Adversarial Likelihood Estimation With One-Way Flows

Omri Ben-Dov[1], Pravir Singh Gupta[2], Victoria Abrevaya[1], Michael J. Black[1], Partha Ghosh[1]
[1]Max Planck Institute for Intelligent Systems, Tübingen, Germany [2]Perceive Inc.
{odov, vabrevaya, black, pghosh}@tuebingen.mpg.de, pravir.singh.gupta@gmail.com

Abstract

Generative Adversarial Networks (GANs) can produce high-quality samples, but do not provide an estimate of the probability density around the samples. However, it has been noted that maximizing the log-likelihood within an energy-based setting can lead to an adversarial framework where the discriminator provides unnormalized density (often called energy). We further develop this perspective, incorporate importance sampling, and show that 1) Wasserstein GAN performs a biased estimate of the partition function, and we propose instead to use an unbiased estimator; and 2) when optimizing for likelihood, one must maximize generator entropy. This is hypothesized to provide a better mode coverage. Different from previous works, we explicitly compute the density of the generated samples. This is the key enabler to designing an unbiased estimator of the partition function and computation of the generator entropy term. The generator density is obtained via a new type of flow network, called one-way flow network, that is less constrained in terms of architecture, as it does not require a tractable inverse function. Our experimental results show that our method converges faster, produces comparable sample quality to GANs with similar architecture, successfully avoids over-fitting to commonly used datasets and produces smooth low-dimensional latent representations of the training data.

1. Introduction

The goal of a generative model is to extract some notion of the data distribution given a training set, either explicitly by computing the probability density function [44], indirectly through distilling a stochastic sampling mechanism [12], or a combination of both [9]. While indirect generative models can achieve state-of-the-art performance in sample quality [12, 17], having explicit densities has several advantages. For example, an explicit density function can be used to quantitatively compare models, or to train models by maximum likelihood estimation (MLE), which has been proven to be statistically asymptotically efficient [15].

Autoregressive models [43, 44] and normalizing flows [8] are the most prominent examples of deep generative models that compute exact probability and directly maximize the log-likelihood of their training dataset. However, it is inefficient to sample from autoregressive models and they do not provide a low-dimensional latent representation of the data. Normalizing flows allow both efficient sampling and density estimation, but make restrictive assumptions on the architecture, requiring the latent space to be of the same dimensionality as that of the input, making it computationally expensive to use in a high-dimensional data regime.

Energy-based models (EBMs) [41], variational autoencoders (VAEs) [22] and diffusion models [32,38] are further examples of deep generative models trained with likelihood maximization. However, VAEs and diffusion models can only compute a lower bound of the likelihood. EBMs, on the other hand, represent an unnormalized density, allowing for greater flexibility in the choice of functional form, at the cost of inefficient sampling and approximate likelihood estimation.

Indirect models such as Generative Adversarial Networks (GANs) [12, 17] have achieved state-of-the-art performance in terms of the quality of the generated data, but do not provide any estimate of the probability density around a sample. However, a connection has been noted between the loss function of these networks, in particular the Wasserstein GAN (WGAN) loss [2] and EBMs [5, 19], in which the discriminator can be regarded as an energy function. With the goal of introducing density estimation within an adversarial training framework, we follow here a similar path, but develop further these observations to arrive at an *unbiased* estimator of the partition function through the explicit computation of the generator density.

Specifically, we begin by exploring the connection between EBMs and GANs which leads to a training objective that closely resembles the WGAN loss, with minor but key differences. We notice that maximizing the log-likelihood of an EBM arrives at the WGAN loss if we take a biased estimation of the normalization constant of the energy function; or alternatively, WGANs perform a one-sample ap-

proximation of the partition function. Based on this observation, and in departure from previous work, we propose to use an unbiased estimator by explicitly computing the generator density P_{G_ψ}.

To calculate P_{G_ψ}, we propose a new type of normalizing flow network that bypasses several architectural constraints found in standard flow models. In particular, we construct a flow that can perform upsampling and downsampling operations, starting from a lower-dimensional latent variable, at the cost of approximate probability computation. This is possible and sufficient since we only need to compute P_{G_ψ} for generated samples, while density estimation of real, non-generated points is relegated to the discriminator.

Our experimental results show that our model is able to capture more modes, trains faster on images, produces comparable sample quality to GANs with similar architecture, and can be used to compute the partition function with a practical number of samples.

In summary, we propose a framework for adversarial generative modeling that simultaneously computes an estimate of the density, with the following key contributions: i) by developing the connection between EBMs and GANs, we show that the WGAN discriminator objective is a biased estimator of the partition function; ii) we propose an unbiased estimate of the partition function of an EBM by explicitly computing the density of the generator; iii) we propose a new flow-based network for the computation of the generator density that enables a more flexible architecture, in contrast to traditional flow models.

