Prescribed Generative Adversarial Networks

Adji B. Dieng[1], Francisco J. R. Ruiz[2, 3],
David M. Blei[1, 2], and Michalis K. Titsias[4]

[1]Department of Statistics, Columbia University
[2]Department of Computer Science, Columbia University
[3]Department of Engineering, University of Cambridge
[4]DeepMind

October 11, 2019

Abstract

Generative adversarial networks (GANs) are a powerful approach to unsupervised learning. They have achieved state-of-the-art performance in the image domain. However, GANs are limited in two ways. They often learn distributions with low support—a phenomenon known as mode collapse—and they do not guarantee the existence of a probability density, which makes evaluating generalization using predictive log-likelihood impossible. In this paper, we develop the prescribed GAN (PresGAN) to address these shortcomings. PresGANs add noise to the output of a density network and optimize an entropy-regularized adversarial loss. The added noise renders tractable approximations of the predictive log-likelihood and stabilizes the training procedure. The entropy regularizer encourages PresGANs to capture all the modes of the data distribution. Fitting PresGANs involves computing the intractable gradients of the entropy regularization term; PresGANs sidestep this intractability using unbiased stochastic estimates. We evaluate PresGANs on several datasets and found they mitigate mode collapse and generate samples with high perceptual quality. We further found that PresGANs reduce the gap in performance in terms of predictive log-likelihood between traditional GANs and variational auto-encoders (VAEs).[1]

Keywords: generative adversarial networks, entropy regularization, log-likelihood evaluation, mode collapse, diverse image generation, deep generative models

1 Introduction

Generative adversarial networks (GANs) (Goodfellow et al., 2014) are a family of generative models that have shown great promise. They achieve state-of-the-art performance in the image domain; for example image generation (Karras et al., 2019; Brock et al., 2018), image super-resolution (Ledig et al., 2017), and image translation (Isola et al., 2017).

[1]**Code:** The code for this paper can be found at https://github.com/adjidieng/PresGANs.

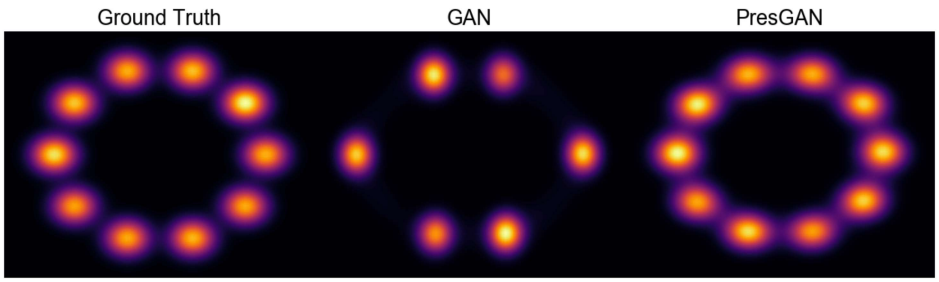

Figure 1: Density estimation with GAN and PresGAN on a toy two-dimensional experiment. The ground truth is a uniform mixture of 10 Gaussians organized on a ring. Given the right set of hyperparameters, a GAN could perfectly fit this target distribution. In this example we chose the GAN hyperparameters such that it collapses—here 4 out of 10 modes are missing. We then fit the PresGAN using the same hyperparameters as the collapsing GAN. The PresGAN is able to correct the collapsing behavior of the GAN and learns a good fit for the target distribution.

GANs learn densities by defining a sampling procedure. A latent variable $\mathbf{z}$ is sampled from a prior $p(\mathbf{z})$ and a sample $\tilde{\mathbf{x}}(\mathbf{z};\theta)$ is generated by taking the output of a neural network with parameters θ, called a generator, that takes $\mathbf{z}$ as input. The density $p_\theta(\mathbf{x})$ implied by this sampling procedure is implicit and undefined (Mohamed and Lakshminarayanan, 2016). However, GANs effectively learn the parameters θ by introducing a classifier D_ϕ—a deep neural network with parameters ϕ, called discriminator—that distinguishes between generated samples $\tilde{\mathbf{x}}(\mathbf{z};\theta)$ and real data $\mathbf{x}$, with distribution $p_d(\mathbf{x})$. The parameters θ and ϕ are learned jointly by optimizing the GAN objective,

$$\mathcal{L}_{\text{GAN}}(\theta,\phi) = \mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}\big[\log D_\phi(\mathbf{x})\big] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\big[\log\big(1 - D_\phi(\tilde{\mathbf{x}}(\mathbf{z};\theta))\big)\big]. \quad (1)$$

GANs iteratively maximize the loss in Eq. 1 with respect to ϕ and minimize it with respect to θ.

In practice, the minimax procedure described above is stopped when the generator produces realistic images. This is problematic because high perceptual quality does not necessarily correlate with goodness of fit to the target density. For example, memorizing the training data is a trivial solution to achieving high perceptual quality. Fortunately, GANs do not merely memorize the training data (Zhang et al., 2017; Arora et al., 2017).

However GANs are able to produce images indistinguishable from real images while still failing to fully capture the target distribution (Brock et al., 2018; Karras et al., 2019). Indeed GANs suffer from an issue known as *mode collapse*. When mode collapse happens, the generative distribution $p_\theta(\mathbf{x})$ is degenerate and of low support (Arora et al., 2017, 2018). Mode collapse causes GANs, as density estimators, to fail both qualitatively and quantitatively. Qualitatively, mode collapse causes lack of diversity in the generated samples. This is problematic for certain applications of GANs, e.g. data augmentation. Quantitatively, mode collapse causes poor generalization to new data. This is because when mode collapse happens, there is a

(support) mismatch between the learned distribution $p_\theta(\mathbf{x})$ and the data distribution. Using annealed importance sampling with a kernel density estimate of the likelihood, Wu et al. (2016) report significantly worse log-likelihood scores for GANs when compared to variational auto-encoders (VAEs). Similarly poor generalization performance was reported by Grover et al. (2018).

A natural way to prevent mode collapse in GANs is to maximize the entropy of the generator (Belghazi et al., 2018). Unfortunately the entropy of GANs is unavailable. This is because the existence of the generative density $p_\theta(\mathbf{x})$ is not guaranteed (Mohamed and Lakshminarayanan, 2016; Arjovsky et al., 2017).

In this paper, we propose a method to alleviate mode collapse in GANs resulting in a new family of GANs called prescribed GANs (PresGANs). PresGANs prevent mode collapse by explicitly maximizing the entropy of the generator. This is done by augmenting the loss in Eq. 1 with the negative entropy of the generator, such that minimizing Eq. 1 with respect to θ corresponds to fitting the data while also maximizing the entropy of the generative distribution. The existence of the generative density is guaranteed by adding noise to the output of a density network (MacKay, 1995; Diggle and Gratton, 1984). This process defines the generative distribution $p_\theta(\mathbf{x})$, not as an implicit distribution as in standard GANs, but as an infinite mixture of well-defined densities as in continuous VAEs (Kingma and Welling, 2013; Rezende et al., 2014). The generative distribution of PresGANs is therefore very flexible.

Although the entropy of the generative distribution of PresGANs is well-defined, it is intractable. However, fitting a PresGAN to data only involves computing the gradients of the entropy and not the entropy itself. PresGANs use unbiased Monte Carlo estimates of these gradients.

An illustrative example. To demonstrate how PresGANs alleviate mode collapse, we form a target distribution by organizing a uniform mixture of $K = 10$ two-dimensional Gaussians on a ring. We draw 5,000 samples from this target distribution. We first fit a GAN, setting the hyperparameters so that the GAN suffers from mode collapse[2]. We then use the same settings for PresGAN to assess whether it can correct the collapsing behavior of the GAN. Figure 1 shows the collapsing behavior of the GAN, which misses 4 modes of the target distribution. The PresGAN, on the other hand, recovers all the modes. Section 5 provides details about the settings of this synthetic experiment.

Contributions. This paper contributes to the literature on the two main open problems in the study of GANs: preventing mode collapse and evaluating log-likelihood.

- How can we perform entropy regularization of the generator of a GAN so as to effectively prevent mode collapse? We achieve this by adding noise to the output of the generator; this ensures the existence of a density $p_\theta(\mathbf{x})$ and makes its entropy well-defined. We then regularize the GAN loss to encourage densities $p_\theta(\mathbf{x})$ with high entropy. During training, we form unbiased estimators of the (intractable) gradients of the entropy regularizer. We show how this prevents

[2] A GAN can perfectly fit this distribution when choosing the right hyperparameters.

mode collapse, as expected, in two sets of experiments (see Section 5). The first experiment follows the current standard for measuring mode collapse in the GAN literature, which is to report the number of modes recovered by the GAN on MNIST (10 modes) and STACKEDMNIST (1,000 modes) and the Kullback-Leibler (KL) divergence between the true label distribution and the one induced by the GAN. We conducted a second experiment which sheds light on another way mode collapse can occur in GANs, which is when the data is imbalanced.

- How can we measure log-likelihood in GANs? Evaluating log-likelihood for GANs allows assessing how they generalize to new data. Existing measures focus on sample quality, which is not a measure of generalization. This inability to measure predictive log-likelihood for GANs has restricted their use to domains where one can use perceptual quality measures (e.g., the image domain). Existing methods for evaluating log-likelihood for GANs either use a proxy to log-likelihood (Sánchez-Martín et al., 2019) or define the likelihood of the generator only at test time, which creates a mismatch between training and testing (Wu et al., 2016), or assume invertibility of the generator of the GAN (Grover et al., 2018). Adding noise to the output of the generator immediately makes tractable predictive log-likelihood evaluation via importance sampling.

