Metropolis-Hastings Generative Adversarial Networks

Ryan Turner Uber AI Labs
Jane Hung Uber AI Labs
Yunus Saatci Uber AI Labs
Jason Yosinski Uber AI Labs

Abstract

We introduce the Metropolis-Hastings generative adversarial network (MH-GAN), which combines aspects of Markov chain Monte Carlo and GANs. The MH-GAN draws samples from the distribution implicitly defined by a GAN's discriminator-generator pair, as opposed to sampling in a standard GAN which draws samples from the distribution defined by the generator. It uses the discriminator from GAN training to build a wrapper around the generator for improved sampling. With a perfect discriminator, this wrapped generator samples from the true distribution on the data exactly even when the generator is imperfect. We demonstrate the benefits of the improved generator on multiple benchmark datasets, including CIFAR-10 and CelebA, using DCGAN and WGAN.

1 Introduction

Traditionally, density estimation is done with a model that can compute the data likelihood. Generative adversarial networks (GANs) [9] present a radically new way to do density estimation: They implicitly represent the density of the data via a classifier that distinguishes real from generated data.

GANs iterate between updating a discriminator D and a generator G, where G generates new (synthetic) samples of data, and D attempts to distinguish samples of G from the real data. In the typical setup, D is thrown away at the end of training, and only G is kept for generating new synthetic data points. In this work we propose the Metropolis-Hastings GAN (MH-GAN), a GAN that constructs a new generator G' that "wraps" G using the information contained in D. This principle is illustrated in Figure 1.[1]

The MH-GAN uses Markov chain Monte Carlo (MCMC) methods to sample from the distribution implicitly defined by the discriminator D learned for the generator G. This is built upon the notion that the discriminator classifies between the generator G and a data distribution:

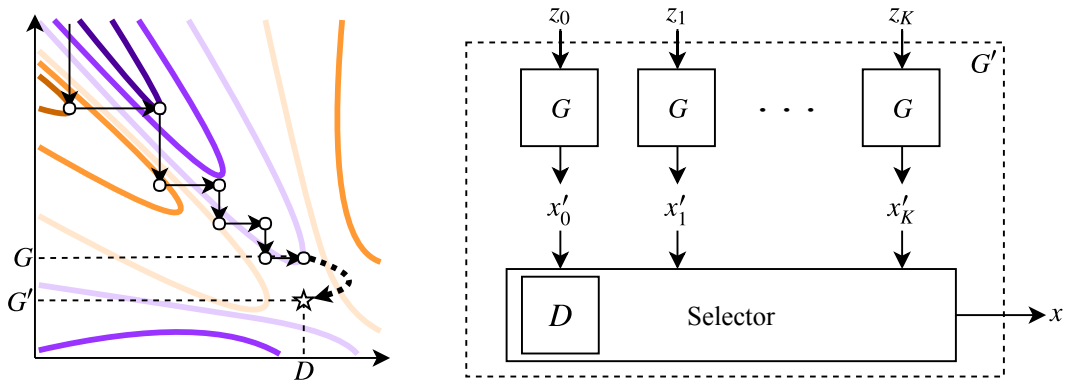
$$D(\mathbf{x}) = \frac{p_D(\mathbf{x})}{p_D(\mathbf{x}) + p_G(\mathbf{x})}\,, \tag{1}$$

where p_G is the (intractable) density of samples from the generator G, and p_D is the data density *implied* by the discriminator D with respect to G. If GAN training reaches its global optimum then this discriminator distribution p_D is equal to the data distribution and the generator distribution ($p_D = p_{\text{data}} = p_G$) [9]. Furthermore, if the discriminator D is optimal for a fixed imperfect generator G then the implied distribution still equals the data distribution ($p_D = p_{\text{data}} \neq p_G$).