2. Related work

Two main categories of generative models are *prescribed* and *implicit* models [7]. Prescribed models recover an explicit parametric specification of the density function and are trained and evaluated by MLE; our work belongs to this family. Implicit models, on the other hand, represent the data distribution indirectly through a stochastic mechanism that generates random samples. In general, this offers more flexibility in terms of learning objective and model architecture, which is hypothesized to be responsible for the high visual quality of the generated samples.

Normalizing flows [9, 21, 24] and autoregressive models [35, 42, 44] are examples of deep *prescribed* generative models. Since these compute the density function explicitly, they can be optimized and evaluated using the train- and test-set log-likelihood. Although autoregressive models can efficiently work with high-dimensional data during training, due to ancestral sampling they are extremely slow at generating new samples. Normalizing flows require an invertible architecture to compute the likelihood, and consequently can only support latent spaces of the same dimensionality as the input data. In addition, they tend to produce large and memory-hungry models, and are therefore not so suitable for high-dimensional data. In this work, we relax the invertibility constraint by computing the flow in only one direction, enabling the use of lower-dimensional latent vectors, and more resource-efficient architectures.

An intermediate category of generative models considers only an *approximation* to the density function. Examples include a lower bound on the likelihood for VAEs and diffusion models [14], or the unnormalized density in the case of EBMs [39]. VAEs are known to suffer from low generation quality, *i.e.* they tend to produce blurry samples. Diffusion models can generate images of very high sample-quality [6, 32]; however, the latent representation needs to be of the same dimension as the input data. EBMs [41] deploy several techniques to obtain the derivative of the normalizing factor with respect to the model parameters. We maximize the same cost function as EBMs (see Eq. (3)), but explicitly model the normalization constant ζ.

GANs [12] are the most prominent example of *implicit* models, and produce state-of-the-art generated sample quality [18]. However, it has been observed that GANs may trade diversity for precision [3, 34, 40]. This results in generators that produce samples from only a few modes of the data distribution, a phenomenon known as "mode collapse". GANs are also well known for having unstable training dynamics [2, 13, 28].

The connection between GANs and EBMs on which we base our analysis has been previously observed in [4, 5, 19]. A common assumption by these works is that the generator density is inaccessible, which forces them to work with a biased partition function. Furthermore, while designing the objective for the generator network, it is observed (as in this work) that the entropy of the generator distribution needs to be maximized. However, entropy estimation is closely related to density estimation, and therefore as hard as the original problem. To address this, [19] assumes that the batch normalization layer maps every intermediate activation to approximately normal distributions, and that the sum of the analytical entropy of these distributions approximate the true generator entropy. In [5] two different approaches are proposed for the generator distribution: 1) assume that the distribution is a mixture of isotropic Gaussians centered around the generator response $G_\psi(z)$, and compute the gradient of such a mixture; and 2) compute its variational lower bound, which requires training yet another network that outputs a parametric form of the approximate posterior and an MCMC integration over the noise variable. A similar approach using variational lower bound has also been explored in [1]. Contrary to these, we explicitly compute the generator density and with it the entropy term, resulting in an unbiased estimate of the partition function.

3. Method

3.1. Density estimation by MLE

Given a dataset of independent and identically distributed samples $\mathcal{X}:=\{x_i \in \mathbb{R}^n\}_{i=1}^m$, drawn from an unknown probability distribution P_{data}, our goal is to learn a parametric model P_{D_θ} that matches the distribution of P_{data}. Following EBMs [39, 41], we define P_{D_θ} as

$$P_{D_\theta}(x) = \frac{e^{D_\theta(x)}}{\zeta}. \tag{1}$$

Here, D_θ is a neural network with parameters θ. The exponentiation ensures a non-negative probability, and $\zeta = \int_{\mathbb{R}^n} e^{D_\theta(x)} \mathrm{d}x$ is a normalizing factor such that P_{D_θ} integrates to unity. Note that traditionally the energy of an EBM is represented as $e^{-D_\theta(x)}$, but here we consume the negative sign inside the $D_\theta(x)$ function for notational simplicity.

Since the partition function ζ is generally intractable and hard to compute, we approximate this integral with importance sampling [23], and rewrite it as

$$\begin{aligned}\zeta &= \int_{\mathbb{R}^n} P_{G_\psi}(x) \frac{e^{D_\theta(x)}}{P_{G_\psi}(x)} \mathrm{d}x = \mathbb{E}_{x\sim P_{G_\psi}}\left[\frac{e^{D_\theta(x)}}{P_{G_\psi}(x)}\right] \\ &\approx \frac{1}{S}\sum_{x\sim P_{G_\psi}} \frac{e^{D_\theta(x)}}{P_{G_\psi}(x)},\end{aligned} \tag{2}$$

with S representing the number of samples used in the summation, and where P_{G_ψ} is an arbitrary distribution that is non-zero in the integration domain, often called the biased density. Here, we choose P_{G_ψ} to be the push-forward density through a neural network $G_\psi : \mathbb{R}^d \to \mathbb{R}^n$ with parameters ψ, such that $y = G_\psi(z)$ is a sample from the biased density P_{G_ψ}, and $z \in \mathbb{R}^d$, $z \sim P(Z)$ is the latent random variable, with $d \leq n$. We further choose $P(Z)$ to be the standard normal density $\mathcal{N}(0, I)$. We will elaborate more on P_{G_ψ} in Sec. 3.3, and on G_ψ in Sec. 3.4.