Outline. The rest of the paper is organized as follows. In Section 2 we set the notation and provide desiderata for deep generative modeling. In Section 3 we describe PresGANs and how we compute their predictive log-likelihood to assess generalization. In Section 4 we discuss related work. We then assess the performance of PresGANs in terms of mode collapse, sample quality, and log-likelihood in Section 5. Finally, we conclude and discuss key findings in Section 6.

2 Prologue

In this paper, we characterize a deep generative model (DGM) by its generative process and by the loss used to fit its parameters. We denote by $p_\theta(\mathbf{x})$ the generative distribution induced by the generative process—it is parameterized by a deep neural network with parameters θ. The loss, that we denote by $\mathscr{L}(\theta, \phi)$, often requires an additional set of parameters ϕ that help learn the model parameters θ. We next describe choices for $p_\theta(\mathbf{x})$ and $\mathscr{L}(\theta, \phi)$ and then specify desiderata for deep generative modeling.

The generative distribution. Recent DGMs define the generative distribution either as an implicit distribution or as an infinite mixture (Goodfellow et al., 2014; Kingma and Welling, 2013; Rezende et al., 2014).

Implicit generative models define a density using a sampling procedure. This is the approach of GANs (Goodfellow et al., 2014). A latent variable $\mathbf{z}$ is sampled from a prior $p(\mathbf{z})$, usually a standard Gaussian or a uniform distributon, and a sample is generated by taking the output of a neural network that takes $\mathbf{z}$ as input. The density $p_\theta(\mathbf{x})$ implied by this sampling procedure is undefined. Any measure that relies on an analytic form of the density $p_\theta(\mathbf{x})$ is therefore unavailable; e.g., the log-likelihood or the entropy.

An alternative way to define the generative distribution is by using the approach of VAEs (Kingma and Welling, 2013; Rezende et al., 2014). They define $p_\theta(\mathbf{x})$ as an infinite mixture,

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}\,|\,\mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}. \tag{2}$$

Here the mixing distribution is the prior $p(\mathbf{z})$. The conditional distribution $p_\theta(\mathbf{x}\,|\,\mathbf{z})$ is an exponential family distribution, such as a Gaussian or a Bernoulli, parameterized by a neural network with parameters θ. Although both the prior $p(\mathbf{z})$ and $p_\theta(\mathbf{x}\,|\,\mathbf{z})$ are simple tractable distributions, the generative distribution $p_\theta(\mathbf{x})$ is highly flexible albeit intractable. Because $p_\theta(\mathbf{x})$ in Eq. 2 is well-defined, the log-likelihood and the entropy are also well-defined (although they may be analytically intractable).

The loss function. Fitting the models defined above requires defining a learning procedure by specifying a loss function. GANs introduce a classifier D_ϕ, a deep neural network parameterized by ϕ, to discriminate between samples from the data distribution $p_d(\mathbf{x})$ and the generative distribution $p_\theta(\mathbf{x})$. The auxiliary parameters ϕ are learned jointly with the model parameters θ by optimizing the loss in Eq. 1. This training procedure leads to high sample quality but often suffers from *mode collapse* (Arora et al., 2017, 2018).

An alternative approach to learning θ is via maximum likelihood. This requires a well-defined density $p_\theta(\mathbf{x})$ such as the one in Eq. 2. Although well-defined, $p_\theta(\mathbf{x})$ is intractable, making it difficult to learn the parameters θ by maximum likelihood. VAEs instead introduce a recognition network—a neural network with parameters ϕ that takes data $\mathbf{x}$ as input and outputs a distribution over the latent variables $\mathbf{z}$—and maximize a lower bound on $\log p_\theta(\mathbf{x})$ with respect to both θ and ϕ,

$$\mathcal{L}_{\text{VAE}}(\theta, \phi) = E_{p_d(\mathbf{x})} E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x})}\right] = -\text{KL}(q_\phi(\mathbf{z}\,|\,\mathbf{x}) p_d(\mathbf{x}) || p_\theta(\mathbf{x}, \mathbf{z})). \tag{3}$$

Here $\text{KL}(\cdot||\cdot)$ denotes the KL divergence. Maximizing $\mathcal{L}_{\text{VAE}}(\theta, \phi)$ is equivalent to minimizing this KL which leads to issues such as latent variable collapse (Bowman et al., 2015; Dieng et al., 2018b). Furthermore, optimizing Eq. 3 may lead to blurriness in the generated samples because of a property of the reverse KL known as *zero-forcing* (Minka et al., 2005).

Desiderata. We now outline three desiderata for DGMs.

High sample quality. A DGM whose parameters θ have been fitted using real data should generate new data with the same qualitative precision as the data it was trained with. For example, if a DGM is trained on a dataset composed of human faces, it should generate data with all features that make up a face at the same resolution as the training data.

High sample diversity. High sample quality alone is not enough. For example, a degenerate DGM that is only able to produce one single sample is not desirable, even if the sample quality is perfect. Therefore we require sample diversity; a DGM should ideally capture all modes of the data distribution.

Tractable predictive log-likelihood. DGMs are density estimators and as such we should evaluate how they generalize to new data. High sample quality and diversity are not

measures of generalization. We therefore require tractable predictive log-likelihood as a desideratum for deep generative modeling.

We next introduce a new family of GANs that fulfills all the desiderata.

3 Prescribed Generative Adversarial Networks

PresGANs generate data following the generative distribution in Eq. 2. Note that this generative process is the same as for standard VAEs (Kingma and Welling, 2013; Rezende et al., 2014). In particular, PresGANs set the prior $p(\mathbf{z})$ and the likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ to be Gaussians,

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0},\mathbf{I}) \quad \text{and} \quad p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mu_\theta(\mathbf{z}),\mathbf{\Sigma}_\theta(\mathbf{z})). \tag{4}$$

The mean $\mu_\theta(\mathbf{z})$ and covariance $\mathbf{\Sigma}_\theta(\mathbf{z})$ of the conditional $p_\theta(\mathbf{x}|\mathbf{z})$ are given by a neural network that takes $\mathbf{z}$ as input.

In general, both the mean $\mu_\theta(\mathbf{z})$ and the covariance $\mathbf{\Sigma}_\theta(\mathbf{z})$ can be functions of $\mathbf{z}$. For simplicity, in order to speed up the learning procedure, we set the covariance matrix to be diagonal with elements independent from $\mathbf{z}$, i.e., $\mathbf{\Sigma}_\theta(\mathbf{z}) = \text{diag}\left(\boldsymbol{\sigma}^2\right)$, and we learn the vector $\boldsymbol{\sigma}$ together with θ. From now on, we parameterize the mean with η, write $\mu_\eta(\mathbf{z})$, and define $\theta = (\eta, \boldsymbol{\sigma})$ as the parameters of the generative distribution.

To fit the model parameters θ, PresGANs optimize an adversarial loss similarly to GANs. In doing so, they keep GANs' ability to generate samples with high perceptual quality. Unlike GANs, the entropy of the generative distribution of PresGANs is well-defined, and therefore PresGANs can prevent mode collapse by adding an entropy regularizer to Eq. 1. Furthermore, because PresGANs define a density over their generated samples, we can measure how they generalize to new data using predictive log-likelihood. We describe the entropy regularization in Section 3.1 and how to approximate the predictive log-likelihood in Section 3.3.

3.1 Avoiding mode collapse via entropy regularization

One of the major issues that GANs face is mode collapse, where the generator tends to model only some parts or modes of the data distribution (Arora et al., 2017, 2018). PresGANs mitigate this problem by explicitly maximizing the entropy of the generative distribution,

$$\mathcal{L}_{\text{PresGAN}}(\theta,\phi) = \mathcal{L}_{\text{GAN}}(\theta,\phi) - \lambda \mathcal{H}\left(p_\theta(\mathbf{x})\right). \tag{5}$$

Here $\mathcal{H}\left(p_\theta(\mathbf{x})\right)$ denotes the entropy of the generative distribution. It is defined as

$$\mathcal{H}\left(p_\theta(\mathbf{x})\right) = -\mathbb{E}_{p_\theta(\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right]. \tag{6}$$

The loss $\mathcal{L}_{\text{GAN}}(\theta,\phi)$ in Eq. 5 can be that of any of the existing GAN variants. In Section 5 we explore the standard deep convolutional generative adversarial network

Algorithm 1: Learning with Prescribed Generative Adversarial Networks (PresGANs)

input : Data $\mathbf{x}$, entropy regularization level λ
Initialize parameters $\eta, \boldsymbol{\sigma}, \phi$
for iteration $t = 1, 2, \ldots$ **do**
  Draw minibatch of observations $\mathbf{x}_1, \ldots, \mathbf{x}_b, \ldots, \mathbf{x}_B$
  for $b = 1, 2, \ldots, B$ **do**
    Get noised data: $\epsilon_b \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\widehat{\mathbf{x}}_b = \mathbf{x}_b + \boldsymbol{\sigma} \odot \epsilon_b$
    Draw latent variable $\mathbf{z}_b \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Generate data: $\mathbf{s}_b \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\tilde{\mathbf{x}}_b = \tilde{\mathbf{x}}_b(\mathbf{z}_b, \mathbf{s}_b; \theta) = \mu_\eta(\mathbf{z}_b) + \boldsymbol{\sigma} \odot \mathbf{s}_b$
  end
  Compute $\nabla_\phi \mathcal{L}_{\text{PresGAN}}(\theta, \phi)$ (Eq. 16) and take a gradient step for ϕ
  Initialize an HMC sampler using $\mathbf{z}_b$
  Draw $\tilde{\mathbf{z}}_b^{(m)} \sim p_\theta(\mathbf{z} \mid \tilde{\mathbf{x}}_b)$ for $m = 1, \ldots, M$ and $b = 1, \ldots, B$ using that sampler
  Compute $\widehat{\nabla}_\eta \mathcal{L}_{\text{PresGAN}}((\eta, \boldsymbol{\sigma}), \phi)$ (Eq. 14) and take a gradient step for η
  Compute $\widehat{\nabla}_{\boldsymbol{\sigma}} \mathcal{L}_{\text{PresGAN}}((\eta, \boldsymbol{\sigma}), \phi)$ (Eq. 15) and take a gradient step for $\boldsymbol{\sigma}$
  Truncate $\boldsymbol{\sigma}$ in the range $[\boldsymbol{\sigma}_{\text{low}}, \boldsymbol{\sigma}_{\text{high}}]$
end

(DCGAN) (Radford et al., 2015) and the more recent StyleGAN (Karras et al., 2019) architectures.