We use an MCMC *independence sampler* [21] to sample from p_D by taking multiple samples from G. Amazingly, using the algorithm we present, one can show that given a perfect discriminator D and a decent (but imperfect) generator G, one can obtain exact samples from the true data distribution p_{data}. Standard MCMC implementations require (unnormalized) densities for the target p_D and

[1] The code for this project is available at:
`https://github.com/uber-research/metropolis-hastings-gans`

Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada.



(a) GAN value function

(b) G' wraps G

Figure 1: **(a)** We diagram how training of D and G in GANs performs coordinate descent on the joint minimax value function, shown in the solid black arrow. If GAN training produces a perfect D for an imperfect G, the MH-GAN wraps G to produce a perfect generator G', as shown in the final dashed arrow. The generator G moves vertically towards the orange region while the discriminator D moves horizontally towards the purple. **(b)** We illustrate how the MH-GAN is essentially a selector from multiple draws of G. In the MH-GAN, the selector is built using a Metropolis-Hastings (MH) acceptance rule from the discriminator scores D.

the proposal p_G, which are both unavailable for GANs. However, the Metropolis-Hastings (MH) algorithm requires only the ratio

$$\frac{p_D(\mathbf{x})}{p_G(\mathbf{x})} = \frac{D(\mathbf{x})}{1 - D(\mathbf{x})}, \tag{2}$$

which we can obtain using only evaluation of $D(\mathbf{x})$.

2 Related Work

A few other works combine GANs and MCMC in some way. Song et al. [19] use a GAN-like procedure to train a RealNVP [6] MCMC proposal for sampling an externally provided target $p^\star$. Whereas Song et al. [19] use GANs to accelerate MCMC, we use MCMC to enhance the samples from a GAN. Similar to Song et al. [19], Kempinska and Shawe-Taylor [11] improve proposals in particle filters rather than MCMC. Song et al. [19] was recently generalized by Neklyudov et al. [15].

The GAN approach to density estimation is complementary to the earlier *density ratio estimation* (DRE) approach [20]. In DRE the generator G is fixed, and the density is found by combining Bayes' rule and the learned classifier D. In GANs, the key is learning G well; while in DRE, the key is learning D well. The MH-GAN has flavors of both in that it uses both G and D to build G'.

2.1 Discriminator Rejection Sampling

A very similar concurrent work from Azadi et al. [2] proposes discriminator rejection sampling (DRS) for GANs, which performs rejection sampling of outputs of G by using the probabilities given by D. This approach is conceptually simpler than our approach at first but suffers from two major shortcomings in practice. First, it is necessary to find an upper-bound on D over all possible samples in order to obtain a valid proposal distribution for rejection sampling. Because this is not possible, one must instead rely on estimating this bound by drawing many pilot samples. Secondly, even if one were to find a good bound, the acceptance rate would become very low due to the high-dimensionality of the sampling space. This leads Azadi et al. [2] to use an extra γ heuristic to shift the logit D scores, making the model sample from a distribution different from p_{data} even when D is perfect. We use MCMC instead, which was invented precisely as a replacement for rejection sampling in higher dimensions. We further improve the robustness of MCMC via use of a *calibrator* on the discriminator to get more accurate probabilities for computing acceptance probabilities.

3 Background and Notation

In this section, we briefly review the notation and equations with MCMC and GANs.

3.1 MCMC Methods

MCMC methods attempt to draw a chain of samples $\mathbf{x}_{1:K} \in \mathcal{X}^K$ that marginally come from a target distribution $p^\star$. We refer to the initial distribution as p_0 and the proposal for the independence sampler as $\mathbf{x}' \sim q(\mathbf{x}'|\mathbf{x}_k) = q(\mathbf{x}')$. The proposal $\mathbf{x}' \in \mathcal{X}$ is accepted with probability

$$\alpha(\mathbf{x}', \mathbf{x}_k) = \min\left(1, \frac{p^\star(\mathbf{x}')q(\mathbf{x}_k)}{p^\star(\mathbf{x}_k)q(\mathbf{x}')}\right) \in [0, 1]\,. \tag{3}$$

If $\mathbf{x}'$ is accepted, $\mathbf{x}_{k+1} = \mathbf{x}'$, otherwise $\mathbf{x}_{k+1} = \mathbf{x}_k$. Note that when estimating the distribution $p^\star$, one must include the duplicates that are a result of rejections in $\mathbf{x}'$.

