Flow-GAN: Bridging implicit and prescribed learning in generative models

Aditya Grover [1] **Manik Dhar** [1] **Stefano Ermon** [1]

Abstract

Evaluating the performance of generative models for unsupervised learning is inherently challenging due to the lack of well-defined and tractable objectives. This is particularly difficult for implicit models such as generative adversarial networks (GANs) which perform extremely well in practice for tasks such as sample generation, but sidestep the explicit characterization of a density. We propose Flow-GANs, a generative adversarial network with the generator specified as a normalizing flow model which can perform *exact* likelihood evaluation. Subsequently, we learn a Flow-GAN using a hybrid objective that integrates adversarial training with maximum likelihood estimation. We show empirically the benefits of Flow-GANs on MNIST and CIFAR-10 datasets in learning generative models that can attain low generalization error based on the log-likelihoods and generate high quality samples. Finally, we show a simple, yet hard to beat baseline for GANs based on Gaussian Mixture Models.

1. Introduction

Highly expressive parametric models have enjoyed great success in supervised learning, where learning objectives and evaluation metrics are typically well-specified and easy to compute. On the other hand, the learning objective for unsupervised settings is less clear. At a fundamental level, the idea is to learn a generative model that minimizes some notion of divergence with respect to the data distribution. Minimizing the classic Kullback-Liebler (KL) divergence between the data distribution and the model, for instance, is equivalent to performing maximum likelihood estimation (MLE) on the observed data. Maximum likelihood estimators are asymptotically statistically efficient, and serve as natural objectives for learning prescribed generative models (Mohamed & Lakshminarayanan, 2016).

In contrast, an alternate principle that has recently attracted much attention is based on adversarial training, where the objective is to generate data indistinguishable from the training data. Adversarially trained generative models can sidestep specifying an explicit density for the observed data and are referred to as implicit models (Diggle & Gratton, 1984). In practice, adversarially trained implicit models such as generative adversarial networks (GAN; (Goodfellow et al., 2014)) generate high-quality samples outperforming samples obtained from models learned using MLE. While the evaluation of implicit models remains a challenge, recent work in approximate inference methods suggests that GAN models attain much poorer log-likelihoods compared to models trained with MLE (Wu et al., 2017). The dichotomy in the current state-of-the-art models is unfortunate, but not very surprising; as (Theis et al., 2016) demonstrates using simple examples in the domain of natural images, maximizing log-likelihoods and generating perceptually good-looking samples are largely uncorrelated.

In this work, we propose Flow-GANs, a prescribed generative model that can be efficiently trained to achieve the dual objectives of good sample quality as well as high log-likelihood. At the heart of our framework we have a normalizing flow model such as NICE (Dinh et al., 2014) or Real-NVP (Dinh et al., 2017) which can transform a simple tractable noise

[1]Stanford University. Correspondence to: Aditya Grover <adityag@cs.stanford.edu>.

Presented at the ICML 2017 Workshop on Principled Approaches to Deep Learning, Sydney, Australia, 2017.

density (such as isotropic Gaussian) into a complex density through a sequence of invertible transformations similar to the generator of a GAN. Crucially, flow models allow us to tractably evaluate the *exact* log-likelihoods as well as perform *exact* posterior inference over the latent variables while still permitting efficient ancestral sampling, desirable properties of any probabilistic model that a typical GAN would not provide.

We use Flow-GANs to propose a new objective for learning and evaluating generative models that bridges implicit and prescribed learning by augmenting the adversarial training objective with an additional term corresponding to the log-likelihood of the data. Alongside, we empirically evaluate some beliefs and conjectures regarding the behavior of models learned using MLE and adversarial training. An exact evaluation of log-likelihoods also permits us to evaluate non-parameteric methods such as Kernel Density Estimation (KDE; (Parzen, 1962)) typically used for density estimation in GANs, and we observe that these methods are ineffective for density estimation underestimating the log-likelihoods by a large margin. We observe that GANs produce poor log-likelihoods and great samples *without* memorizing the data. Perhaps surprisingly, we conclude by comparing GANs with a simple yet hard to beat baseline based on Gaussian Mixture Models that outperforms GANs on metrics corresponding to both log-likelihood evaluation and generating high-quality samples for complex real-world datasets.

2. Preliminaries

We begin with a review of generative modeling in the context of generative adversarial networks and normalizing flow models. For ease of presentation, all distributions are w.r.t. any arbitrary $\mathbf{x} \in \mathbb{R}^d$, unless otherwise specified. We use upper-case symbols to denote probability distributions and assume they all admit absolutely continuous densities (denoted by the corresponding lower-case notation) on a reference measure $\mathrm{d}\mathbf{x}$.

Consider the following setting for parametric learning in generative models. Given some data $X = \{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^m$ sampled i.i.d. from an unknown probability density p_{data}, we are interested in learning a parametric density p_θ where θ denotes the parameters of the model.

2.1. Generative Adversarial Networks

Generative Adversarial Networks (Goodfellow et al., 2014) are a class of probabilistic models consisting of a pair of generator and discriminator. The generator, denoted by a deterministic function $G_\theta : \mathbb{R}^k \to \mathbb{R}^d$ takes as input a source of randomness $\mathbf{z} \in \mathbb{R}^k$ sampled from a tractable prior distribution $p(\mathbf{z})$ and transforms it to a sample $G_\theta(\mathbf{z})$. Samples generated by the model along with the true data X are then passed on to the discriminator, denoted by another function $D_\phi : \mathbb{R}^d \to \mathbb{R}$ parameterized by ϕ.

The discriminator acts as a binary classifier that tries to distinguish the true data from the model generated samples by maximizing a suitable objective (such as the negative cross entropy). The generator minimizes the same objective, such that the overall objective assumes a minimax formulation,

$$\min_\theta \max_\phi \mathbb{E}_{x\sim P_{\mathrm{data}}} \left[\log D_\phi(x)\right] + \mathbb{E}_{z\sim P_z}\left[\log\left(1 - D_\phi(G_\theta(z))\right)\right]. \quad (1)$$

where we have assumed that the discriminator is maximizing the negative cross-entropy as originally proposed in (Goodfellow et al., 2014). The generators and discriminators are typically deep neural networks.