The constant λ in Eq. 5 is a hyperparameter that controls the strength of the entropy regularization. In the extreme case when $\lambda = 0$, the loss function of PresGAN coincides with the loss of a GAN, where we replaced its implicit generative distribution with the infinite mixture in Eq. 2. In the other extreme when $\lambda = \infty$, optimizing $\mathcal{L}_{\text{PresGAN}}(\theta, \phi)$ corresponds to fitting a maximum entropy generator that ignores the data. For any intermediate values of λ, the first term of $\mathcal{L}_{\text{PresGAN}}(\theta, \phi)$ encourages the generator to fit the data distribution, whereas the second term encourages to cover all of the modes of the data distribution.

The entropy $\mathcal{H}(p_\theta(\mathbf{x}))$ is intractable because the integral in Eq. 6 cannot be computed. However, fitting the parameters θ of PresGANs only requires the gradients of the entropy. In Section 3.2 we describe how to form unbiased Monte Carlo estimates of these gradients.

3.2 Fitting Prescribed Generative Adversarial Networks

We fit PresGANs following the same adversarial procedure used in GANs. That is, we alternate between updating the parameters of the generative distribution θ and the parameters of the discriminator ϕ. The full procedure is given in Algorithm 1. We now describe each part in detail.

Fitting the generator. We fit the generator using stochastic gradient descent. This requires computing the gradients of the PresGAN loss with respect to θ,

$$\nabla_\theta \mathcal{L}_{\text{PresGAN}}(\theta, \phi) = \nabla_\theta \mathcal{L}_{\text{GAN}}(\theta, \phi) - \lambda \nabla_\theta \mathcal{H}(p_\theta(\mathbf{x})). \tag{7}$$

We form stochastic estimates of $\nabla_\theta \mathscr{L}_{\text{GAN}}(\theta, \phi)$ based on reparameterization (Kingma and Welling, 2013; Rezende et al., 2014; Titsias and Lázaro-Gredilla, 2014); this requires differentiating Eq. 1. Specifically, we introduce a noise variable ϵ to reparameterize the conditional from Eq. 4,[3]

$$\mathbf{x}(\mathbf{z}, \epsilon; \theta) = \mu_\eta(\mathbf{z}) + \boldsymbol{\sigma} \odot \epsilon, \tag{8}$$

where $\theta = (\eta, \boldsymbol{\sigma})$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Here $\mu_\eta(\mathbf{z})$ and $\boldsymbol{\sigma}$ denote the mean and standard deviation of the conditional $p_\theta(\mathbf{x}|\mathbf{z})$, respectively. We now write the first term of Eq. 7 as an expectation with respect to the latent variable $\mathbf{z}$ and the noise variable ϵ and push the gradient into the expectation,

$$\nabla_\theta \mathscr{L}_{\text{GAN}}(\theta, \phi) = \mathbb{E}_{p(\mathbf{z})p(\epsilon)}\big[\nabla_\theta \log\big(1 - D_\phi(\mathbf{x}(\mathbf{z}, \epsilon; \theta))\big)\big]. \tag{9}$$

In practice we use an estimate of Eq. 9 using one sample from $p(\mathbf{z})$ and one sample from $p(\epsilon)$,

$$\widehat{\nabla}_\theta \mathscr{L}_{\text{GAN}}(\theta, \phi) = \nabla_\theta \log\big(1 - D_\phi(\mathbf{x}(\mathbf{z}, \epsilon; \theta))\big). \tag{10}$$

The second term in Eq. 7, corresponding to the gradient of the entropy, is intractable. We estimate it using the same approach as Titsias and Ruiz (2018). We first use the reparameterization in Eq. 8 to express the gradient of the entropy as an expectation,

$$\begin{aligned}
\nabla_\theta \mathscr{H}(p_\theta(\mathbf{x})) &= -\nabla_\theta \mathbb{E}_{p_\theta(\mathbf{x})}[\log p_\theta(\mathbf{x})] = -\nabla_\theta \mathbb{E}_{p(\epsilon)p(\mathbf{z})}\Big[\log p_\theta(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}(\mathbf{z},\epsilon;\theta)}\Big] \\
&= -\mathbb{E}_{p(\epsilon)p(\mathbf{z})}\Big[\nabla_\theta \log p_\theta(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}(\mathbf{z},\epsilon;\theta)}\Big] \\
&= -\mathbb{E}_{p(\epsilon)p(\mathbf{z})}\Big[\nabla_\mathbf{x} \log p_\theta(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}(\mathbf{z},\epsilon;\theta)} \nabla_\theta \mathbf{x}(\mathbf{z}, \epsilon; \theta)\Big],
\end{aligned}$$

where we have used the score function identity $\mathbb{E}_{p_\theta(\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{x})] = 0$ on the second line. We form a one-sample estimator of the gradient of the entropy as

$$\widehat{\nabla}_\theta \mathscr{H}(p_\theta(\mathbf{x})) = -\nabla_\mathbf{x} \log p_\theta(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}(\mathbf{z},\epsilon;\theta)} \times \nabla_\theta \mathbf{x}(\mathbf{z}, \epsilon; \theta). \tag{11}$$

In Eq. 11, the gradient with respect to the reparameterization transformation $\nabla_\theta \mathbf{x}(\mathbf{z}, \epsilon; \theta)$ is tractable and can be obtained via back-propagation. We now derive $\nabla_\mathbf{x} \log p_\theta(\mathbf{x})$,

$$\begin{aligned}
\nabla_\mathbf{x} \log p_\theta(\mathbf{x}) = \frac{\nabla_\mathbf{x} p_\theta(\mathbf{x})}{p_\theta(\mathbf{x})} &= \frac{\int \nabla_\mathbf{x} p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}}{p_\theta(\mathbf{x})} = \int \frac{\frac{\nabla_\mathbf{x} p_\theta(\mathbf{x}|\mathbf{z})}{p_\theta(\mathbf{x}|\mathbf{z})} p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x})} d\mathbf{z} \\
&= \int \nabla_\mathbf{x} \log p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}|\mathbf{x}) d\mathbf{z} = \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\nabla_\mathbf{x} \log p_\theta(\mathbf{x}|\mathbf{z})].
\end{aligned}$$

While this expression is still intractable, we can estimate it. One way is to use self-normalized importance sampling with a proposal learned using moment matching

[3] With this reparameterization we use the notation $\mathbf{x}(\mathbf{z}, \epsilon; \theta)$ instead of $\tilde{\mathbf{x}}(\mathbf{z}; \theta)$ to denote a sample from the generative distribution.

with an encoder (Dieng and Paisley, 2019). However, this would lead to a biased (albeit asymptotically unbiased) estimate of the entropy. In this paper, we form an unbiased estimate of $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$ using samples $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(M)}$ from the posterior,

$$\widehat{\nabla}_{\mathbf{x}} \log p_\theta(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x} \mid \mathbf{z}^{(m)}), \qquad \mathbf{z}^{(m)} \sim p_\theta(\mathbf{z} \mid \mathbf{x}). \tag{12}$$

We obtain these samples using Hamiltonian Monte Carlo (HMC) (Neal et al., 2011). Crucially, in order to speed up the algorithm, we initialize the HMC sampler at stationarity. That is, we initialize the HMC sampler with the sample $\mathbf{z}$ that was used to produce the generated sample $\mathbf{x}(\mathbf{z}, \epsilon; \theta)$ in Eq. 8, which by construction is an exact sample from $p_\theta(\mathbf{z} \mid \mathbf{x})$. This implies that only a few HMC iterations suffice to get good estimates of the gradient (Titsias and Ruiz, 2018). We also found this holds empirically; for example in Section 5 we use 2 burn-in iterations and $M = 2$ HMC samples to form the Monte Carlo estimate in Eq. 12.