Independent samples Each chain samples $\mathbf{x}_0 \sim p_0$ and then does K MH iterations to get $\mathbf{x}_K$ as the output of the chain; in this case it is also the output of G'. Therefore, each MCMC chain results in a single sample $\mathbf{x}_K$, and independent chains are used for multiple samples from G'.

Detailed balance The detailed balance condition implies that if $\mathbf{x}_k \sim p^\star$ exactly then $\mathbf{x}_{k+1} \sim p^\star$ exactly as well. Additionally, even if $\mathbf{x}_k$ is not exactly distributed according to $p^\star$, the Kullback-Leibler (KL) divergence between the implied density it is drawn from and $p^\star$ will always decrease as k increases [14, 3].[2]

3.2 GANs

GANs implicitly model the data $\mathbf{x}$ via a synthetic data generator $G \in \mathbb{R}^d \to \mathcal{X}$:

$$\mathbf{x} = G(\mathbf{z})\,, \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)\,. \tag{4}$$

This implies a (intractable) distribution on the data $\mathbf{x} \sim p_G$. We refer to the unknown true distribution on the data $\mathbf{x}$ as p_{data}. The discriminator $D \in \mathcal{X} \to [0, 1]$ is a soft classifier predicting if a data point is real as opposed to being sampled from p_G. If D converges optimally for a fixed G, then $D = p_{\text{data}}/(p_{\text{data}} + p_G)$, and if both D and G converge then $p_G = p_{\text{data}}$ [9]. GAN training forms a game between D and G. In practice D is often better at estimating the density ratio than G is at generating high-fidelity samples [18]. This motivates wrapping an imperfect G to obtain an improved G' by using the density ratio information encapsulated in D.

4 Methods

In this section we show how to sample from the distribution p_D implied by the discriminator D. We apply (2) and (3) for a target of $p^\star = p_D$ and proposal $q = p_G$:

$$\frac{p_D}{p_G} = \frac{1}{D^{-1} - 1} \implies \alpha(\mathbf{x}', \mathbf{x}_k) = \min\left(1, \frac{D(\mathbf{x}_k)^{-1} - 1}{D(\mathbf{x}')^{-1} - 1}\right)\,. \tag{5}$$

The ratio p_D/p_G is computed entirely from the discriminator scores D. If D is perfect, $p_D = p_{\text{data}}$, so the sampler will marginally sample from p_{data}. A toy one-dimensional example with just such a perfect discriminator is shown in Figure 2.

Calibration The probabilities for D must not merely provide a good AUC score, but must also be well *calibrated*. Put in other terms, if one were to warp the probabilities of the perfect discriminator in (1) it may still suffice for standard GAN training, but it will not work in the MCMC procedure defined in (5), as it will result in erroneous density ratio values.

[2] The curious reader may wonder why we do not simply stop the chain after the first accept. In general, allowing chain length K to be conditioned in any way on the state of the chain (including which samples were accepted or rejected) has the potential to introduce bias to the samples [4].