Recent work has generalized the objective in Eq. (1) to arbitrary f-divergences (Nowozin et al., 2016) and integral probability metrics (Zhao et al., 2017; Sutherland et al., 2017). In particular, Wasserstein GAN (WGAN, (Arjovsky et al., 2017)), where the minimax is over the Wasserstein distance between the model and data distributions, has been shown to exhibit good theoretical properties and empirical performance in stabilizing GAN training,

$$\min_\theta \max_{\phi\in\mathcal{F}} \mathbb{E}_{x\sim P_{\mathrm{data}}}\left[D_\phi(x)\right] - \mathbb{E}_{z\sim P_z}\left[D_\phi(G_\theta(z))\right]. \quad (2)$$

where $\mathcal{F}$ is the class of $1-$Lipschitz functions. In practice, the Lipschitz constraint is imposed by clipping weights for the discriminator (referred to as critic

in the original work (Arjovsky et al., 2017)) or adding a gradient penalty term to the objective (Gulrajani et al., 2017).

2.2. Normalizing flow models

A normalizing flow specifies a parametric transformation from a prior density $p(\mathbf{z}) : \mathbb{R}^d \to \mathbb{R}_0^+$ to another density over the same space, $p_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}_0^+$. The transformation is specified via an invertible transformation, $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$. Using the change-of-variables formula and letting $\mathbf{z} = f_\theta(\mathbf{x})$ we have

$$p_\theta(\mathbf{x}) = p(\mathbf{z}) \left| \det \frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}} \right| \tag{3}$$

where $\frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}}$ denotes the Jacobian of f_θ at $\mathbf{x}$. The above formula can be applied recursively over compositions of many invertible transformations to produce a complex final density. Hence, we can evaluate and optimize for the log-likelihood assigned by the model to a data point as long as the prior density is tractable and the determinant of the Jacobian of f_θ evaluated at $\mathbf{x}$ can be efficiently computed.

The Jacobian of high-dimensional distributions can however be computationally expensive to compute. If the transformations are designed such that the Jacobian is an upper or lower triangular matrix, then the determinant can be easily evaluated as the product of its diagonal entries. Based on prior work in flow models, we consider two such family of transformations.

1. Volume preserving transformations in NICE (Dinh et al., 2014): The model consists of several layers of additive coupling layers that perform a location transformation with a unit Jacobian determinant. The top layer is a diagonal scaling matrix.

2. Non-volume preserving transformations in Real-NVP (Dinh et al., 2017): Every coupling layer performs both a location and scale transformation, and hence, the Jacobian determinant is not necessarily unity.

We direct the reader to (Dinh et al., 2014; 2017) for the precise specifications of the transformations. The learning objective for both Real-NVP and NICE corresponds to MLE,

$$\max_\theta \mathbb{E}_{\mathbf{x} \sim P_{\text{data}}} [\log p_\theta(\mathbf{x})] \tag{4}$$

Since the transformations are invertible, ancestral sampling is possible in these models by sampling a random vector $\mathbf{z} \sim P_z$ and transforming it to a model generated sample via $g_\theta = f_\theta^{-1}$.

3. Flow generative adversarial networks

Evaluating the log-likelihood assigned by a generative adversarial network is challenging because the model density p_θ is specified only implicitly using the prior density p_Z and the deterministic generator function G_θ.

One way to make the model density explicit is to view the GAN as a latent variable model and parameterize the observation model, $p_\theta(\mathbf{x}|\mathbf{z})$ as some noise distribution such as a Gaussian (Wu et al., 2017). This assumption alone is not sufficient since the marginal likelihood of the observed data, $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) \mathrm{d}\mathbf{z}$ in this case would be intractable as it requires integrating over all the latent factors of variation. This would then require approximate inference (e.g., Monte Carlo or variational methods) which in itself is a challenge for high-dimensional distributions.

Alternatively if the generator function G_θ is designed to be reversible such that G_θ^{-1} exists and the prior density p_Z can be evaluated tractably, we can compute the log-likelihood assigned by a model directly using the change-of-variables formula. Substituting for $f = G_\theta^{-1}$ to Eq. (3), we have the log-likelihood assigned by a GAN model to a data point as,

$$\log p_\theta(\mathbf{x}) = \log p(G_\theta^{-1}(\mathbf{x})) + \log \left| \det \frac{\partial G_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right| \tag{5}$$

where $\frac{\partial G_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}}$ denotes the Jacobian of G_θ^{-1} at $\mathbf{x}$. Denoting $\mathbf{z} = G_\theta^{-1}(\mathbf{x})$, we see that the first term corresponds to the prior density in a GAN which is typically chosen to correspond to a simple analytical distribution such as an isotropic Gaussian.

Requiring the generator function G_θ to be reversible imposes a constraint on the dimensionality of the la-

tent variable $\mathbf{z}$ to match that of the data $\mathbf{x}$. Thereafter, we require the transformations between the various layers of the generator to be invertible such that their overall composition results in an invertible G_θ. For instance, these transformations could correspond to the ones used in volume preserving or non-volume preserving models as discussed in the previous section.

Crucially, we can exactly evaluate log-likelihoods of any GAN model by specifying the generator as a normalizing flow model. We refer to such models as flow generative adversarial networks (Flow-GANs). A Flow-GAN can be trained using MLE, adversarial training or a hybrid objective based on both these inductive principles. Without loss of generality, let $V(G_\theta, D_\phi)$ denote the minimax objective of a GAN. The hybrid objective of a flow-GAN can be expressed as,

$$\min_\theta \max_\phi V(G_\theta, D_\phi) - \lambda \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}} [\log p_\theta(\mathbf{x})] \quad (6)$$

where $\lambda \geq 0$ is a hyperparameter for the algorithm. By varying λ, we can interpolate between plain adversarial training ($\lambda = 0$) and MLE (very high λ). Consequently, the generator is inclined to generate samples indistinguishable from the real data to fool the discriminator as well as assign high log-likelihoods to the observed data. Overfitting to the observed data can be detected by cross-validation, and performing early stopping of training when the log-likelihoods on the training and validation sets begin to diverge.

