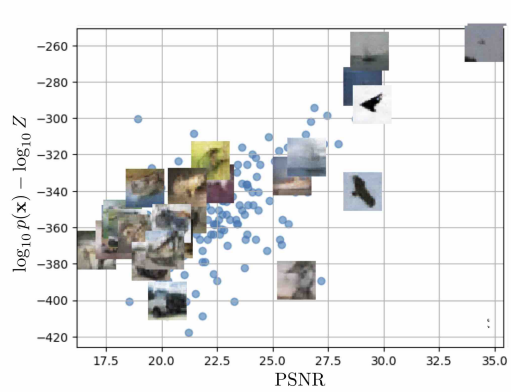


(e) SNDCGAN celebA

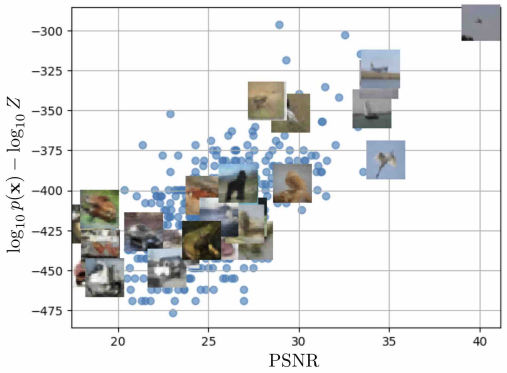
Figure 18: We plot the most and least probable images for each case according to the EvalGAN probability measure.



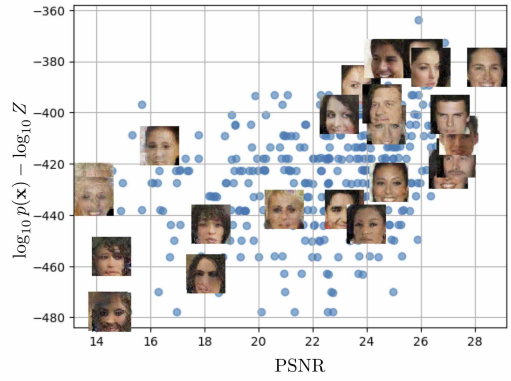
(a) WGANGP CIFAR10



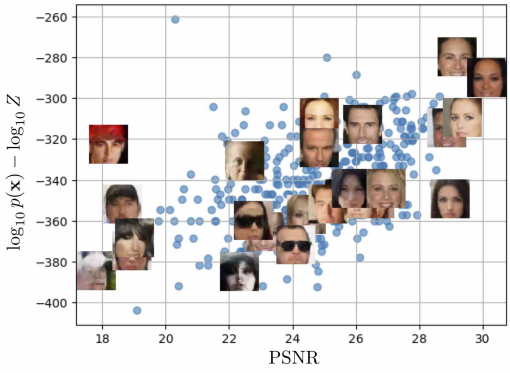
(b) WGAN CIFAR10



(c) SNDCGAN CIFAR10

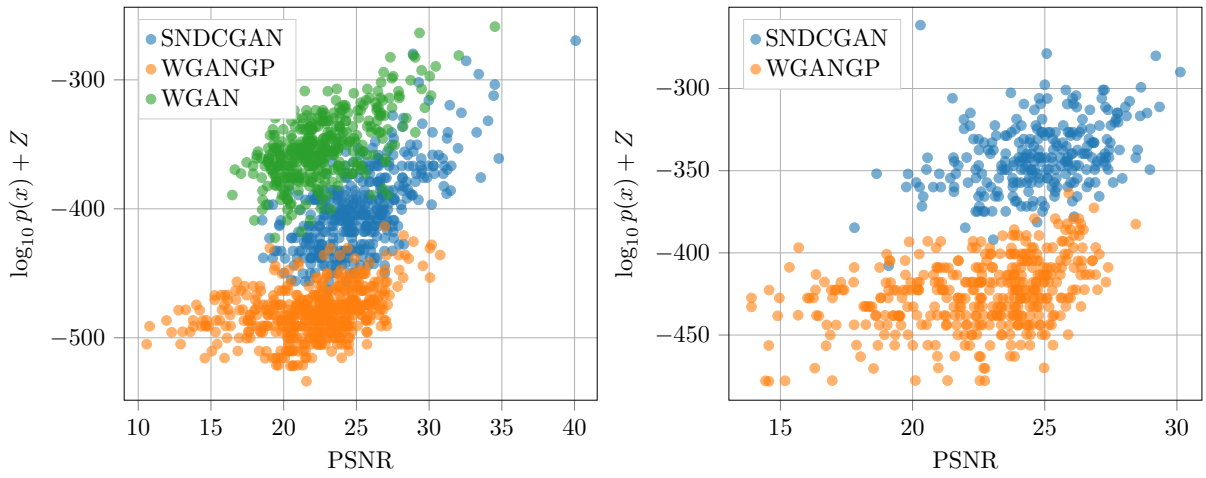


(d) WGAN-GP celebA



(e) SNDCGAN celebA

Figure 19: Scatter plot of PSNR versus estimated loglikelihood obtained with EvalGAN. We also show some reconstructed images $G(\mathbf{z}_c^*)$ overlaying their corresponding location in the plot.



(a) CIFAR10

(b) CelebA

Figure 20: Comparison of different models using EvalGAN for CIFAR10 in (a) and CelebA in (b)