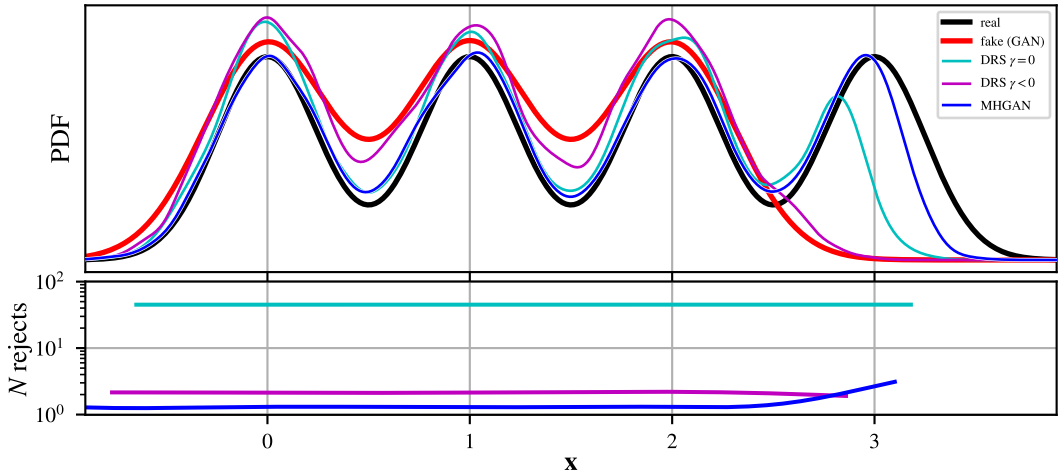


Figure 2: Illustration comparing the MH-GAN setup with the formulation of DRS on a univariate example. This figures uses a p_{data} of four Gaussian mixtures while p_G is missing one of the mixtures. The top row shows the resulting density of samples, while the bottom row shows the typical number of rejects before accepting a sample at that $\mathbf{x}$ value. The MH-GAN recovers the true density except in the far right tail where there is an exponentially small chance of getting a sample from the proposal p_G. DRS with $\gamma = 0$ shift should also be able to recover the density exactly, but it has an even larger error in the right tail. These errors arise because we must approximate the max D score and use only 10,000 pilot samples to do so, as in Azadi et al. [2]. Additionally, due to the large maximum D, it needs a large number of draws before a single accept. DRS with γ shift is much more sample efficient, but completely misses the right mode as the setup invalidates the rejection sampling equations. The MH-GAN is more adaptive in that it quickly accepts samples for the areas p_G models well; more MCMC rejections occur before accepting a sample in the right poorly modeled mode. In all cases the MH-GAN is more efficient than DRS without γ shift. Presumably, this effect becomes greater in high dimensions.

To calibrate D, we use a *held out* calibration set (10% of training data) and either logistic, isotonic, or beta [12] regression to warp the output of D. Furthermore, for WGAN, calibration is required as it does not learn a typical GAN probabilistic discriminator.

We also detect miscalibration of D using the statistic of Dawid [5] on held out samples $\mathbf{x}_{1:N}$ and real/fake labels $y_{1:N} \in \{0,1\}^N$. If D is well calibrated, i.e., y is indistinguishable from a $y \sim \text{Bern}(D(\mathbf{x}))$,

$$Z := \frac{\sum_{i=1}^N y_i - D(\mathbf{x}_i)}{\sqrt{\sum_{i=1}^N D(\mathbf{x}_i)(1 - D(\mathbf{x}_i))}} \in \mathbb{R} \implies Z \sim \mathcal{N}(0,1)\,. \tag{6}$$

This means that for large values of Z, such as when $|Z| > 2$, we reject the hypothesis that D is well-calibrated.

Initialization We also avoid the burn-in issues that usually plague MCMC methods. Recall that via the detailed balance property [8, Ch. 1], if the marginal distribution of a Markov chain state $\mathbf{x} \in \mathcal{X}$ at time step k matches the target p_D ($\mathbf{x}_k \sim p_D$), then the marginal at time step $k+1$ will also follow p_D ($\mathbf{x}_{k+1} \sim p_D$). In most MCMC applications it is not possible to get an initial sample from the target distribution ($\mathbf{x}_0 \sim p_D$). However, in MH-GANs, we are sampling from the *data* distribution, so we may use a trick: Initialize the chain at a sample of real data (the correct distribution) to avoid burn-in. If no generated sample is accepted by the end of the chain, restart sampling from a synthetic sample to ensure the initial real sample is never output. To make these restarts rare, we set K large (often 640).