We train D_θ by maximizing the log-likelihood $\log P_D(x)$ of the dataset $\mathcal{X}$:

$$\begin{aligned}\theta^* &= \arg\max_\theta \left\{\sum_{x\in\mathcal{X}} \log P_{D_\theta}(x)\right\} \\ &= \arg\max_\theta \left\{\sum_{x\in\mathcal{X}} \left(D_\theta(x) - \log\zeta\right)\right\} \\ &\approx \arg\max_\theta \left\{\sum_{x\in\mathcal{X}} \left[D_\theta(x) - \log \sum_{y\sim P_{G_\psi}} \frac{e^{D_\theta(y)}}{P_{G_\psi}(y)}\right]\right\}.\end{aligned} \tag{3}$$

The summation over $y \sim P_{G_\psi}$ is the ζ integral approximation from Eq. (2), summed over S samples. A full derivation can be found in Appendix A.1. Interestingly, if we take a one-sample approximation we get the objective

$$\begin{aligned}\theta^* &= \arg\max_\theta \left\{\sum_{x\in\mathcal{X}} \left[D_\theta(x) - \log e^{D_\theta(y)} + \log P_{G_\psi}(y)\right]\right\} \\ &= \arg\max_\theta \left\{\sum_{x\in\mathcal{X}} \left[D_\theta(x) - D_\theta(G_\psi(z))\right]\right\}.\end{aligned} \tag{4}$$

Here the term $\log P_{G_\psi}(y)$ can be discarded because it does not depend on θ. Eq. (4) is exactly the objective for the WGAN discriminator [2]. Hence, we note a connection between the unnormalized log-density estimator D_θ and the WGAN objective function, which can be re-interpreted as performing a one-sample approximation of ζ within an energy-based framework.

There is an alternative view to Eq. (4). If we simply drop the importance sampling scheme and embrace a biased estimate of the partition function, we again recover the WGAN objective as was shown in [5]. The introduction of the importance weights as in Eq. (3) is known to produce an unbiased estimator [33] of the normalizing constant ζ, although it does so at the cost of added variance. Since it is intractable to theoretically compute the variance of this estimator, even when we have access to the variance of the importance weight, we will show the empirical relevance of the unbiased estimator in Sec. 4.

3.2. Learning P_{G_ψ} for importance sampling

The construction of P_{G_ψ} in Eqs. (2) and (3) is important, since an appropriate choice can dramatically reduce the number of samples required to achieve an accurate approximation of ζ.

To reduce the number of samples needed we minimize the variance of the approximation error, which is proportional to $\frac{P_{D_\theta}}{P_{G_\psi}}$ [33]. This occurs when P_{G_ψ} matches P_{D_θ} up to a multiplicative factor. Therefore, we train G_ψ by minimizing the KL-divergence [26] between the two distributions, leading to the objective function:

$$\psi^* = \arg\max_\psi \left\{H\left(G_\psi(Z)\right) + \frac{1}{m}\sum_{z\sim Z} D_\theta\left(G_\psi(z)\right)\right\}, \tag{5}$$

where Z is a random variable that is used as input to G_ψ and $H\left(P_{G_\psi}(Z)\right)$ is the entropy of the generator distribution. The full derivation is in Appendix A.2. [1]

Notably, we obtain in Eq. (5) the WGAN *generator* objective, with an additional entropy term $H\left(P_{G_\psi}(Z)\right)$ that

[1] In practice, the entropy $H\left(G_\psi(Z)\right)$ is not the same order of magnitude as the discriminator response, hence we add a weight w to the entropy term. We mathematically justify this and correct the objective and probability using this weight in Appendix C.1.

requires maximization. We hypothesize that this term is responsible for ensuring diversity in the generated samples, and that its introduction can reduce the well-known problem of mode collapse in GANs. This has also been observed in [5, 19], where the authors proposed ad-hoc solutions to the computation of $H\left(P_{G_\psi}(Z)\right)$ since the distribution of P_{G_ψ} was unknown.

Note that any other choice of divergence is in principle valid as objective function. We take here the KL-divergence because it leads to an objective for G_ψ that is independent of the normalizing constant ζ of the distribution given by D_θ, and because it leads to a natural connection with the WGAN loss. We leave the exploration of other divergences for future work.

3.3. Estimating probabilities for the generator

We require a tractable $P_{G_\psi}(y)$ for the approximation of the integral in Eq. (3). One design option that fits this requirement is a normalizing flow network [9], where the density at a point y sampled using the generator network $G_\psi(z)$ is computed using the change of variables formula:

$$P\left(G_\psi\left(z\right)\right) = P_Z\left(z\right)\left|\det\left(\frac{\partial G_\psi\left(z\right)}{\partial z^\intercal}\right)\right|^{-1}, \tag{6}$$

with P_Z the latent density from which z is sampled, and $\frac{\partial G_\psi(z)}{\partial z^\intercal}$ the Jacobian of $G_\psi\left(z\right)$.