4. Experiments

We conduct three different kinds of experiments. In the first set of experiments, we evaluate the merits of training Flow-GANs (Eq. (6)) over plain maximum likelihood estimation and adversarial training in terms of log-likelihoods and recently proposed sample quality metrics. The second set of experiments compares non-parameteric methods for density estimation (e.g., KDE) with the exact log-likelihoods computed by a flow model. Finally, we show a simple baseline based on Gaussian Mixture Models that outperforms adversarial training in terms of both log-likelihoods and sample quality metrics.

Datasets. We experiment with the MNIST dataset of handwritten digits (LeCun et al., 2010) and the CIFAR-10 dataset of natural images (Krizhevsky & Hinton, 2009).The MNIST dataset contains $50,000$ train, $10,000$ validation, and $10,000$ test images of dimensions 28×28. The CIFAR-10 dataset (Krizhevsky & Hinton, 2009) contains $50,000$ train and $10,000$ test images of dimensions $32 \times 32 \times 3$ by default. We randomly hold out $5,000$ training set images as validation set. Since we are modeling densities for discrete datasets (pixels can take a finite set of values ranging from 1 to 255), the model can assign arbitrarily high log-likelihoods to these discrete points. Following (Uria et al., 2013), we avoid this scenario by dequantizing the data where uniform noise between 0 and 1 is added to every pixel. Finally, we scale the pixels to lie between 0 and 1.

Model architecture. We used the NICE (Dinh et al., 2014) and Real-NVP (Dinh et al., 2017) architectures as the normalizing flow generators for the MNIST and CIFAR-10 datasets respectively. For adversarial training, we consider the improved WGAN objective (Eq. (2)) which enforces the Lipschitz constraint over the critic by penalizing the norm of the gradient with respect to the input (Gulrajani et al., 2017). The critic architecture for the MNIST dataset is the same as the one used in the DCGAN (Radford et al., 2015). The batch normalization layer is removed as recommended (Gulrajani et al., 2017). For the CIFAR-10 experiments we again use the DCGAN architecture with an extra convolutional layer. Batch normalization (Ioffe & Szegedy, 2015) here was substituted by layer normalization (Ba et al., 2016).

Hyperparameters. The hyperaparmeters specific to adversarial training with the Wasserstein distance were fixed as per (Gulrajani et al., 2017). For all models trained on MNIST, we used a logistic prior distribution for the NICE architecture. The weights were initialized using the truncated normal distribution. For the models trained using MLE, batch size was set to 200, learning rate was 0.001 with a decay of 0.9995 and the optimizer used was Adam with $\beta_1 = 0.9, \beta_2 = 0.999$. For the other models, the batch size was set to 100, and the optimizer used was Adam with $\beta_1 = 0.5, \beta_2 = 0.999$ and a learning rate was 0.0001. Finally, we set $\lambda = 0.1$ in Eq. (6) for the

Flow-GAN model.

In the case of models trained on CIFAR-10, an isotropic Gaussian prior was used. We also used weight normalization and data dependent initialization (Salimans & Kingma, 2016) for all the models. The models trained with MLE used a batch size of 100, a learning rate of 0.0001 with a decay of 0.999995 and the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$. The other models used a batch size of 64, a learning rate of 0.0001 and the Adam optimizer with $\beta_1 = 0.5, \beta_2 = 0.999$. We set $\lambda = 1$ in Eq. (6) for the Flow-GAN model.

4.1. Log-likelihood vs. sample quality

Experimental setup. Using a normalizing-flow generator, we compare the performance of three learning objectives. The first objective corresponds to the maximum likelihood estimation (MLE, Eq. (4)), the second corresponds to adversarial training (WGAN, Eq. (2)), and the final one is the hybrid objective (Flow-GAN, Eq. (6)) we propose.

For evaluating sample quality, we use two metrics designed for labeled datasets that are consistent with standard practice. For the MNIST dataset, we report the recently proposed MODE score (Che et al., 2017) calculated as,

$$\exp\left(\mathbb{E}_{\mathbf{x}\in P_\theta}[KL(p(y|\mathbf{x})\|p^*(y)] - KL(p^*(y)\|p(y))\right)$$

where $\mathbf{x}$ is a sample generated by the model, $p(y|\mathbf{x})$ is the softmax probability assigned by a pretrained classifier, $p^*(y)$ is the true distribution of labels and $p(y)$ is the distribution of labels in the generated samples (as predicted by the pretrained classifier). For the CIFAR-10 dataset, the common practice is to report the Inception scores calculated as,

$$\exp\left(\mathbb{E}_{\mathbf{x}\in P_\theta}[KL(p(y|\mathbf{x})\|p(y)]\right).$$

For each of the three objectives, we keep track of the learning curves corresponding to the negative log-likelihood on the training and validation sets and the MODE/Inception scores. Additionally, for the WGAN and Flow-GAN models, we also track the Wasserstein loss.

Results. Due to lack of space and in order to keep the presentation clean, all the samples and learning curves are deferred to the supplementary. Please refer to the supplementary to aid the following discussion. The learning curves for the three models trained on the MNIST and CIFAR-10 datasets are reported in Figures 2, 3, 4, and Figures 5, 6, 7 respectively. The best MODE/Inception scores and the test negative log-likelihoods (NLL) corresponding to the best cross-validated models are listed in Table 1 and Table 2. Following standard practice, we report the NLLs for MNIST in nats and the NLLs for CIFAR-10 in bits/dimension. For each learning objective, we also show the actual samples at various points of training with the samples attaining the best performance on sample quality metrics in a green box at the bottom-right.