Finally, using Eqs. 7 and 10 to 12 we can approximate the gradient of the entropy-regularized adversarial loss with respect to the model parameters θ,

$$\begin{aligned}\widehat{\nabla}_\theta \mathcal{L}_{\text{PresGAN}}(\theta, \phi) = \nabla_\theta \log\big(1 - D_\phi(\mathbf{x}(\mathbf{z}, \epsilon; \theta))\big) \\ + \frac{\lambda}{M} \sum_{m=1}^{M} \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x} \mid \mathbf{z}^{(m)})\Big|_{\mathbf{x}=\mathbf{x}(\mathbf{z}^{(m)}, \epsilon; \theta)} \times \nabla_\theta \mathbf{x}\big(\mathbf{z}^{(m)}, \epsilon; \theta\big).\end{aligned} \tag{13}$$

In particular, the gradient with respect to the generator's parameters η is unbiasedly approximated by

$$\begin{aligned}\widehat{\nabla}_\eta \mathcal{L}_{\text{PresGAN}}(\theta, \phi) = \nabla_\eta \log\big(1 - D_\phi(\mathbf{x}(\mathbf{z}, \epsilon; \theta))\big) \\ - \frac{\lambda}{M} \sum_{m=1}^{M} \frac{\mathbf{x}(\mathbf{z}^{(m)}, \epsilon; \theta) - \mu_\eta\big(\mathbf{z}^{(m)}\big)}{\boldsymbol{\sigma}^2} \nabla_\eta \mu_\eta(\mathbf{z}^{(m)}),\end{aligned} \tag{14}$$

and the gradient estimator with respect to the standard deviation $\boldsymbol{\sigma}$ is

$$\begin{aligned}\widehat{\nabla}_{\boldsymbol{\sigma}} \mathcal{L}_{\text{PresGAN}}(\theta, \phi) = \nabla_{\boldsymbol{\sigma}} \log\big(1 - D_\phi(\mathbf{x}(\mathbf{z}, \epsilon; \theta))\big) \\ - \frac{\lambda}{M} \sum_{m=1}^{M} \frac{\mathbf{x}(\mathbf{z}^{(m)}, \epsilon; \theta) - \mu_\eta\big(\mathbf{z}^{(m)}\big)}{\boldsymbol{\sigma}^2} \cdot \epsilon.\end{aligned} \tag{15}$$

These gradients are used in a stochastic optimization algorithm to fit the generative distribution of PresGAN.

Fitting the discriminator. Since the entropy term in Eq. 5 does not depend on ϕ, optimizing the discriminator of a PresGAN is analogous to optimizing the discriminator of a GAN,

$$\nabla_\phi \mathcal{L}_{\text{PresGAN}}(\theta, \phi) = \nabla_\phi \mathcal{L}_{\text{GAN}}(\theta, \phi). \tag{16}$$

To prevent the discriminator from getting stuck in a bad local optimum where it can perfectly distinguish between real and generated data by relying on the added noise, we apply the same amount of noise to the real data $\mathbf{x}$ as the noise added to

the generated data. That is, when we train the discriminator we corrupt the real data according to

$$\widehat{\mathbf{x}} = \mathbf{x} + \boldsymbol{\sigma} \odot \epsilon, \tag{17}$$

where $\boldsymbol{\sigma}$ is the standard deviation of the generative distribution and $\mathbf{x}$ denotes the real data. We then let the discriminator distinguish between $\widehat{\mathbf{x}}$ and $\mathbf{x}(\mathbf{z}, \epsilon; \theta)$ from Eq. 8.

This data noising procedure is a form of *instance noise* (Sønderby et al., 2016). However, instead of using a fixed annealing schedule for the noise variance as Sønderby et al. (2016), we let $\boldsymbol{\sigma}$ be part of the parameters of the generative distribution and fit it using gradient descent according to Eq. 15.

Stability. Data noising stabilizes the training procedure and prevents the discriminator from perfectly being able to distinguish between real and generated samples using the background noise. We refer the reader to Huszár (2016) for a detailed exposition.

When fitting PresGANs, data noising is not enough to stabilize training. This is because there are two failure cases brought in by learning the variance $\boldsymbol{\sigma}^2$ using gradient descent. The first failure mode is when the variance gets very large, leading to a generator completely able to fool the discriminator. Because of data noising, the discriminator cannot distinguish between real and generated samples when the variance of the noise is large.

The second failure mode is when $\boldsymbol{\sigma}^2$ gets very small, which makes the gradient of the entropy in Eq. 14 dominate the overall gradient of the generator. This is problematic because the learning signal from the discriminator is lost.

To stabilize training and avoid the two failure cases discussed above we truncate the variance of the generative distribution, $\boldsymbol{\sigma}_{\text{low}} \leq \boldsymbol{\sigma} \leq \boldsymbol{\sigma}_{\text{high}}$ (we apply this truncation element-wise). The limits $\boldsymbol{\sigma}_{\text{low}}$ and $\boldsymbol{\sigma}_{\text{high}}$ are hyperparameters.

3.3 Enabling tractable predictive log-likelihood approximation

Replacing the implicit generative distribution of GANs with the infinite mixture distribution defined in Eq. 2 has the advantage that the predictive log-likelihood can be tractably approximated. Consider an unseen datapoint $\mathbf{x}^*$. We estimate its log marginal likelihood $\log p_\theta(\mathbf{x}^*)$ using importance sampling,

$$\log p_\theta(\mathbf{x}^*) \approx \log \left(\frac{1}{S} \sum_{s=1}^{S} \frac{p_\theta\left(\mathbf{x}^* \,|\, \mathbf{z}^{(s)}\right) \cdot p\left(\mathbf{z}^{(s)}\right)}{r\left(\mathbf{z}^{(s)} \,|\, \mathbf{x}^*\right)} \right), \tag{18}$$

where we draw S samples $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(S)}$ from a proposal distribution $r(\mathbf{z} \,|\, \mathbf{x}^*)$.

There are different ways to form a good proposal $r(\mathbf{z} \,|\, \mathbf{x}^*)$, and we discuss several alternatives in Section 7.1 of the appendix. In this paper, we take the following approach. We define the proposal as a Gaussian distribution,

$$r(\mathbf{z} \,|\, \mathbf{x}^*) = \mathcal{N}(\mu_r, \boldsymbol{\Sigma}_r). \tag{19}$$

We set the mean parameter μ_r to the *maximum a posteriori* solution, i.e., $\mu_r = \arg\max_{\mathbf{z}} (\log p_\theta(\mathbf{x}^* \mid \mathbf{z}) + \log p(\mathbf{z}))$. We initialize this maximization algorithm using the mean of a pre-fitted encoder, $q_\gamma(\mathbf{z} \mid \mathbf{x}^*)$. The encoder is fitted by minimizing the reverse KL divergence between $q_\gamma(\mathbf{z} \mid \mathbf{x})$ and the true posterior $p_\theta(\mathbf{z} \mid \mathbf{x})$ using the training data. This KL is

$$\mathrm{KL}\left(q_\gamma(\mathbf{z} \mid \mathbf{x}) || p_\theta(\mathbf{z} \mid \mathbf{x})\right) = \log p_\theta(\mathbf{x}) - \mathbb{E}_{q_\gamma(\mathbf{z} \mid \mathbf{x})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) - \log q_\gamma(\mathbf{z} \mid \mathbf{x})\right]. \tag{20}$$

Because the generative distribution is fixed at test time, minimizing the KL here is equivalent to maximizing the second term in Eq. 20, which is the evidence lower bound (ELBO) objective of VAEs.

We set the proposal covariance $\mathbf{\Sigma}_r$ as an overdispersed version[4] of the encoder's covariance matrix, which is diagonal. In particular, to obtain $\mathbf{\Sigma}_r$ we multiply the elements of the encoder's covariance by a factor γ. In Section 5 we set γ to 1.2.

4 Related Work

GANs (Goodfellow et al., 2014) have been extended in multiple ways, using alternative distance metrics and optimization methods (see, e.g., Li et al., 2015; Dziugaite et al., 2015; Nowozin et al., 2016; Arjovsky et al., 2017; Ravuri et al., 2018; Genevay et al., 2017) or using ideas from VAEs (Makhzani et al., 2015; Mescheder et al., 2017; Dumoulin et al., 2016; Donahue et al., 2016; Tolstikhin et al., 2017; Ulyanov et al., 2018; Rosca et al., 2017).

Other extensions aim at improving the sample diversity of GANs. For example, Srivastava et al. (2017) use a reconstructor network that reverses the action of the generator. Lin et al. (2018) use multiple observations (either real or generated) as an input to the discriminator to prevent mode collapse. Azadi et al. (2018) and Turner et al. (2018) use sampling mechanisms to correct errors of the generative distribution. Xiao et al. (2018) relies on identifying the geometric structure of the data embodied under a specific distance metric. Other works have combined adversarial learning with maximum likelihood (Grover et al., 2018; Yin and Zhou, 2019); however, the low sample quality induced by maximum likelihood still occurs. Finally, Cao et al. (2018) introduce a regularizer for the discriminator to encourage diverse activation patterns in the discriminator across different samples. In contrast to these works, PresGANs regularize the entropy of the generator to prevent mode collapse.

The idea of entropy regularization has been widely applied in many problems that involve estimation of unknown probability distributions. Examples include approximate Bayesian inference, where the variational objective contains an entropy penalty (Jordan, 1998; Bishop, 2006; Wainwright et al., 2008; Blei et al., 2017); reinforcement learning, where the entropy regularization allows to estimate more uncertain and explorative policies (Schulman et al., 2015; Mnih et al., 2016); statistical learning, where entropy regularization allows an inferred probability

[4] In general, overdispersed proposals lead to better importance sampling estimates.

distribution to avoid collapsing to a deterministic solution (Freund and Schapire, 1997; Soofi, 2000; Jaynes, 2003); or optimal transport (Rigollet and Weed, 2018). More recently, Kumar et al. (2019) have developed maximum-entropy generators for energy-based models using mutual information as a proxy for entropy.