Normalizing flows require the mapping G_ψ to be bijective for the change of variable formula given by Eq. (6) to hold. Additionally, for any $x \in \mathcal{X}$, normalizing flows must find a z such that $G_\psi\left(z\right) = x$. This requires G_ψ to be designed in such a way that it can be efficiently inverted, which greatly restricts the choice of architecture of G_ψ, and prevents from adopting the recent progress made by empirical research on GAN architectures [18, 31].

In our setting, however, we need to evaluate the generator density only at points *sampled from the generator* $y = G_\psi(z)$. Therefore, we do not need to compute the inverse function $G_\psi^{-1}(x)$ explicitly, only the forward $G_\psi(z)$ and its Jacobian determinant. This allows to use any architecture for G_ψ whose Jacobian determinant can be computed efficiently. In Section 3.4 we show how to build such architecture, which we call *one-way flow*.

3.4. One-way flow generator network

Motivated by the generation quality of GANs we design a generator that maps a latent space ($\mathbb{R}^d$) to the data space ($\mathbb{R}^n$) with $d \ll n$, gradually increasing dimensionality while retaining computational efficiency in the estimation of the density.

First, we define a function $g_u : \mathbb{R}^d \to \mathbb{R}^n$ that increases dimensionality by concatenating a random vector $r \in \mathbb{R}^{n-d}$ as $g_u\left(z\right) = \begin{pmatrix} z \\ r \end{pmatrix}$. Since r and z are independent by design and since $P\left(r\right)$ and $P\left(z\right)$ are known, the probability of the output is

$$P\left(g_u\left(z\right)\right) = P\left(z\right)P\left(r\right). \tag{7}$$

We can compose any number of functions that are either bijective or concatenate random noise as in g_u, although in practice we did not find a need to use more than one such layer. We encapsulate the subsequent layers into a function of the form $g_n : \mathbb{R}^n \to \mathbb{R}^n$. For the encapsulated layers, we allow any architecture design, with the constraint that nowhere inside this part of the network the dimension of the activation be smaller than n.

Finally, we construct the generator as $G_\psi\left(z\right) = \left(g_n \circ g_u\right)\left(z\right)$. The form $G_\psi : \mathbb{R}^d \to \mathbb{R}^n$ makes it possible to compute the probability $P_{G_\psi}\left(G_\psi\left(z\right)\right)$ of an n-dimensional sample $G_\psi\left(z\right)$ on its corresponding d-dimensional point z on the manifold. It is unnecessary to compute $P_{G_\psi}\left(x\right)$ on any arbitrary n-dimensional point x, since our model requires only probabilities of generated samples.

However, computing the Jacobian of g_n and its determinant in high dimensions is a computationally heavy task. To efficiently approximate the determinant of the Jacobian we use the equality

$$\left|J\right|^{-1} = \mathbb{E}_{v\sim S^{n-1}}\left[\left\|Jv\right\|^{-n}\right] \tag{8}$$

from [37], where $J \in \mathbb{R}^{n\times n}$ is a matrix, *i.e.* the Jacobian, and v is a random unit vector. To further increase efficiency, we use a one-sample approximation, which allows us to rewrite it in log form as

$$\log\left|J\right| \approx n\log\left\|Jv\right\|. \tag{9}$$

We show in Appendix B that using this form is sufficient for our purposes.

For our definition of G_ψ, with Eqs. (6) and (7), we get the computationally efficient form of the generator density evaluated at a generated point as follows:

$$P_{G_\psi}\left(G_\psi\left(z\right)\right) = P\left(z\right)P\left(r\right)\left|\det\left(\frac{\partial g_n\left(z\right)}{\partial z^\intercal}\right)\right|^{-1}. \tag{10}$$

Using the approximation of Eq. (9) we obtain a computationally efficient unbiased estimator of the entropy using an m-sample empirical mean as

$$H\left(G_\psi\left(z\right)\right) \approx -\frac{1}{m}\sum_{z\sim P_Z}\left[\log\left(P\left(z\right)P\left(r\right)\right) - n\log\left\|Jv\right\|\right]. \tag{11}$$

This in turn lets us write the generator objective as

$$\psi^* = \arg\max_\psi\left\{\sum_{z\sim P_Z}\left(n\log\left\|Jv\right\| + D_\theta\left(G_\psi\left(z\right)\right)\right)\right\}. \tag{12}$$

The objective for the generator includes a maximization of the $\log$ determinant of the generator Jacobian. This ensures that the Jacobian stays full rank during training. Furthermore, due to optimization dynamics, if it so happens that the Jacobian ceases to be full rank or approaches singularity, the cost function approaches negative infinity making the training dynamics shift and "focus" on restoring Jacobian rank. In practice, we do not see the Jacobian approach singularity, and it stays well-behaved.