The learning curves shed light on key observations regarding the dynamics of various objectives. For plain MLE objective (Figure 2 and Figure 5), we see that the models attains low validation NLLs (blue) in very few iterations of generator training. Overfitting is not a big concern as the train (brown) and validation curves almost overlap each other. Additionally, the sample quality metrics (red) show a gradual increase as training progresses.

However, the models trained using the WGAN objective (Figure 3 and Figure 6) attain much higher scores on the sample quality metrics (red, see Table 1 and Table 2 for maximum MODE/Inception scores). We additionally trace the WGAN loss (green) which we observe to be strongly correlated with the sample quality metrics (the WGAN loss decreases as the sample quality metrics increase). Surprisingly, overfitting is again not an issue with the WGAN models since the train (brown) and validation (blue) negative log-likelihoods overlap with each other suggesting that *WGANs are not simply memorizing the training data*. However, the negative log-likelihood curves show a largely consistent *increase* as training progresses (for the CIFAR-10 dataset, these curves are shown on a log-scale!) confirming the widely held-viewpoint that good sample quality scores do not imply good log-likelihoods. Dissatisfyingly, these results suggest that the log-likelihoods get worse as learning progresses for models learned using plain adversarial training.

Finally, the Flow-GAN models (Figure 4 and Figure 7) trained using the hybrid objective attain both

Table 1. Best MODE scores and test negative log-likelihood estimates (NLL, in nats) for the best cross-validated NICE models evaluated using the change-of-variables formula (Exact) and kernel density estimation (KDE) respectively for the MNIST dataset.

Objective	MODE Score	Exact NLL	KDE
MLE	7.42	-3334.56	-159.89 ± 1.31
WGAN	9.24	-1604.09	486.27 ± 0.30
Flow-GAN	9.37	-3342.95	-63.51 ± 0.66

Table 2. Best Inception scores and test negative log-likelihood estimates (NLL, in bits/dim) for the best cross-validated Real-NVP models evaluated using the change-of-variables formula (Exact) and kernel density estimation (KDE) respectively for the CIFAR-10 dataset.

Objective	Inception Score	Exact NLL	KDE
MLE	2.92	3.54	7.82 ± 0.002
WGAN	5.76	8.53	8.28 ± 0.002
Flow-GAN	3.90	4.21	7.79 ± 0.002

good log-likelihoods (blue) and sample quality scores (red) as desired (see Table 1 and Table 2 for actual numbers). For the MNIST dataset, Flow-GAN attains a test log-likelihood of 3342.95 which is a few nats above the one obtained by MLE (3334.56). Similarly, samples generated by the Flow-GAN have a MODE score of 9.37 which surpasses the MODE score of 9.24 attained by the WGAN model. Although subjective, the perceptual quality of the samples with the best MODE scores shown at the bottom-right of Figures 4 and Figures 3 seems to suggest the same. For the CIFAR-10 dataset, the Flow-GAN model smoothly interpolates between the extremes attaining reasonable log-likelihoods unlike the WGAN model and better sample quality metrics than the Real-NVP model.

4.2. Non-parameteric density estimation

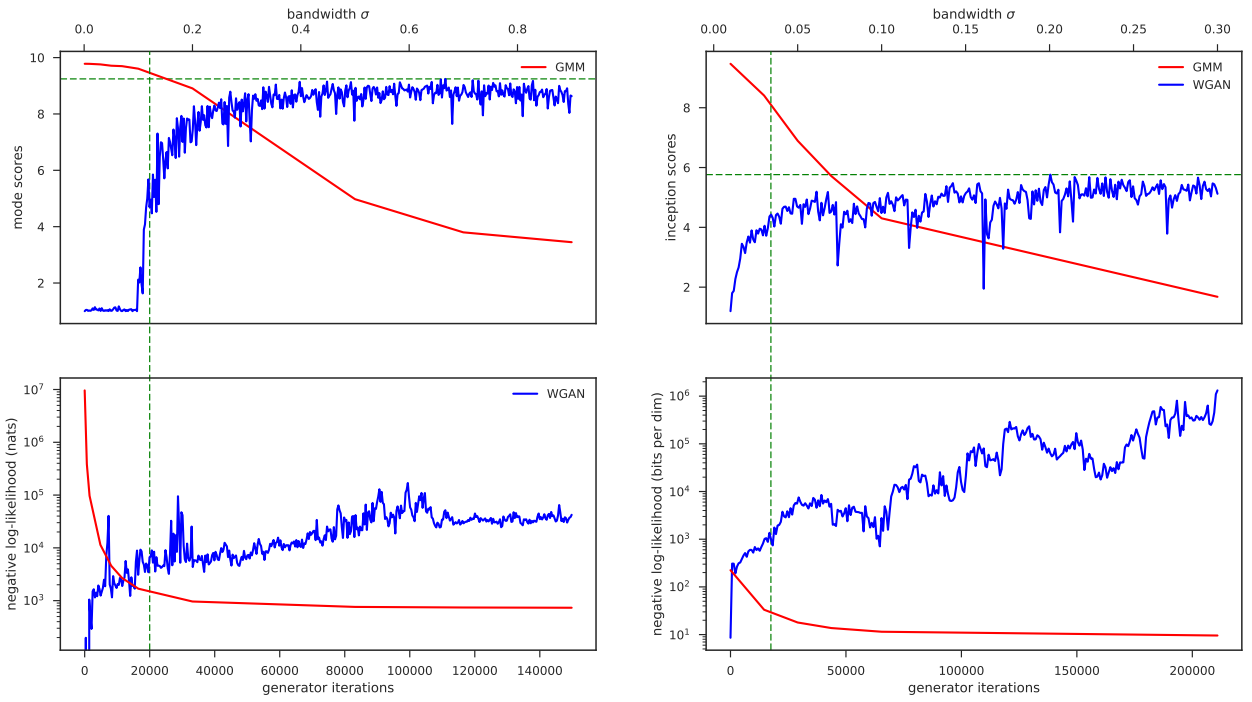
Experimental setup. We draw 10,000 samples from the MNIST dataset and 5,000 samples from the CIFAR-10 dataset. For an implicit model that only allows forward sampling, we can compute its density using non-parameteric methods such as Kernel Density Estimation (KDE) by fitting Parzen windows to samples generated from the model (Bengio et al., 2013; Goodfellow et al., 2014). In this experiment, we compute the KDE for the best performing models (as measured by validation log-likelihoods) on the test set for the different learning objectives and compare it with the *exact* log-likelihoods computed by the flow model itself.