Perfect Discriminator The assumption of a perfect D may be weakened for two reasons: (A) Because we recalibrate the discriminator, actual probabilities can be incorrect as long as the decision boundary between real and fake is correct. (B) Because the discriminator is only ever evaluated at samples from G or the initial real sample $\mathbf{x}_0$, D only needs to be accurate on the manifold of samples from the generator p_G and the real data p_{data}.

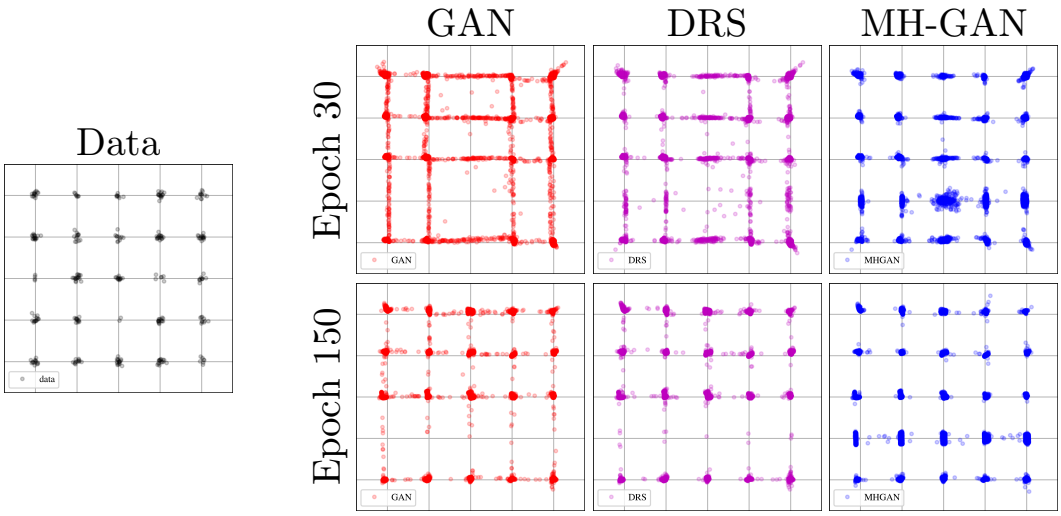


Figure 3: The 25 Gaussians example. We show the state of the generators at epoch 30 (when MH-GAN begins showing large gains) on the top row and epoch 150 (the final epoch) on the bottom row. The MH-GAN corrects areas of mis-assigned mass in the original GAN. DRS appears visually closer to the original GAN than the data, whereas the MH-GAN appears closer to the actual data.

5 Results

We first show an illustrative synthetic mixture model example followed by real data with images.

5.1 Mixture of 25 Gaussians

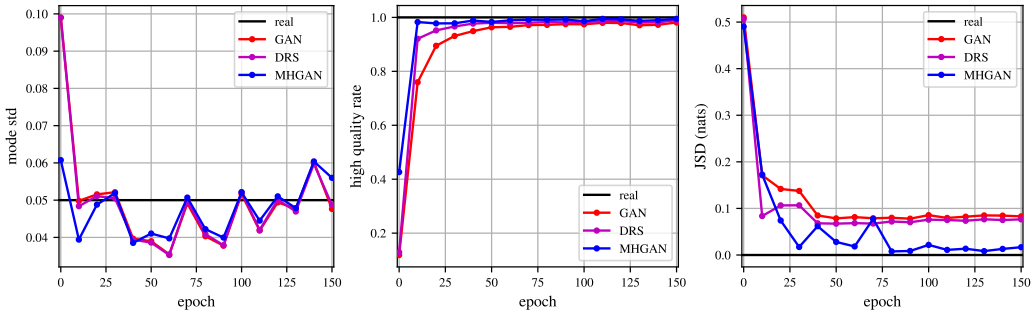
We consider the 5×5 grid of two-dimensional Gaussians used in Azadi et al. [2], which has become a popular toy example in the GAN literature [7]. The means are arranged on the grid $\mu \in \{-2, -1, 0, 1, 2\}$ and use a standard deviation of $\sigma = 0.05$.