4. Experiments

In this section we provide experimental results for the generated data and the density estimation using both synthetic (Sec. 4.1) and real (Sec. 4.2) datasets. We show qualitative examples in Sec. 4.3. Implementation details and architectures can be found in Appendix C.2 and Appendix C.3.

4.1. Synthetic data

Table 1. The high quality (HQ) samples percentages and modes of various generators for 2D GMMs. The numbers for all models, except our model, are taken from [40].

	2D Ring		2D Grid	
	Modes (Max 8)	% HQ	Modes (Max 25)	% HQ
GAN [12]	1	**99.3**	3.3	0.5
ALI [11]	2.8	0.13	15.8	1.6
UGAN [29]	7.6	35.6	23.6	16
VEEGAN [40]	**8**	52.9	24.6	40
Ours	**8**	71.2	**25**	**52.5**

We begin by comparing the density estimation and sampling capabilities of our model. To this end, we perform an experiment on a synthetic 2D dataset, the same as the one presented in VEEGAN [40]. We train our model on two sets of Gaussian Mixture Models (GMM), with one set comprising 8 modes forming a ring (Fig. 1a) and another set comprising 25 modes in a grid (Fig. 1c). In order to test the theoretical analysis (Sec. 3) without the consequences of approximating the determinant of the Jacobian (Eq. (9)), we performed the 2D experiments while computing the exact determinant of the 2×2 Jacobians.

To quantify the quality of the density captured by the **generator** we use the "high-quality samples and modes" metric from [40], where a generated point is considered *high quality* if it is within a 3σ distance from the nearest mode, and a mode is counted if it is the nearest mode to at least one high-quality sample. We generate 2,500 points and report the percentage of points that are high quality and the number of modes over five runs. We can see in Tab. 1 that our generator is able to capture all the modes, while also producing higher quality samples than other models.

Since GAN models do not return a direct estimate of the probability of the data we cannot compare density estimation. Therefore, instead of a quantitative comparison, we qualitatively evaluate the density estimation of our discriminator by plotting in Fig. 1 its density map next to the ground truth density. Fig. 1 shows that our discriminator captures all the modes by giving them high values.

To show the effectiveness of the samples created by the generator, in Fig. 2(a) we show the $\log \zeta$ approximated by a different number of samples using three different bias distributions: 1) a standard normal distribution, 2) our generator distribution and 3) the ground-truth distribution. For each distribution and each number of samples, we run the computation 10 times, and use an error bar to represent the standard deviation of the results. Fig. 2(a) shows that using our generator is more accurate than using a normal distribution, and requires fewer samples to converge.

4.2. Real data

To evaluate our loss objectives (Eqs. (3) and (5)) on real datasets, we use the DCGAN [30] architecture and train the model with various numbers of samples to approximate the integral (Eq. (2)). Appendix C.2 details our training parameters, Appendix C.3.2 explains our DCGAN-based architecture and Appendix C.4 compares the runtime difference between WGAN and our method. As seen in Tab. 2, using our formulation we achieve better Fréchet Inception Distance (FID) values. The table also shows results from the traditional normalizing flow-based GLOW [21].

Following [10], we provide histograms of unnormalized log-likelihoods for train and test data in Fig. 3. We remark that there is a large overlap between the test and train distribution. This indicates that the discriminator generalizes to the test set and gives evidence against over-fitting.

Fig. 2(b-c) shows the value of ζ according to different numbers of samples. Here we see that the computed values converge under a practical number of samples. Note that since ζ is a constant of D_θ, its computation is required only once and saved for further density estimations.

4.3. Qualitative results

In Figs. 4 and 5 we visually assess the probabilities assigned by the discriminator by providing random samples sorted by density. We can see here that high-quality images score higher probabilities than low-quality images, suggesting that our trained discriminator has captured the distribution of the given dataset.

In Fig. 7 we show qualitative results for the generator on

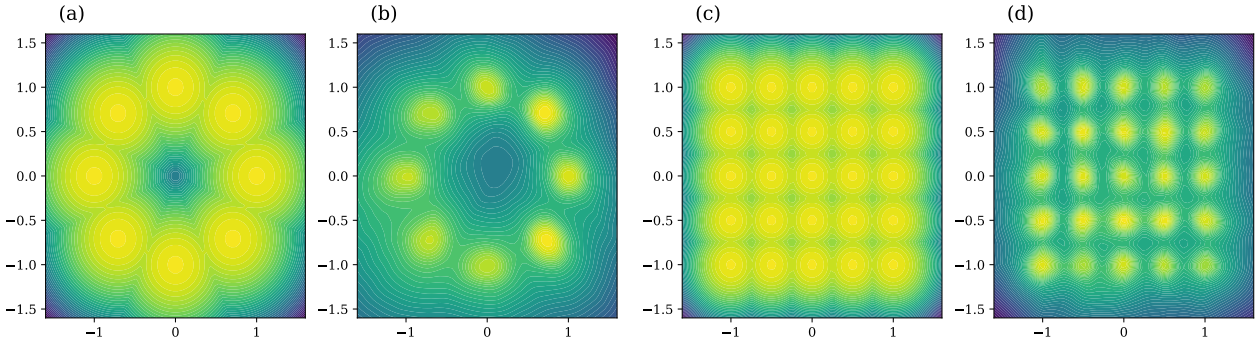

Figure 1. True distribution v.s. discriminator distribution in log space. Brighter colors represent higher values. (a) Ground truth distribution of the 8-modes ring. (b) Discriminator density estimation of the 8-modes ring. (c) Ground truth distribution of the 25-modes grid. (d) Discriminator density estimation of the 25-modes grid.