Results. As we see in both Table 1 and Table 2, KDE underestimates the true log-likelihood of the model almost each time. In fact, as we see in Table 2, it is often not even able to find the correct ranking of the models, consistent with similar observations made previously (Theis et al., 2016).

4.3. Generative Adversarial Networks vs. Gaussian Mixture Models

Experimental setup. We use a mixture of m isotropic Gaussians with equal weights centered at each of the m training points as the baseline Gaussian Mixture Model (GMM). The bandwidth parameter, σ, is the same for each of the mixture components and optimized by doing a line search over the validation log-likelihood in (0, 1]. We compare this model with the WGAN-only model (same as the previous experiment) on the basis of test log-likelihood and sample quality metrics as before. We overload the x-axis of Figure 1 to denote the bandwith values for the GMM and the number of iterations of generator training for the WGAN as before.

Results. When the bandwidth, σ, is low, the GMM (red) produces excellent samples (very high MODE/Inception scores) but high validation NLLs because of overfitting. On increasing σ, the GMM starts to assign log-likelihoods based on non-trivial contributions from several mixture components increasing its generalization ability to points outside the training set. However, the sample quality metrics reduces in this case representing a clear trade-off. For



(a) MNIST

(b) CIFAR-10

Figure 1. WGAN vs. Gaussian Mixture Models (GMM). Sample quality metrics and validation negative log likelihoods (NLL) for the MNIST and CIFAR-10 respectively. For appropriate values of the bandwidth parameter, the GMM baseline (vertical green dashed lines) can achieve better NLL estimates (even on the test set, not shown) and sample quality scores.

the WGAN model (blue), on the other hand, the sample quality increases as training progresses but the negative log-likelihoods also increase sharply again representing a trade-off.

The horizontal green dashed line in the top plots of Figure 1 denotes the best attainable MODE/Inception scores by the WGAN model. The GMM clearly can attain better sample quality metrics. Surprisingly, the simple GMM also outperforms the WGAN model for several values of the bandwidth parameter. An example bandwidth parameter for a single GMM model that outperforms the highest MODE score attained by the WGAN model as well as the lowest negative likelihoods attained by the WGAN model at any point during learning can be seen by noting the bandwidth corresponding to the vertical green dashed line.

5. Discussion and related work

A wide variety of perspectives have contributed to advancements in generative modeling. A useful characterization is distinguishing between prescribed and implicit density models (Goodfellow, 2016; Mohamed & Lakshminarayanan, 2016). The former are equipped with a natural ability to specify an explicit characterization of the density assigned by the model, and hence, are typically trained using the maximum likelihood principle. Note that if the prescribed model is directed, ancestral sampling is also efficient and hence, we can easily obtain samples from the model.

In the case of implicit models such as generative adversarial networks, we only have tractable access to samples from the model. Hence, we need to define a notion of divergence between the data distribution (only accessible through its samples) and the model distribution (again, only accessible through its samples). While this might seem to be a limitation at

first, the adversarial training objective has achieved remarkable success across many applications such as semi-supervised learning (Salimans et al., 2016) and image super-resolution (Ledig et al., 2016) amongst many others. The original GAN objective has been generalized to include f-divergences (Nowozin et al., 2016), and many integral probability metrics (Sutherland et al., 2017; Arjovsky et al., 2017).

One of the key takeaways of this work is the simple observation that even prescribed models can be trained adversarially as long as sampling from the model is efficient. This line of reasoning has been explored to some extent in prior work mostly focussed on combining the objectives of prescribed latent variable models such as VAEs (maximizing an evidence lower bound on the data) with adversarial learning typical in implicit models (Larsen et al., 2015; Mescheder et al., 2017). However, the benefits of adversarial learning in such latent variable models does not come for "free" since we still need to resort to some form of approximate inference to get a handle on the log-likelihoods. This could be expensive, for instance combining a VAE with a GAN introduces an additional inference network increasing the overall model complexity.

The approach adopted in this paper sidesteps the additional complexity due to approximate inference by considering a normalizing flow model. The trade-off made by a normalizing flow model in this case is that the generator function needs to be invertible while other generative such as VAEs have no such requirement. On the positive side, we can exactly evaluate log-likelihoods assigned by the model which can then be combined with the adversarial learning objective. Other methods for evaluating log-likelihoods such as KDE exist, but suffer from the curse of dimensionality (Theis et al., 2016). Recently, (Wu et al., 2017) proposed to use Annealed Importance Sampling (AIS) for evaluating log-likelihoods assigned by the generator. In order to do so, one needs to assume a noisy observation model (such as a Gaussian), and subsequently run potentially expensive Markov chains. Besides (Dinh et al., 2014; 2017), normalizing flow models have also been used to increase the expressiveness of posterior distributions for variational inference in latent variable models (Rezende & Mohamed, 2015).