Another body of related work is about how to quantitatively evaluate GANs. Inception scores measure the sample quality of GANs and are used extensively in the GAN literature (Salimans et al., 2016; Heusel et al., 2017; Bińkowski et al., 2018). However, sample quality measures only assess the quality of GANs as data generators and not as density estimators. Density estimators are evaluated for generalization to new data. Predictive log-likelihood is a measure of goodness of fit that has been used to assess generalization; for example in VAEs. Finding ways to evaluate predictive log-likelihood for GANs has been an open problem, because GANs do not define a density on the generated samples. Wu et al. (2016) use a kernel density estimate (Parzen, 1962) and estimate the log-likelihood with annealed importance sampling (Neal, 2001). Balaji et al. (2018) show that an optimal transport GAN with entropy regularization can be viewed as a generative model that maximizes a variational lower bound on average sample likelihoods, which relates to the approach of VAEs (Kingma and Welling, 2013). Sánchez-Martín et al. (2019) propose EvalGAN, a method to estimate the likelihood. Given an observation $\mathbf{x}^\star$, EvalGAN first finds the closest observation $\widetilde{\mathbf{x}}$ that the GAN is able to generate, and then it estimates the likelihood $p(\mathbf{x}^\star)$ by approximating the proportion of samples $\mathbf{z} \sim p(\mathbf{z})$ that lead to samples $\mathbf{x}$ that are close to $\widetilde{\mathbf{x}}$. EvalGAN requires selecting an appropriate distance metric for each problem and evaluates GANs trained with the usual implicit generative distribution. Finally, Grover et al. (2018) assume invertibility of the generator to make log-likelihood tractable.

5 Empirical Study

Here we demonstrate PresGANs' ability to prevent mode collapse and generate high-quality samples. We also evaluate its predictive performance as measured by log-likelihood.

5.1 An Illustrative Example

In this section, we fit a GAN to a toy synthetic dataset of 10 modes. We choose the hyperparameters such that the GAN collapses. We then apply these same hyperparameters to fit a PresGAN on the same synthetic dataset. This experiment demonstrates the PresGAN's ability to correct the mode collapse problem of a GAN.

We form the target distribution by organizing a uniform mixture of $K = 10$ two-dimensional Gaussians on a ring. The radius of the ring is $r = 3$ and each Gaussian has standard deviation 0.05. We then slice the circle into K parts. The location of the centers of the mixture components are determined as follows. Consider the k^{th}

mixture component. Its coordinates in the 2D space are

$$\text{center}_x = r \cdot \cos\Big(k \cdot \frac{2\pi}{K}\Big) \quad \text{and} \quad \text{center}_y = r \cdot \sin\Big(k \cdot \frac{2\pi}{K}\Big).$$

We draw 5,000 samples from the target distribution and fit a GAN and a PresGAN.

We set the dimension of the latent variables $\mathbf{z}$ used as the input to the generators to 10. We let both the generators and the discriminators have three fully connected layers with tanh activations and 128 hidden units in each layer. We set the minibatch size to 100 and use Adam for optimization (Kingma and Ba, 2014), with a learning rate of 10^{-3} and 10^{-4} for the discriminator and the generator respectively. The Adam hyperparameters are $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We take one step to optimize the generator for each step of the discriminator. We pick a random minibatch at each iteration and run both the GAN and the PresGAN for 500 epochs.

For PresGAN we set the burn-in and the number of HMC samples to 2. We choose a standard number of 5 leapfrog steps and set the HMC learning rate to 0.02. The acceptance rate is fixed at 0.67. The log-variance of the noise of the generative distribution of PresGAN is initialized at 0.0. We put a threshold on the variance to a minimum value of $\boldsymbol{\sigma}_{\text{low}} = 10^{-2}$ and a maximum value of $\boldsymbol{\sigma}_{\text{high}} = 0.3$. The regularization parameter λ is 0.1. We fit the log-variance using Adam with a learning rate of 10^{-4}.

Figure 1 demonstrates how the PresGAN alleviates mode collapse. The distribution learned by the regular GAN misses 4 modes of the target distribution. The PresGAN is able to recover all the modes of the target distribution.

5.2 Assessing mode collapse

In this section we evaluate PresGANs' ability to mitigate mode collapse on real datasets. We run two sets of experiments. In the first set of experiments we adopt the current experimental protocol for assessing mode collapse in the GAN literature. That is, we use the MNIST and STACKEDMNIST datasets, for which we know the true number of modes, and report two metrics: the number of modes recovered by the PresGAN and the KL divergence between the label distribution induced by the PresGAN and the true label distribution. In the second set of experiments we demonstrate that mode collapse can happen in GANs even when the number of modes is as low as 10 but the data is imbalanced.

Increased number of modes. We consider the MNIST and STACKEDMNIST datasets. MNIST is a dataset of hand-written digits,[5] in which each $28 \times 28 \times 1$ image corresponds to a digit. There are 60,000 training digits and 10,000 digits in the test set. MNIST has 10 modes, one for each digit. STACKEDMNIST is formed by concatenating triplets of randomly chosen MNIST digits along the color channel to form images of size $28 \times 28 \times 3$ (Metz et al., 2017). We keep the same size as the original MNIST, 60,000 training digits for 10,000 test digits. The total number of modes in STACKEDMNIST is 1,000, corresponding to the number of possible triplets.

[5] See http://yann.lecun.com/exdb/mnist.

Table 1: Assessing mode collapse on MNIST. The true total number of modes is 10. All methods capture all the 10 modes. The KL captures a notion of discrepancy between the labels of real versus generated images. PresGAN generates images whose distribution of labels is closer to the data distribution, as evidenced by lower KL scores.

Method	Modes	KL
DCGAN (Radford et al., 2015)	10 ± 0.0	0.902 ± 0.036
VEEGAN (Srivastava et al., 2017)	10 ± 0.0	0.523 ± 0.008
PACGAN (Lin et al., 2018)	10 ± 0.0	0.441 ± 0.009
PresGAN (this paper)	$\mathbf{10 \pm 0.0}$	$\mathbf{0.003 \pm 0.001}$

Table 2: Assessing mode collapse on STACKEDMNIST. The true total number of modes is 1,000. All methods suffer from collapse except PresGAN, which captures nearly all the modes of the data distribution. Furthermore, PresGAN generates images whose distribution of labels is closer to the data distribution, as evidenced by lower KL scores.

Method	Modes	KL
DCGAN (Radford et al., 2015)	392.0 ± 7.376	8.012 ± 0.056
VEEGAN (Srivastava et al., 2017)	761.8 ± 5.741	2.173 ± 0.045
PACGAN (Lin et al., 2018)	992.0 ± 1.673	0.277 ± 0.005
PresGAN (this paper)	$\mathbf{999.6 \pm 0.489}$	$\mathbf{0.115 \pm 0.007}$

We consider DCGAN as the base architecture and, following Radford et al. (2015), we resize the spatial resolution of images to 64×64 pixels.

To measure the degree of mode collapse we form two diversity metrics, following Srivastava et al. (2017). Both of these metrics require to fit a classifier to the training data. Once the classifier has been fit, we sample S images from the generator. The first diversity metric is the *number of modes captured*, measured by the number of classes that are captured by the classifier. We say that a class k has been captured if there is at least one generated sample for which the probability of being assigned to class k is the largest. The second diversity metric is the KL *divergence* between two discrete distributions: the empirical average of the (soft) output of the classifier on generated images, and the empirical average of the (soft) output of the classifier on real images from the test set. We choose the number of generated images S to match the number of test samples on each dataset. That is, $S = 10{,}000$ for both MNIST and STACKEDMNIST. We expect the KL divergence to be zero if the distribution of the generated samples is indistinguishable from that of the test samples.

We measure the two mode collapse metrics described above against DCGAN (Radford et al., 2015) (the base architecture of PresGAN for this experiment). We also compare against other methods that aim at alleviating mode collapse in GANs, namely, VEEGAN (Srivastava et al., 2017) and PACGAN (Lin et al., 2018). For PresGAN we set the entropy regularization parameter λ to 0.01. We chose the variance thresholds to be $\boldsymbol{\sigma}_{\text{low}} = 0.001$ and $\boldsymbol{\sigma}_{\text{high}} = 0.3$.

Tables 1 and 2 show the number of captured modes and the KL for each method.

Table 3: Assessing the impact of the entropy regularization parameter λ on mode collapse on MNIST and STACKEDMNIST. When $\lambda = 0$ (i.e., no entropy regularization is applied to the generator), then mode collapse occurs as expected. When entropy regularization is applied but the value of λ is very small ($\lambda = 10^{-6}$) then mode collapse can still occur as the level of regularization is not enough. When the value of λ is appropriate for the data then mode collapse does not occur. Finally, when λ is too high then mode collapse can occur because the entropy maximization term dominates and the data is poorly fit.