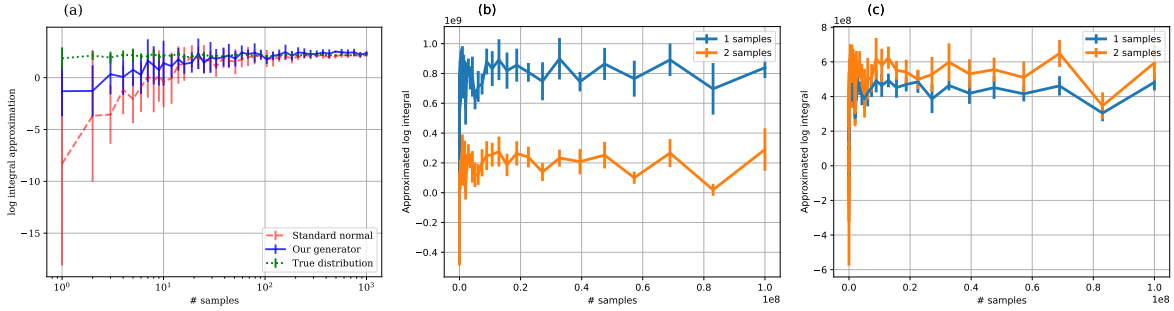

Figure 2. Approximated integral by number of samples used (as in Eq. (2)) for the grid distribution. The vertical error bars represent the standard deviation over 10 computations. (a) Synthetic 2D grid. (b) CelebA. (c) CIFAR-10.

an interpolation experiment, which shows that our framework retains smoothness in the latent space.

Finally, because our generator introduces noise when increasing dimensionality (Eq. (7)), we wanted to see what characteristics are controlled by the initial latent space. For that we used the same latent vector as an input to the generator multiple times and show the results in Fig. 8. We observe that the structure stays the same among the images, with slight variations, *e.g.* hair color. From here we hypothesize that changing the variance of groups of latent variables can be used as a mechanism to capture qualitative modes of the data. This corresponds to similar observations in StyleGAN [17] where the authors found interesting roles of intermediate auxiliary noise as they were introduced at different resolutions. However, in contrast here different noise variance plays a crucial role and highlights qualitatively high level of disentanglement in our generator.

5. Discussion

As described in Sec. 3.2, considering a likelihood-based approach for GAN training leads to maximization of the generator entropy in addition to the WGAN objective. Moreover, the new discriminator objective formulation, as described in Sec. 3.1, assists in removing the bias from the WGAN objective. Both of these differences from the WGAN objectives are made possible by having the generator provide the density of the generated samples. Whereas normalizing flows require computing the density of arbitrary points in order to train with log-likelihood maximization, a crucial difference with our model is that the computation of the density of real data points is not required.

Table 2. Comparison of FID values between our model and (1) WGAN with DCGAN architecture, (2) GLOW

	CELEBA FID	CIFAR-10 FID
GLOW	24	95
WGAN-GP	24	61
OURS - 1 SAMPLE	**22.5**	**42.4**
OURS - 2 SAMPLES	22.9	51.2

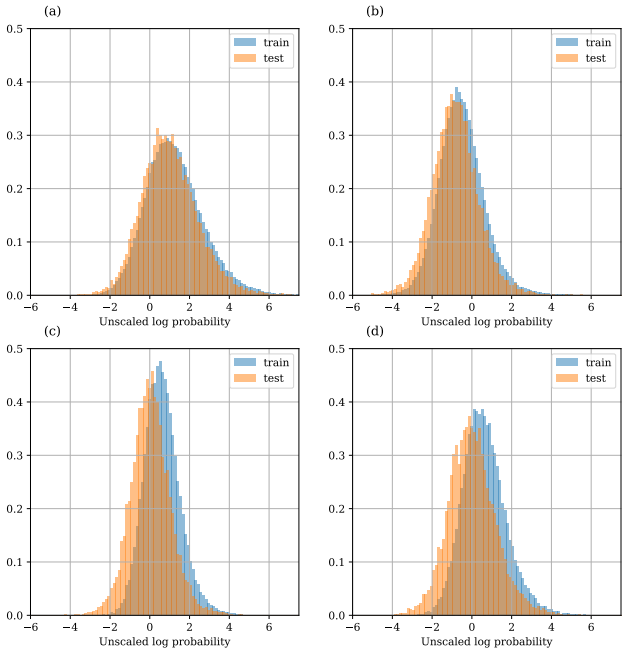

Figure 3. Overfit test. Histograms of the values returned by the discriminator for the train and test sets. Top row (a-b) for CelebA and bottom row (c-d) for CIFAR-10. The left column (a,c) uses a 1-sample approximation and ζ and the right column uses a 2-samples approximations.