6. Conclusion

As an attempt to more quantitatively evaluate generative models, we introduced Flow-GAN. It is a generative adversarial network which allows for tractable likelihood evaluation, exactly like in a flow model. Since it can be trained both adversarially (like a GAN) and in terms of MLE (like a flow model), we can quantitatively evaluate the trade-offs involved in a principled fashion. In particular, we also consider a hybrid objective function which involves both types of losses. On MNIST, our hybrid Flow-GAN objective outperforms both pure MLE and a pure adversarial training, both in terms of sample quality and log likelihood. On CIFAR-10, our Flow-GAN objective outperforms a pure MLE approach in terms of visual sample quality, and is only slightly worse in terms of log-likelihood. Flow-GAN dramatically outperforms a pure GAN objective in terms of log-likelihood, while still providing reasonable samples.

A second objective of this research was to evaluate the effectiveness of existing approximate likelihood evaluation schemes in a domain where ground truth is available. Our results confirm that KDE is a very poor approximation of the true log-likelihood (at least for the models we considered), and should therefore not be used to rank different generative models.

Finally, the availability of quantitative metrics allow us to compare to simple baselines which essentially "remember" the training data. Our final results show that naive Gaussian Mixture Models outperforms plain WGAN on both samples quality and log-likelihood for both MNIST and CIFAR-10 which we hope will lead to new directions for both implicit and prescribed learning in generative models.

References

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

Ba, Jimmy Lei, Kiros, Jamie Ryan, and Hinton, Geoffrey E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bengio, Yoshua, Mesnil, Grégoire, Dauphin, Yann, and Rifai, Salah. Better mixing via deep representations. In *ICML*, 2013.

Che, Tong, Li, Yanran, Jacob, Athul Paul, Bengio, Yoshua, and Li, Wenjie. Mode regularized generative adversarial networks. In *ICLR*, 2017.

Diggle, Peter J and Gratton, Richard J. Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 193–227, 1984.

Dinh, Laurent, Krueger, David, and Bengio, Yoshua. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, Laurent, Sohl-Dickstein, Jascha, and Bengio, Samy. Density estimation using real NVP. In *ICLR*, 2017.

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

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *NIPS*, 2014.

Gulrajani, Ishaan, Ahmed, Faruk, Arjovsky, Martin, Dumoulin, Vincent, and Courville, Aaron. Improved training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028*, 2017.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. 2009.

Larsen, Anders Boesen Lindbo, Sønderby, Søren Kaae, Larochelle, Hugo, and Winther, Ole. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

LeCun, Yann, Cortes, Corinna, and Burges, Christopher JC. MNIST handwritten digit database. *http://yann. lecun. com/exdb/mnist*, 2010.

Ledig, Christian, Theis, Lucas, Huszár, Ferenc, Caballero, Jose, Cunningham, Andrew, Acosta, Alejandro, Aitken, Andrew, Tejani, Alykhan, Totz, Johannes, Wang, Zehan, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

Mescheder, Lars, Nowozin, Sebastian, and Geiger, Andreas. Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. *arXiv preprint arXiv:1701.04722*, 2017.

Mohamed, Shakir and Lakshminarayanan, Balaji. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Nowozin, Sebastian, Cseke, Botond, and Tomioka, Ryota. f-GAN: Training generative neural samplers using variational divergence minimization. In *NIPS*, 2016.

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

Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Rezende, Danilo Jimenez and Mohamed, Shakir. Variational inference with normalizing flows. In *ICML*, 2015.

Salimans, Tim and Kingma, Diederik P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training GANs. In *NIPS*, 2016.

Sutherland, Dougal J, Tung, Hsiao-Yu, Strathmann, Heiko, De, Soumyajit, Ramdas, Aaditya, Smola, Alex, and Gretton, Arthur. Generative models and model criticism via optimized maximum mean discrepancy. In *ICLR*, 2017.

Theis, Lucas, Oord, Aäron van den, and Bethge, Matthias. A note on the evaluation of generative models. In *ICLR*, 2016.

Uria, Benigno, Murray, Iain, and Larochelle, Hugo. RNADE: The real-valued neural autoregressive density-estimator. In *NIPS*, 2013.

Wu, Yuhuai, Burda, Yuri, Salakhutdinov, Ruslan, and Grosse, Roger. On the quantitative analysis of decoder-based generative models. In *ICLR*, 2017.

Zhao, Junbo, Mathieu, Michael, and LeCun, Yann. Energy-based generative adversarial network. In *ICLR*, 2017.

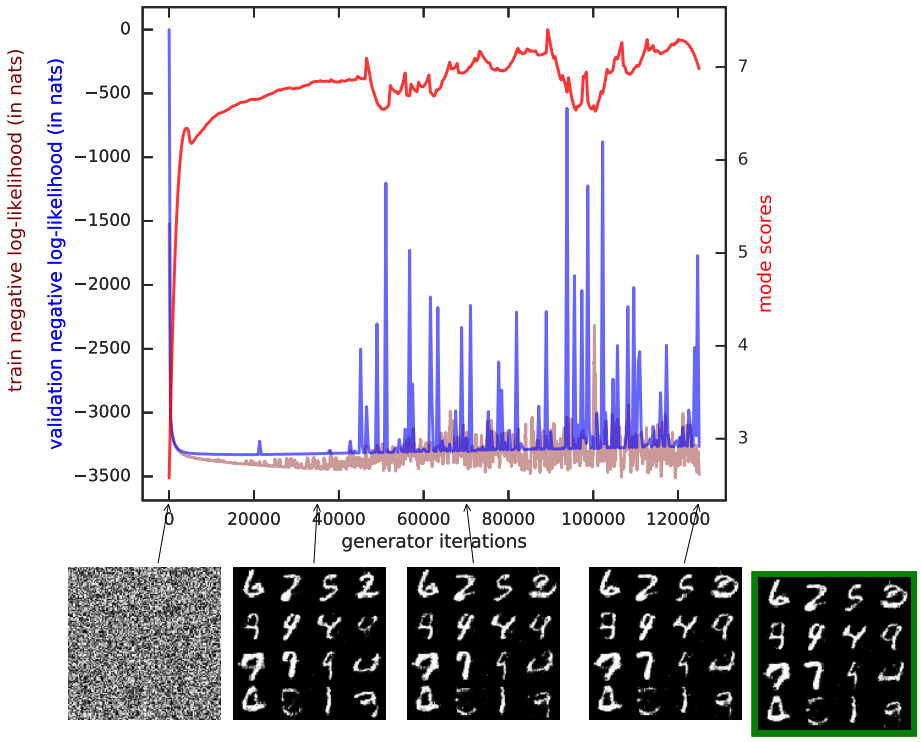


Figure 2. MNIST: The log-likelihoods from MLE are impressive, but the MODE scores are poor.