	MNIST		STACKEDMNIST	
λ	Modes	KL	Modes	KL
0	10 ± 0.0	0.050 ± 0.0035	418.2 ± 7.68	4.151 ± 0.0296
10^{-6}	10 ± 0.0	0.005 ± 0.0008	989.8 ± 1.72	0.239 ± 0.0059
10^{-2}	$\mathbf{10 \pm 0.0}$	$\mathbf{0.003 \pm 0.0006}$	$\mathbf{999.6 \pm 0.49}$	0.115 ± 0.0074
5×10^{-2}	10 ± 0.0	0.004 ± 0.0008	999.4 ± 0.49	$\mathbf{0.099 \pm 0.0047}$
10^{-1}	10 ± 0.0	0.005 ± 0.0004	999.4 ± 0.80	0.102 ± 0.0032
5×10^{-1}	10 ± 0.0	0.006 ± 0.0011	907.0 ± 9.27	0.831 ± 0.0209

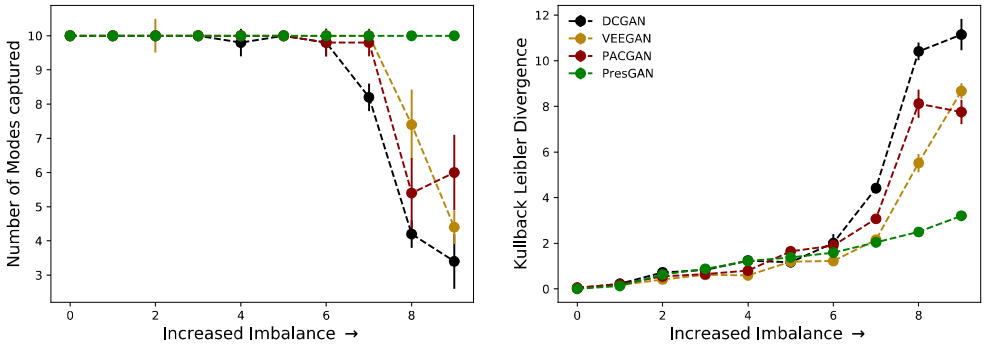

Figure 2: Assessing mode collapse under increased data imbalance on MNIST. The figures show the number of modes captured (higher is better) and the KL divergence (lower is better) under increasingly imbalanced settings. The maximum number of modes in each case is 10. All methods suffer from mode collapse as the level of imbalance increases except for the PresGAN which is robust to data imbalance.

The results are averaged across 5 runs. All methods capture all the modes of MNIST. This is not the case on STACKEDMNIST, where the PresGAN is the only method that can capture all the modes. Finally, the proportion of observations in each mode of PresGAN is closer to the true proportion in the data, as evidenced by lower KL divergence scores.

We also study the impact of the entropy regularization by varying the hyperparameter λ from 0 to 0.5. Table 3 illustrates the results. Unsurprisingly, when there is no entropy regularization, i.e., when $\lambda = 0$, then mode collapse occurs. This is also the case when the level of regularization is not enough ($\lambda = 10^{-6}$). There is a whole range of values for λ such that mode collapse does not occur ($\lambda \in \{0.01, 0.05, 0.1\}$). Finally, when λ is too high for the data and architecture under study, mode collapse can still occur. This is because when λ is too high, the entropy regularization term dominates the loss in Eq. 5 and in turn the generator does not fit the data as well.

This is also evidenced by the higher KL divergence score when $\lambda = 0.5$ vs. when $0 < \lambda < 0.5$.

Increased data imbalance. We now show that mode collapse can occur in GANs when the data is imbalanced, even when the number of modes of the data distribution is small. We follow Dieng et al. (2018a) and consider a perfectly balanced version of MNIST as well as nine imbalanced versions. To construct the balanced dataset we used 5,000 training examples per class, totaling 50,000 training examples. We refer to this original balanced dataset as D_0. Each additional training set D_k leaves only 5 training examples for each class $j \leq k$, and 5,000 for the rest. (See the Appendix for all the class distributions.)

We used the same classifier trained on the unmodified MNIST but fit each method on each of the 9 new MNIST distributions. We chose $\lambda = 0.1$ for PresGAN. Figure 2 illustrates the results in terms of both metrics—number of modes and KL divergence. DCGAN, VEEGAN, and PACGAN face mode collapse as the level of imbalance increases. This is not the case for PresGAN, which is robust to imbalance and captures all the 10 modes.

5.3 Assessing sample quality

In this section we assess PresGANs' ability to generate samples of high perceptual quality. We rely on perceptual quality of generated samples and on Fréchet Inception distance (FID) scores (Heusel et al., 2017). We also consider two different GAN architectures, the standard DCGAN and the more recent StyleGAN, to show robustness of PresGANs vis-a-vis the underlying GAN architecture.

DCGAN. We use DCGAN (Radford et al., 2015) as the base architecture and build PresGAN on top of it. We consider four datasets: MNIST, STACKEDMNIST, CIFAR-10, and CelebA. CIFAR-10 (Krizhevsky et al., 2009) is a well-studied dataset of 32×32 images that are classified into one of the following categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. CelebA (Liu et al., 2015) is a large-scale face attributes dataset. Following Radford et al. (2015), we resize all images to 64×64 pixels. We use the default DCGAN settings. We refer the reader to the code we used for DCGAN, which was taken from https://github.com/pytorch/examples/tree/master/dcgan. We set the seed to 2019 for reproducibility.

There are hyperparameters specific to PresGAN. These are the noise and HMC hyperparameters. We set the learning rate for the noise parameters $\boldsymbol{\sigma}$ to 10^{-3} and constrain its values to be between 10^{-3} and 0.3 for all datasets. We initialize $\log \boldsymbol{\sigma}$ to -0.5. We set the burn-in and the number of HMC samples to 2. We choose a standard number of 5 leapfrog steps and set the HMC learning rate to 0.02. The acceptance rate is fixed at 0.67. We found that different λ values worked better for different datasets. We used $\lambda = 5 \times 10^{-4}$ for CIFAR-10 and CELEBA $\lambda = 0.01$ for MNIST and STACKEDMNIST.

We found the PresGAN's performance to be robust to the default settings for most of these hyperparameters. However we found the initialization for $\boldsymbol{\sigma}$ and its learning

Table 4: Fréchet Inception distance (FID) (lower is better). PresGAN has lower FID scores than DCGAN, VEEGAN, and PACGAN. This is because PresGAN mitigates mode collapse while preserving sample quality.

Method	Dataset	FID
DCGAN (Radford et al., 2015)	MNIST	113.129 ± 0.490
VEEGAN (Srivastava et al., 2017)	MNIST	68.749 ± 0.428
PACGAN (Lin et al., 2018)	MNIST	58.535 ± 0.135
PresGAN (this paper)	MNIST	$\mathbf{42.019 \pm 0.244}$
DCGAN	STACKEDMNIST	97.788 ± 0.199
VEEGAN	STACKEDMNIST	86.689 ± 0.194
PACGAN	STACKEDMNIST	117.128 ± 0.172
PresGAN	STACKEDMNIST	$\mathbf{23.965 \pm 0.134}$
DCGAN	CIFAR-10	103.049 ± 0.195
VEEGAN	CIFAR-10	95.181 ± 0.416
PACGAN	CIFAR-10	54.498 ± 0.337
PresGAN	CIFAR-10	$\mathbf{52.202 \pm 0.124}$
DCGAN	CELEBA	39.001 ± 0.243
VEEGAN	CELEBA	46.188 ± 0.229
PACGAN	CELEBA	36.058 ± 0.212
PresGAN	CELEBA	$\mathbf{29.115 \pm 0.218}$

rate to play a role in the quality of the generated samples. The hyperparameters mentioned above for $\boldsymbol{\sigma}$ worked well for all datasets.

Table 4 shows the FID scores for DCGAN and PresGAN across the four datasets. We can conclude that PresGAN generates images of high visual quality. In addition, the FID scores are lower because PresGAN explores more modes than DCGAN. Indeed, when the generated images account for more modes, the FID sufficient statistics (the mean and covariance of the Inception-v3 pool3 layer) of the generated data get closer to the sufficient statistics of the empirical data distribution.

We also report the FID for VEEGAN and PACGAN in Table 4. VEEGAN achieves better FID scores than DCGAN on all datasets but CELEBA. This is because VEEGAN collapses less than DCGAN as evidenced by Table 1 and Table 2. PACGAN achieves better FID scores than both DCGAN and VEEGAN on all datasets but on STACKEDMNIST where it achieves a significantly worse FID score. Finally, PresGAN outperforms all of these methods on the FID metric on all datasets signaling its ability to mitigate mode collapse while preserving sample quality.

Besides the FID scores, we also assess the visual quality of the generated images. In Section 7.3 of the appendix, we show randomly generated (not cherry-picked) images from DCGAN, VEEGAN, PACGAN, and PresGAN. For PresGAN, we show the mean of the conditional distribution of $\mathbf{x}$ given $\mathbf{z}$. The samples generated by PresGAN have high visual quality; in fact their quality is comparable to or better than the DCGAN samples.

StyleGAN. We now consider a more recent GAN architecture (StyleGAN) (Karras et al., 2019) and a higher resolution image dataset (FFHQ). FFHQ is a diverse dataset

Figure 3: Generated images on FFHQ for StyleGAN (left) and PresGAN (right). The PresGAN maintains the high perceptual quality of the StyleGAN.

of faces from Flickr[6] introduced by Karras et al. (2019). The dataset contains 70,000 high-quality PNG images with considerable variation in terms of age, ethnicity, and image background. We use a resolution of 128×128 pixels.

StyleGAN feeds multiple sources of noise $\mathbf{z}$ to the generator. In particular, it adds Gaussian noise after each convolutional layer before evaluating the nonlinearity. Building PresGAN on top of StyleGAN therefore requires to sample all noise variables $\mathbf{z}$ through HMC at each training step. To speed up the training procedure, we only sample the noise variables corresponding to the input latent code and condition on all the other Gaussian noise variables. In addition, we do not follow the progressive growing of the networks of Karras et al. (2019) for simplicity.

For this experiment, we choose the same HMC hyperparameters as for the previous experiments but restrict the variance of the generative distribution to be $\sigma_{\text{high}} = 0.2$. We set $\lambda = 0.001$ for this experiment.