Experimental setup Following Azadi et al. [2], we use four fully connected layers with ReLU activations for both the generator and discriminator. The final output layer of the discriminator is a sigmoid, and no nonlinearity is applied to the final generator layer. The latent $\mathbf{z} \in \mathbb{R}^2$ and observed $\mathbf{x} \in \mathbb{R}^2$ vectors have dimension two. All hidden layers have size 100. We used $64{,}000$ training points and generated $10{,}000$ points in test. The training data was standardized before training.

Visual results In Figure 3, we show the original data along with samples generated by the GAN. We also show samples enhanced via the MH-GAN (with calibration) and with DRS. The standard GAN creates spurious links along the grid lines between modes. It also missed some modes along the bottom row. DRS is able to "clean up" some of the spurious links but not fill in the missing modes. The MH-GAN recovers these under-estimated modes and further "clean up" the spurious links.

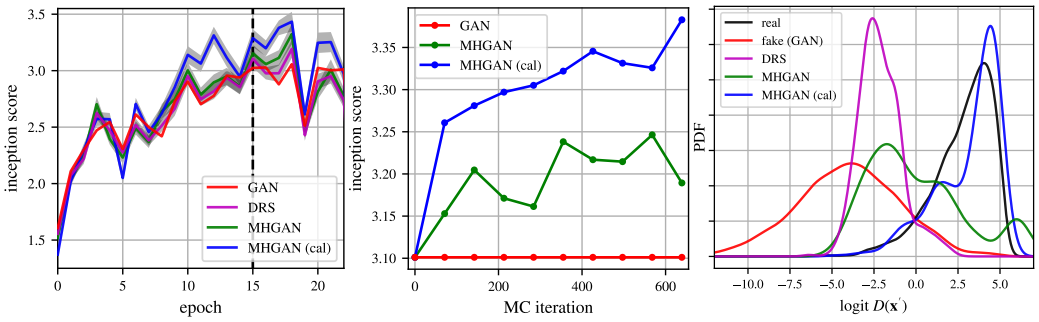
Quantitative results These results are made more quantitative in Figure 4, where we follow some of the metrics for the example from Azadi et al. [2]. We consider the standard deviations within each mode in Figure 4a and the rate of "high quality" samples in Figure 4b. A sample is assigned to a mode if its L_2 distance is within four standard deviations ($\leq 4\sigma = 0.2$) of its mean. Samples within four standard deviations of any mixture component are considered "high quality". The within standard deviation plot (Figure 4a) shows a slight improvement for MH-GAN, and the high quality sample rate (Figure 4b) approaches 100% faster for the MH-GAN than the GAN or DRS.

To test the spread of the distribution, we inspect the categorical distribution of the closest mode. Far away (non-high quality) samples are assigned to a 26th unassigned category. This categorical distribution should be uniform over the 25 real modes for a perfect generator. To assess generator quality, we look at the Jensen-Shannon divergence (JSD) between the sample mode distribution and a uniform distribution. This is a much more stringent test of appropriate spread of probability mass than checking if a single sample is produced near a mode (as was done in Azadi et al. [2]).



(a) mode std. dev. (b) high quality rate (c) Jensen-Shannon divergence

Figure 4: Results of the MH-GAN experiments on the mixture of 25 Gaussians example. On the left, we show the standard deviation of samples within a single mode. The black lines represent values for the true distribution. In the center, we show the high quality rate (samples near a real mode) across different GAN setups. On the right, we show the Jensen-Shannon divergence (JSD) between the distribution on the nearest mode vs a uniform, which is the generating distribution on mixture components. The MH-GAN shows, on average, a $5\times$ improvement in JSD over DRS. We considered adding error bars to these plots via a bootstrap analysis, but the error bars are too small to be visible.