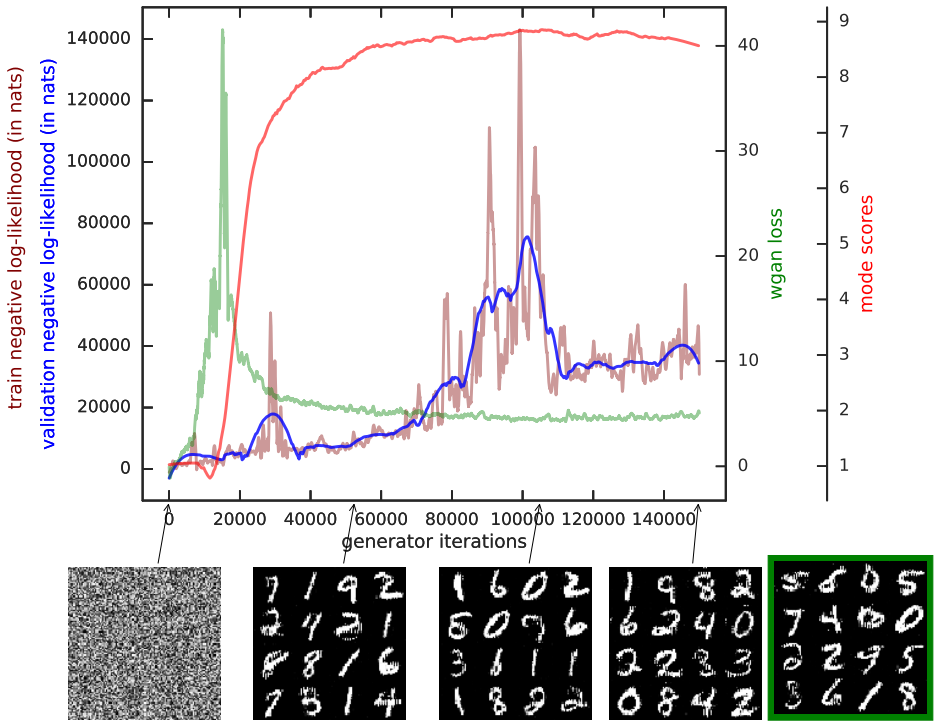


Figure 3. MNIST: WGAN attains high MODE scores, but poor log-likelihoods.

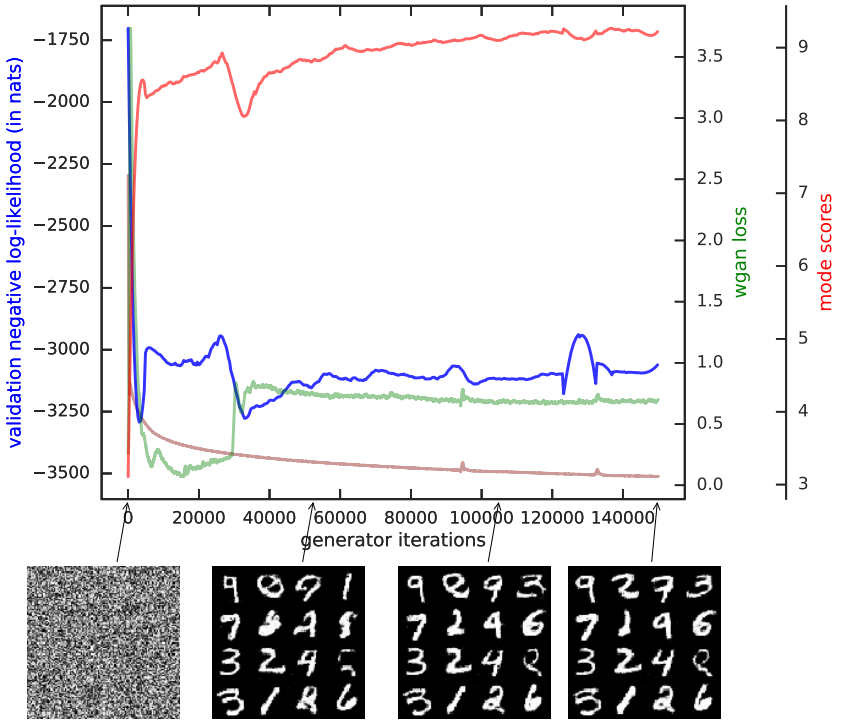


Figure 4. MNIST: Flow-GAN attains higher log-likelihoods and higher MODE scores than both pure MLE and adversarial-based learning.