Figure 3 shows cherry-picked images generated from StyleGAN and PresGAN. We can observe that the PresGAN maintains as good perceptual quality as the base architecture. In addition, we also observed that the StyleGAN tends to produce

[6] See https://github.com/NVlabs/ffhq-dataset.

some redundant images (these are not shown in Figure 3), something that we did not observe with the PresGAN. This lack of diversity was also reflected in the FID scores which were 14.72 ± 0.09 for StyleGAN and 12.15 ± 0.09 for PresGAN. These results suggest that entropy regularization effectively reduces mode collapse while preserving sample quality.

5.4 Assessing held-out predictive log-likelihood

In this section we evaluate PresGANs for generalization using predictive log-likelihood. We use the DCGAN architecture to build PresGAN and evaluate the log-likelihood on two benchmark datasets, MNIST and CIFAR-10. We use images of size 32×32.

We compare the generalization performance of the PresGAN against the VAE (Kingma and Welling, 2013; Rezende et al., 2014) by controlling for the architecture and the evaluation procedure. In particular, we fit a VAE that has the same decoder architecture as the PresGAN. We form the VAE encoder by using the same architecture as the DCGAN discriminator and getting rid of the output layer. We used linear maps to get the mean and the log-variance of the approximate posterior.

To measure how PresGANs compare to traditional GANs in terms of log-likelihood, we also fit a PresGAN with $\lambda = 0$.

Evaluation. We control for the evaluation procedure and follow what's described in Section 3.3 for all methods. We use $S = 2{,}000$ samples to form the importance sampling estimator. Since the pixel values are normalized in $[-1, +1]$, we use a truncated Gaussian likelihood for evaluation. Specifically, for each pixel of the test image, we divide the Gaussian likelihood by the probability (under the generative model) that the pixel is within the interval $[-1, +1]$. We use the truncated Gaussian likelihood at test time only.

Settings. For the PresGAN, we use the same HMC hyperparameters as for the previous experiments. We constrain the variance of the generative distribution using $\boldsymbol{\sigma}_{\text{low}} = 0.001$ and $\boldsymbol{\sigma}_{\text{high}} = 0.2$. We use the default DCGAN values for the remaining hyperparameters, including the optimization settings. For the CIFAR-10 experiment, we choose $\lambda = 0.001$. We set all learning rates to 0.0002. We set the dimension of the latent variables to 100. We ran both the VAE and the PresGAN for a maximum of 200 epochs. For MNIST, we use the same settings as for CIFAR-10 but use $\lambda = 0.0001$ and ran all methods for a maximum of 50 epochs.

Results. Table 5 summarizes the results. Here GAN denotes the PresGAN fitted using $\lambda = 0$. The VAE outperforms both the GAN and the PresGAN on both MNIST and CIFAR-10. This is unsurprising given VAEs are fitted to maximize log-likelihood. The GAN's performance on CIFAR-10 is particularly bad, suggesting it suffered from mode collapse. The PresGAN, which mitigates mode collapse achieves significantly better performance than the GAN on CIFAR-10. To further analyze the generalization performance, we also report the log-likelihood on the training set in Table 5. We can observe that the difference between the training log-likelihood and the test log-likelihood is very small for all methods.

Table 5: Generalization performance as measured by negative log-likelihood (lower is better) on MNIST and CIFAR-10. Here the GAN denotes a PresGAN fitted without entropy regularization ($\lambda = 0$). The PresGAN reduces the gap in performance between the GAN and the VAE on both datasets.

	MNIST		CIFAR-10	
	Train	Test	Train	Test
VAE	−3483.94	−3408.16	−1978.91	−1665.84
GAN	−1410.78	−1423.39	−572.25	−569.17
PresGAN	−1418.91	−1432.50	−1050.16	−1031.70

6 Epilogue

We introduced the PresGAN, a variant of GANs that addresses two of their limitations. PresGANs prevent mode collapse and are amenable to predictive log-likelihood evaluation. PresGANs model data by adding noise to the output of a density network and optimize an entropy-regularized adversarial loss. The added noise stabilizes training, renders approximation of predictive log-likelihoods tractable, and enables unbiased estimators for the gradients of the entropy of the generative distribution. We evaluated PresGANs on several image datasets. We found they effectively prevent mode collapse and generate samples of high perceptual quality. We further found that PresGANs reduce the gap in performance between GANs and VAEs in terms of predictive log-likelihood.

We found the level of entropy regularization λ plays an important role in mode collapse. We leave as future work the task of finding the optimal λ. We now discuss some insights that we concluded from our empirical study in Section 5.

Implicit distributions and sample quality. It's been traditionally observed that GANs generate samples with higher perceptual quality than VAEs. This can be explained by looking at the two ways in which GANs and VAEs differ; the generative distribution and the objective function. VAEs use prescribed generative distributions and optimize likelihood whereas GANs use implicit generative distributions and optimize an adversarial loss. Our results in Section 5 suggest that the implicit generators of traditional GANs are not the key to high sample quality; rather, the key is the adversarial loss. This is because PresGANs use the same prescribed generative distributions as VAEs and achieve similar or sometimes better sample quality than GANs.

Mode collapse, diversity, and imbalanced data. The current literature on measuring mode collapse in GANs only focuses on showing that mode collapse happens when the number of modes in the data distribution is high. Our results show that mode collapse can happen not only when the number of modes of the data distribution is high, but also when the data is imbalanced; even when the number of modes is low. Imbalanced data are ubiquitous. Therefore, mitigating mode collapse in GANs is important for the purpose of diverse data generation.

GANs and generalization. The main method to evaluate generalization for density estimators is predictive log-likelihood. Our results agree with the current literature

that GANs don't generalize as well as VAEs which are specifically trained to maximize log-likelihood. However, our results show that entropy-regularized adversarial learning can reduce the gap in generalization performance between GANs and VAEs. Methods that regularize GANs with the maximum likelihood objective achieve good generalization performance when compared to VAEs but they sacrifice sample quality when doing so (Grover et al., 2018). In fact we also experienced this tension between sample quality and high log-likelihood in practice.

Why is there such a gap in generalization, as measured by predictive log-likelihood, between GANs and VAEs? In our empirical study in Section 5 we controlled for the architecture and the evaluation procedure which left us to compare maximizing likelihood against adversarial learning. Our results suggest mode collapse alone does not explain the gap in generalization performance between GANs and VAEs. Indeed Table 5 shows that even on MNIST, where mode collapse does not happen, the VAE achieves significantly better log-likelihood than a GAN.

We looked more closely at the encoder fitted at test time to evaluate log-likelihood for both the VAE and the GAN (not shown in this paper). We found that the encoder implied by a fitted GAN is very underdispersed compared to the encoder implied by a fitted VAE. Underdispersed proposals have a negative impact on importance sampling estimates of log-likelihood. We tried to produce a more overdispersed proposal using the procedure described in Section 3.3. However we leave as future work learning overdispersed proposals for GANs for the purpose of log-likelihood evaluation.

Acknowledgements

We thank Ian Goodfellow, Andriy Mnih, Aaron Van den Oord, and Laurent Dinh for their comments. Francisco J. R. Ruiz is supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 706760. Adji B. Dieng is supported by a Google PhD Fellowship.

References

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223.

Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (GANs). In *International Conference on Machine Learning*.

Arora, S., Risteski, A., and Zhang, Y. (2018). Do gans learn the distribution? some theory and empirics.

Azadi, S., Olsson, C., Darrell, T., Goodfellow, I., and Odena, A. (2018). Discriminator rejection sampling. *arXiv preprint arXiv:1810.06758*.

Balaji, Y., Hassani, H., Chellappa, R., and Feizi, S. (2018). Entropic GANs meet VAEs: A statistical approach to compute sample likelihoods in gans. *arXiv preprint arXiv:1810.04147*.

Belghazi, M. I., Baratin, A., Rajeswar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, R. D. (2018). Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*.

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. (2018). Demystifying MMD GANs. *arXiv:1801.01401*.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015). Generating sentences from a continuous space. *arXiv:1511.06349*.

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *arXiv:1809.11096*.

Cao, Y., Ding, G. W., Lui, K. Y.-C., and Huang, R. (2018). Improving gan training via binarized representation entropy (bre) regularization. *arXiv preprint arXiv:1805.03644*.

Dieng, A. B., Cho, K., Blei, D. M., and LeCun, Y. (2018a). Learning with reflective likelihoods.

Dieng, A. B., Kim, Y., Rush, A. M., and Blei, D. M. (2018b). Avoiding latent variable collapse with generative skip models. *arXiv:1807.04863*.

Dieng, A. B. and Paisley, J. (2019). Reweighted expectation maximization. *arXiv preprint arXiv:1906.05850*.

Diggle, P. J. and Gratton, R. J. (1984). Monte Carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(2):193–212.

Donahue, J., Krähenbühl, P., and Darrell, T. (2016). Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. (2016). Adversarially learned inference. *arXiv preprint arXiv:1606.00704*.

Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. (2015). Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Genevay, A., Peyré, G., and Cuturi, M. (2017). Learning generative models with sinkhorn divergences. *arXiv preprint arXiv:1706.00292*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Grover, A., Dhar, M., and Ermon, S. (2018). Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*.

Huszár, F. (2016). Instance noise: a trick for stabilising gan training. https://www.inference.vc/instance-noise-a-trick-for-stabilising-gan-training/.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge university press.

Jordan, M. I. (1998). *Learning in graphical models*, volume 89. Springer Science & Business Media.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Conference on Computer Vision and Pattern Recognition*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer.