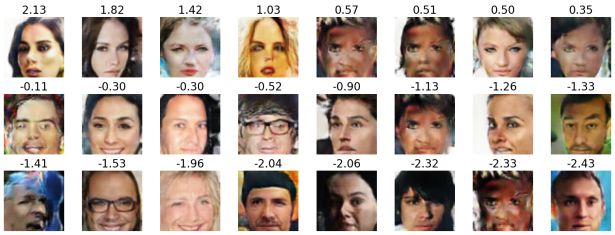

Figure 4. Random samples of generated CelebA images sorted by their discriminator-assigned unnormalized log probability. The value above each image is the discriminator score.

Only the density of the generated data points needs to be computed. This leads to a considerable relaxation in the generator architecture, where the model is allowed to increase dimensionality throughout the generator. The ability to increase dimensions appears to contribute to getting better quality images, as seen in Tab. 2, where GLOW, which has constant dimensionality, generates lower quality images than DCGAN or more modern GANs. To keep this operation tractable we adopted the approximate Jacobian determinant computation in Sec. 3.4. This arguably introduces noise in the gradient. We leave to future work the task

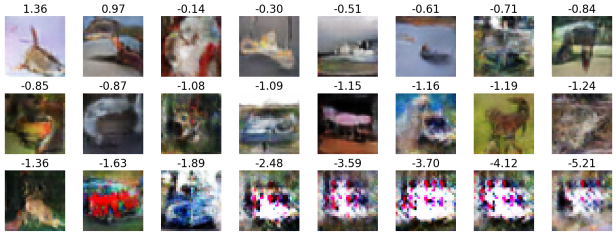

Figure 5. Random samples of generated CIFAR-10 images sorted by their discriminator-assigned unnormalized log-probability. The value above each image is the discriminator score.

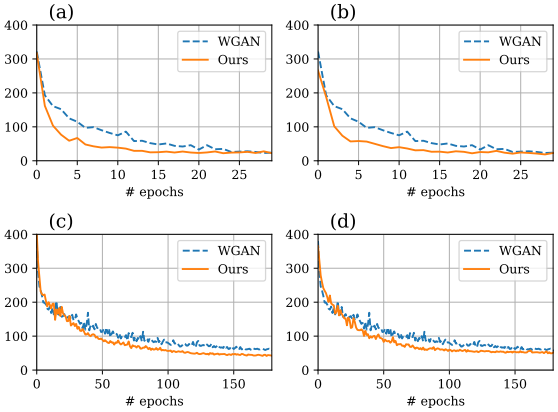

Figure 6. Evolution of FID during training using WGAN loss and our loss. Top row (a-b) for CelebA and bottom row (c-d) for CIFAR-10. The left column (a,c) uses a 1-sample approximation of ζ and the right column uses a 2-sample approximation.

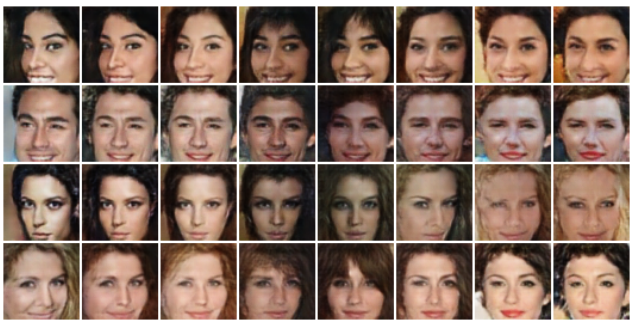

Figure 7. Generated images from linear interpolations of the latent space using the CelebA dataset. Each row is independent of the other.

of building a generator architecture with layers that have a closed-form Jacobian. For instance, the computation of the Jacobian determinant for a convolution operation can be obtained from [16, 36], and the Jacobian for element-wise layers is a diagonal matrix. We expect this will further speed up and stabilize GAN training. Furthermore, to track the probability of the output of a dimension-reducing layer, as in the

Figure 8. Generator outputs given the same latent vector.

final convolution layer of DC-GAN, the removed dimensions have to be marginalized, which is a difficult and expensive computation. When tractable down-sampling operation is discovered, it could be applied in our model as well. While experimenting with different number of samples for ζ, we observed that increasing the number of samples did not necessarily improve the image quality. We suspect that this is because, given more samples, the training focuses more on increasing the variance of the generated images. We also suspect that the architecture we used for testing did not have the capacity to accommodate these variances. We leave it to future work to unlock the full potential of using multiple samples for approximating the normalizing factor.

Finally, we leave to future work the application of the proposed work to more modern GAN generator and discriminator architectures (*e.g.* [18]).