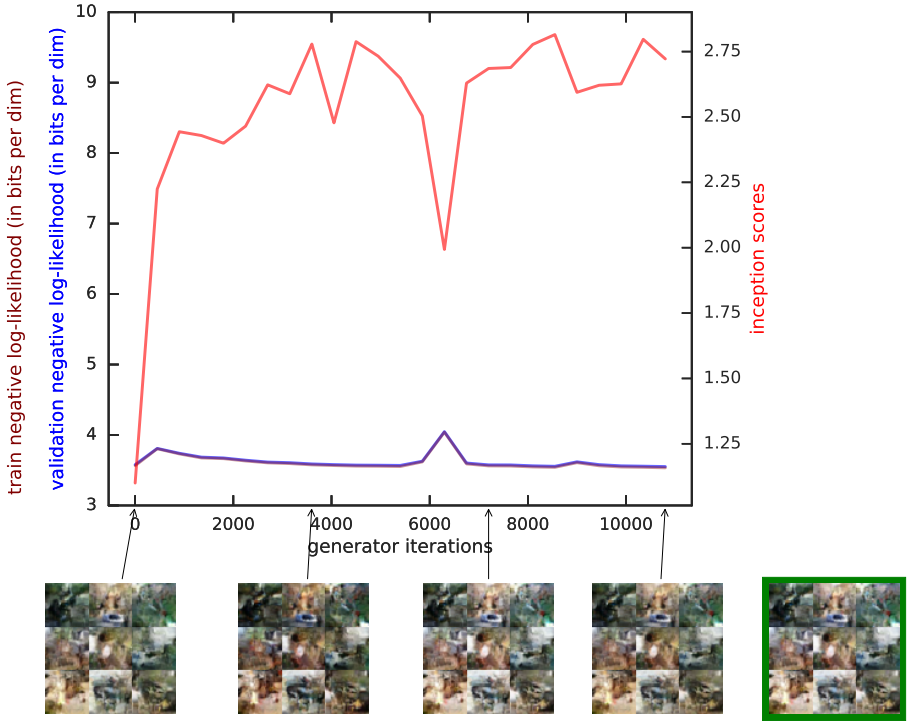


Figure 5. CIFAR-10: The log-likelihoods from MLE are impressive, but the Inception scores are poor.

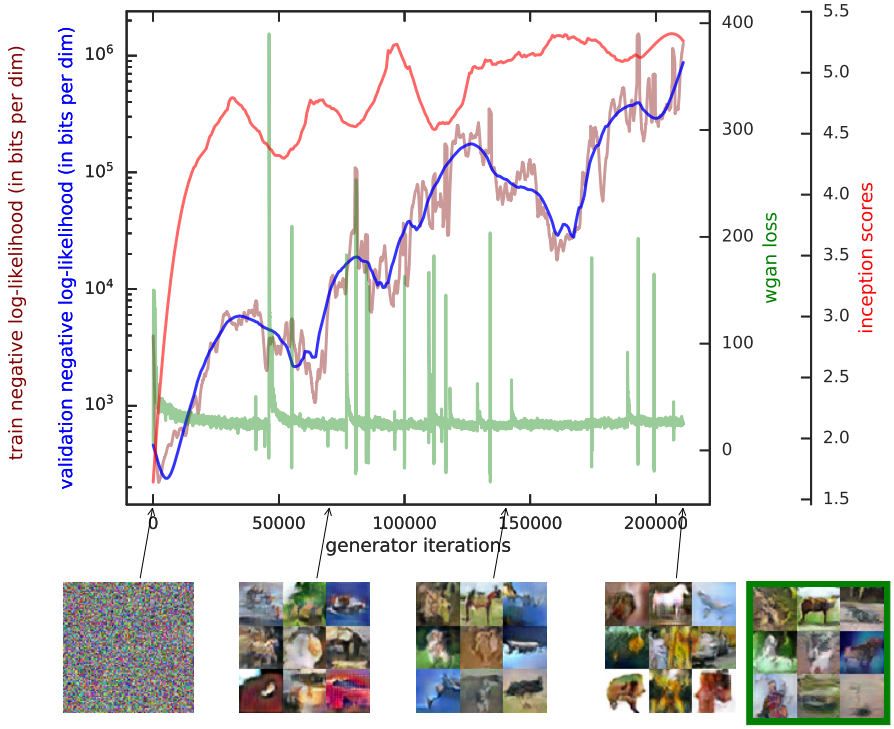


Figure 6. CIFAR-10: WGAN attains very high Inception scores, but poor log-likelihoods.

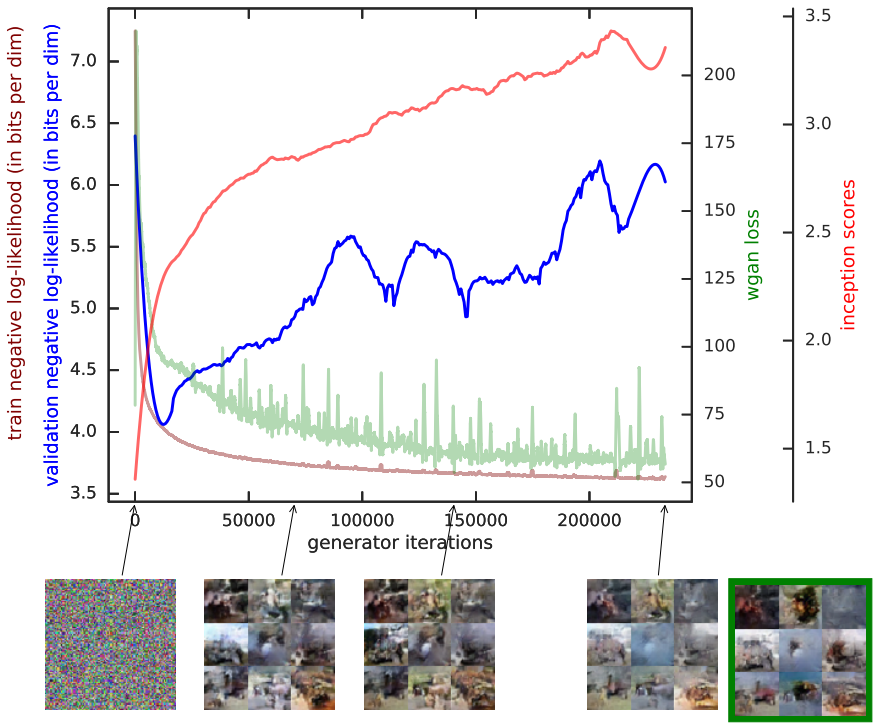


Figure 7. CIFAR-10: Flow-GAN attains reasonably high log-likelihoods and Inception scores.