Kumar, R., Goyal, A., Courville, A., and Bengio, Y. (2019). Maximum entropy generators for energy-based models. *arXiv preprint arXiv:1901.08508*.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

Li, Y., Swersky, K., and Zemel, R. (2015). Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727.

Lin, Z., Khetan, A., Fanti, G., and Oh, S. (2018). Pacgan: The power of two samples in generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1498–1507.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738.

MacKay, D. J. (1995). Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 354(1):73–80.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Mescheder, L., Nowozin, S., and Geiger, A. (2017). Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2391–2400. JMLR. org.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. (2017). Unrolled generative adversarial networks. In *International Conference on Learning Representations*.

Minka, T. et al. (2005). Divergence measures and message passing. Technical report, Technical report, Microsoft Research.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Mohamed, S. and Lakshminarayanan, B. (2016). Learning in implicit generative models. *arXiv:1610.03483*.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and computing*, 11(2):125–139.

Neal, R. M. et al. (2011). Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279.

Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Ravuri, S., Mohamed, S., Rosca, M., and Vinyals, O. (2018). Learning implicit generative models with the method of learned moments. *arXiv preprint arXiv:1806.11006*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning*.

Rigollet, P. and Weed, J. (2018). Entropic optimal transport is maximum-likelihood deconvolution. *Comptes Rendus Mathématique*, 356(11-12):1228–1235.

Rosca, M., Lakshminarayanan, B., Warde-Farley, D., and Mohamed, S. (2017). Variational approaches for auto-encoding generative adversarial networks. *arXiv:1706.04987*.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. In *Advances in neural information processing systems*.

Sánchez-Martín, P., Olmos, P. M., and Pérez-Cruz, F. (2019). Out-of-sample testing for gans. *arXiv preprint arXiv:1901.09557*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897.

Sønderby, C. K., Caballero, J., Theis, L., Shi, W., and Huszár, F. (2016). Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*.

Soofi, E. S. (2000). Principal information theoretic approaches. *Journal of the American Statistical Association*, 95(452):1349–1353.

Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., and Sutton, C. (2017). Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318.

Titsias, M. and Lázaro-Gredilla, M. (2014). Doubly stochastic variational bayes for non-conjugate inference. In *International conference on machine learning*, pages 1971–1979.

Titsias, M. K. and Ruiz, F. J. (2018). Unbiased implicit variational inference. *arXiv preprint arXiv:1808.02078*.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2017). Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*.

Turner, R., Hung, J., Saatci, Y., and Yosinski, J. (2018). Metropolis-hastings generative adversarial networks. *arXiv preprint arXiv:1811.11357*.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2018). It takes (only) two: Adversarial generator-encoder networks. In *AAAI Conference on Artificial Intelligence*.

Wainwright, M. J., Jordan, M. I., et al. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Wu, Y., Burda, Y., Salakhutdinov, R., and Grosse, R. (2016). On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*.

Xiao, C., Zhong, P., and Zheng, C. (2018). BourGAN: Generative networks with metric embeddings. In *Advances in Neural Information Processing Systems*.

Yin, M. and Zhou, M. (2019). Semi-implicit generative model. *arXiv preprint arXiv:1905.12659*.

Zhang, P., Liu, Q., Zhou, D., Xu, T., and He, X. (2017). On the discrimination-generalization tradeoff in gans. *arXiv preprint arXiv:1711.02771*.

Appendix

7.1 Other Ways to Compute Predictive Log-Likelihood

Here we discuss different ways to obtain a proposal in order to approximate the predictive log-likelihood. For a test instance $\mathbf{x}^*$, we estimate the marginal log-likelihood $\log p_\theta(\mathbf{x}^*)$ using importance sampling,

$$\log p_\theta(\mathbf{x}^*) \approx \log\left(\frac{1}{S}\sum_{s=1}^{S}\frac{p_\theta\left(\mathbf{x}^* \mid \mathbf{z}^{(s)}\right)\, p\left(\mathbf{z}^{(s)}\right)}{r\left(\mathbf{z}^{(s)} \mid \mathbf{x}^*\right)}\right), \tag{21}$$

where we draw the S samples $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(S)}$ from a proposal distribution $r(\mathbf{z}|\mathbf{x}^*)$. We next discuss different ways to form the proposal $r(\mathbf{z}|\mathbf{x}^*)$.

One way to obtain the proposal is to set $r(\mathbf{z}|\mathbf{x}^*)$ as a Gaussian distribution whose mean and variance are computed using samples from an HMC algorithm with stationary distribution $p_\theta(\mathbf{z}|\mathbf{x}^*) \propto p_\theta(\mathbf{x}^*|\mathbf{z})p(\mathbf{z})$. That is, the mean and variance of $r(\mathbf{z}|\mathbf{x}^*)$ are set to the empirical mean and variance of the HMC samples.

The procedure above requires to run an HMC sampler, and thus it may be slow. We can accelerate the procedure with a better initialization of the HMC chain. Indeed, the second way to evaluate the log-likelihood also requires the HMC sampler, but it is initialized using a mapping $\mathbf{z} = g_\eta(\mathbf{x}^\star)$. The mapping $g_\eta(\mathbf{x}^\star)$ is a network that maps from observed space $\mathbf{x}$ to latent space $\mathbf{z}$. The parameters η of the network can be learned at test time using generated data. In particular, η can be obtained by generating data from the fitted generator of PresGAN and then fitting $g_\eta(\mathbf{x}^\star)$ to map $\mathbf{x}$ to $\mathbf{z}$ by maximum likelihood. This is, we first sample M pairs $(\mathbf{z}_m, \mathbf{x}_m)_{m=1}^M$ from the learned generative distribution and then we obtain η by minimizing $\sum_{m=1}^M ||\mathbf{z}_m - g_\eta(\mathbf{x}_m)||_2^2$. Once the mapping is fitted, we use it to initialize the HMC chain.

A third way to obtain the proposal is to learn an encoder network $q_\eta(\mathbf{z}|\mathbf{x})$ jointly with the rest of the PresGAN parameters. This is effectively done by letting the discriminator distinguish between pairs $(\mathbf{x}, \mathbf{z}) \sim p_d(\mathbf{x}) \cdot q_\eta(\mathbf{z}|\mathbf{x})$ and $(\mathbf{x}, \mathbf{z}) \sim p_\theta(\mathbf{x}, \mathbf{z})$ rather than discriminate $\mathbf{x}$ against samples from the generative distribution. These types of discriminator networks have been used to learn a richer latent space for GAN (Donahue et al., 2016; Dumoulin et al., 2016). In such cases, we can use the encoder network $q_\eta(\mathbf{z}|\mathbf{x})$ to define the proposal, either by setting $r(\mathbf{z}|\mathbf{x}^*) = q_\eta(\mathbf{z}|\mathbf{x}^*)$ or by initializing the HMC sampler at the encoder mean.

The use of an encoder network is appealing but it requires a discriminator that takes pairs $(\mathbf{x}, \mathbf{z})$. The approach that we follow in the paper also uses an encoder network but keeps the discriminator the same as for the base DCGAN. We found this approach to work better in practice. More in detail, we use an encoder network $q_\eta(\mathbf{z}|\mathbf{x})$; however the encoder is fitted at test time by maximizing the variational ELBO, given

by $\sum_n \mathbb{E}_{q_\eta(\mathbf{z}_n|\mathbf{x}_n)}\big[\log p_\theta(\mathbf{x}_n, \mathbf{z}_n) - \log q_\eta(\mathbf{z}_n|\mathbf{x}_n)\big]$. We set the proposal $r(\mathbf{z}|\mathbf{x}^*) = q_\eta(\mathbf{z}|\mathbf{x}^*)$. (Alternatively, the encoder can be used to initialize a sampler.)

7.2 Assessing mode collapse under increased data imbalance

In the main paper we show that mode collapse can happen not only when there are increasing number of modes, as done in the GAN literature, but also when the data is imbalanced. We consider a perfectly balanced version of MNIST by using 5,000 training examples per class, totalling 50,000 training examples. We refer to this original balanced dataset as **D1**. We build nine additional training sets from this balanced dataset. Each additional training set **D***k* leaves only 5 training examples for each class $j < k$. See Table 6 for all the class distributions.

Table 6: Class distributions using the MNIST dataset. There are 10 class—one class for each of the 10 digits in MNIST. The distribution D1 is uniform and the other distributions correspond to different imbalance settings as given by the proportions in the table. Note these proportions might not sum to one exactly because of rounding.

Dist	0	1	2	3	4	5	6	7	8	9
D1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
D2	10^{-3}	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11
D3	10^{-3}	10^{-3}	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12
D4	10^{-3}	10^{-3}	10^{-3}	0.14	0.14	0.14	0.14	0.14	0.14	0.14
D5	10^{-3}	10^{-3}	10^{-3}	10^{-3}	0.17	0.17	0.17	0.17	0.17	0.17
D6	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	0.20	0.20	0.20	0.20	0.20
D7	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	0.25	0.25	0.25	0.25
D8	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	0.33	0.33	0.33
D9	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	0.49	0.49
D10	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	0.99

7.3 Sample quality

Here we show some sample images generated by DCGAN and PresGAN, together with real images from each dataset. These images were not cherry-picked, we randomly selected samples from all models. For PresGAN, we show the mean of the generator distribution, conditioned on the latent variable z. In general, we observed the best image quality is achieved by the entropy-regularized PresGAN.

(a) Real images.

(b) DCGAN Samples

(c) VEEGAN Samples

(d) PACGAN Samples

(e) PresGAN Samples

Figure 4: Real and generated images on CELEBA.