6. Conclusion

We presented a framework for density estimation within GANs, and explored the connection between EBMs and GANs to develop an unbiased estimator of the partition function of an EBM. This led to an objective function that is closely related to Wasserstein GAN with an additional entropy maximization criterion for the generator training that enables greater diversity of the generated samples. Furthermore, we proposed a modified flow network as generator, called one-way flow, which provides both samples and density estimates to compute empirical expectations while maintaining architectural flexibility. This allows for an efficient way of evaluating the generator density and generator entropy, which has historically proven hard. Our experimental results show that our model produces samples that are on par with other GAN generators, along with accurate density estimations and faster convergence. Our model provides new understandings of the properties of the discriminator and insights into GANs from a maximum likelihood perspective, while connecting these to EBMs. To accommodate maximum flexibility we have used a stochastic Jacobian-determinant approximator; we leave as future work its exact computation, which we hypothesize can reduce variance and speed up training.

Acknowledgments

The authors thank the International Max Planck Research School for Intelligent Systems for supporting OB. MJB has received research gift funds from Adobe, Intel, Nvidia, Meta/Facebook, and Amazon. MJB has financial interests in Amazon, Datagen Technologies, and Meshcapade GmbH. While MJB is a consultant for Meshcapade, his research in this project was performed solely at, and funded solely by, the Max Planck Society.

References

[1] M Ehsan Abbasnejad, Qinfeng Shi, Anton van den Hengel, and Lingqiao Liu. A generative adversarial density estimator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10782–10791, 2019. 2

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *International inproceedings on Machine Learning (ICML)*, 2017. 1, 2, 3

[3] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016. 2

[4] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is secretly an energy-based model and you should use discriminator driven latent sampling. *Advances in Neural Information Processing Systems*, 33:12275–12287, 2020. 2

[5] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. In *International inproceedings on Learning Representations (ICLR)*, 2017. 1, 2, 3, 4

[6] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[7] Peter J Diggle and Richard J Gratton. Monte Carlo Methods of Inference for Implicit Statistical Models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1984. 2

[8] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear Independent Components Estimation. *arXiv preprint arXiv:1410.8516*, 2014. 1

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density Estimation Using Real NVP. In *International inproceedings on Learning Representations (ICLR)*, 2017. 1, 2, 4

[10] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019. 5

[11] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially learned inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 5

[12] 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 (NeurIPS)*, 2014. 1, 2, 5

[13] 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. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[15] Peter Huber. The behavior of maximum likelihood estimates under nonstandard conditions. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1967. 1

[16] Mahdi Karami, Dale Schuurmans, Jascha Sohl-Dickstein, Laurent Dinh, and Daniel Duckworth. Invertible convolutional flow. *Advances in Neural Information Processing Systems*, 32, 2019. 7

[17] Tero Karras, Samuli Laine, and Timo Aila. A style-based Generator Architecture for Generative Adversarial Networks. In *Proceedings of the IEEE inproceedings on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 6

[18] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and Improving the Image Quality of StyleGAN. In *Proceedings of the IEEE inproceedings on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 4, 8

[19] Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016. 1, 2, 4

[20] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 13

[21] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2, 5

[22] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International inproceedings on Learning Representations (ICLR)*, 2014. 1

[23] Tuen Kloek and Herman K Van Dijk. Bayesian estimates of equation system parameters: an application of integration by monte carlo. *Econometrica: Journal of the Econometric Society*, pages 1–19, 1978. 3

[24] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2020. 2

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images.(2009), 2009. 13

[26] Solomon Kullback and Richard A Leibler. On Information and Sufficiency. *The annals of mathematical statistics*, 1951. 3

[27] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, 2015. 13

[28] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *International inproceedings on Machine Learning (ICML)*, 2018. 2

[29] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 5

[30] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434*, 2015. 5, 14

[31] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative Adversarial Text to Image Synthesis. In *International inproceedings on Machine Learning (ICML)*, 2016. 4

[32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 1, 2

[33] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo Method*. John Wiley & Sons, 2016. 3

[34] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016. 2

[35] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and other Modifications. In *International inproceedings on Learning Representations (ICLR)*, 2017. 2

[36] Hanie Sedghi, Vineet Gupta, and Philip M. Long. The Singular Values of Convolutional Layers. In *International inproceedings on Learning Representations (ICLR)*, 2019. 7

[37] Jascha Sohl-Dickstein. Two equalities expressing the determinant of a matrix in terms of expectations over matrix-vector products. *arXiv preprint arXiv:2005.06553*, 2020. 4

[38] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 1

[39] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021. 2, 3

[40] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. VEEGAN: Reducing Mode

Collapse in GANs Using Implicit Variational Learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2, 5

[41] Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003. 1, 2, 3

[42] Lucas Theis and Matthias Bethge. Generative Image Modeling Using Spatial LSTMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 2

[43] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional Image Generation with PixelCNN Decoders. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 1

[44] Aaron Van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel Recurrent Neural Networks. In *International inproceedings on Machine Learning (ICML)*, 2016. 1, 2