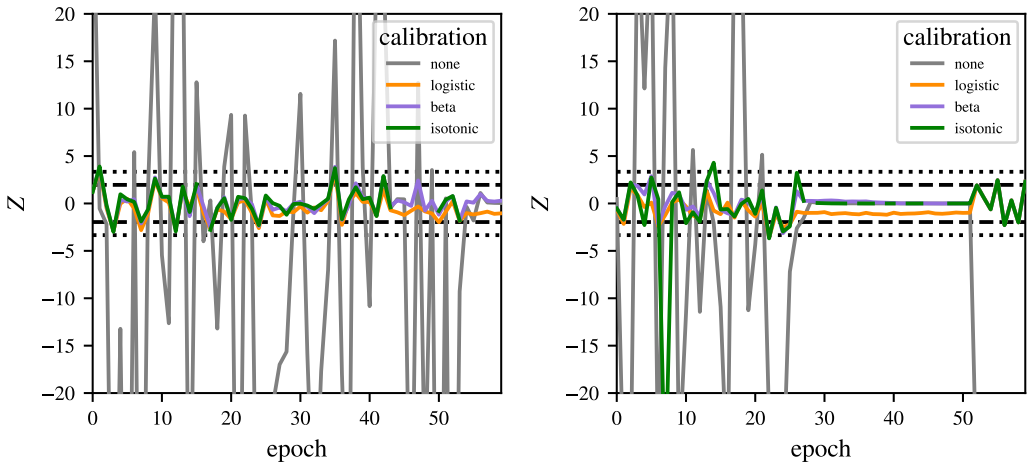
(a) performance by epoch (b) performance by MCMC iteration (c) epoch 13 scores

Figure 5: Results of the MH-GAN experiments on CIFAR-10 using the DCGAN. On the left, we show the Inception score by training epoch of the DCGAN at $K = 640$. MH-GAN denotes using the raw discriminator scores and "MH-GAN (cal)" for the calibrated scores. The error bars on MH-GAN performance (in gray) are computed using a t-test on the variation per batch across 80 splits of the Inception score. In the center we show the Inception score vs. MCMC iteration k for the GAN at epoch 15. On the right, we show the scores at epoch 13 where there is some overlap between the scores of fake and real images. When there is overlap, the MH-GAN corrects the p_G distribution to have scores looking similar to the real data. DRS fails to fully shift the distribution because 1) it does not use calibration and 2) its "γ shift" setup violates the validity of rejection sampling.

In Figure 4c, we see that the MH-GAN improves the JSD over DRS by $5\times$ on average, meaning it achieves a much more balanced spread across modes. DRS fails to make gains after epoch 30. Using the principled approach of the MH-GAN along with calibrated probabilities ensures a correct spread of probability mass.

5.2 Real Data

For real data experiments we considered the CelebA [13] and CIFAR-10 [22] data sets modeled using the DCGAN [16] and WGAN [1, 10]. To evaluate the generator G', we plot Inception scores [17] per epoch in Figure 5a after $K = 640$ MCMC iterations. The actual performance boost realized by MH-GAN oscillates from one epoch to the next, perhaps due to fluctuations in the density ratio estimation performance per epoch. Accordingly the statistical significance of the boost from G to G' with calibration varies from no significant change to boost with $p < 10^{-4}$. Figure 5b shows Inception score per MCMC iteration: most gains are made in the first $k = 100$ iterations, but smaller gains continue to $k = 400$.



(a) CIFAR-10 (b) CelebA

Figure 6: We show the calibration statistic Z (6) for the discriminator on held out data for the DCGAN. The results for CIFAR-10 are shown on the right, and CelebA on the left. The raw discriminator is clearly miscalibrated being far outside the region expected by chance (dashed black), and after multiple comparison correction (dotted black). All the calibration methods give roughly equivalent results. CelebA has a period of training instability during epochs 30–50 which gives trivially calibrated classifiers.

Table 1: Results showing Inception score improvements from MH-GAN on DCGAN and WGAN at epoch 60. Like Figure 5a, the error bars and p-values are computed using a paired t-test across Inception score batches. All results except for DCGAN on celebA are significant at $p < 10^{-4}$. WGAN does not learn a typical GAN discriminator that outputs a probability, so calibration is actually required in this case.

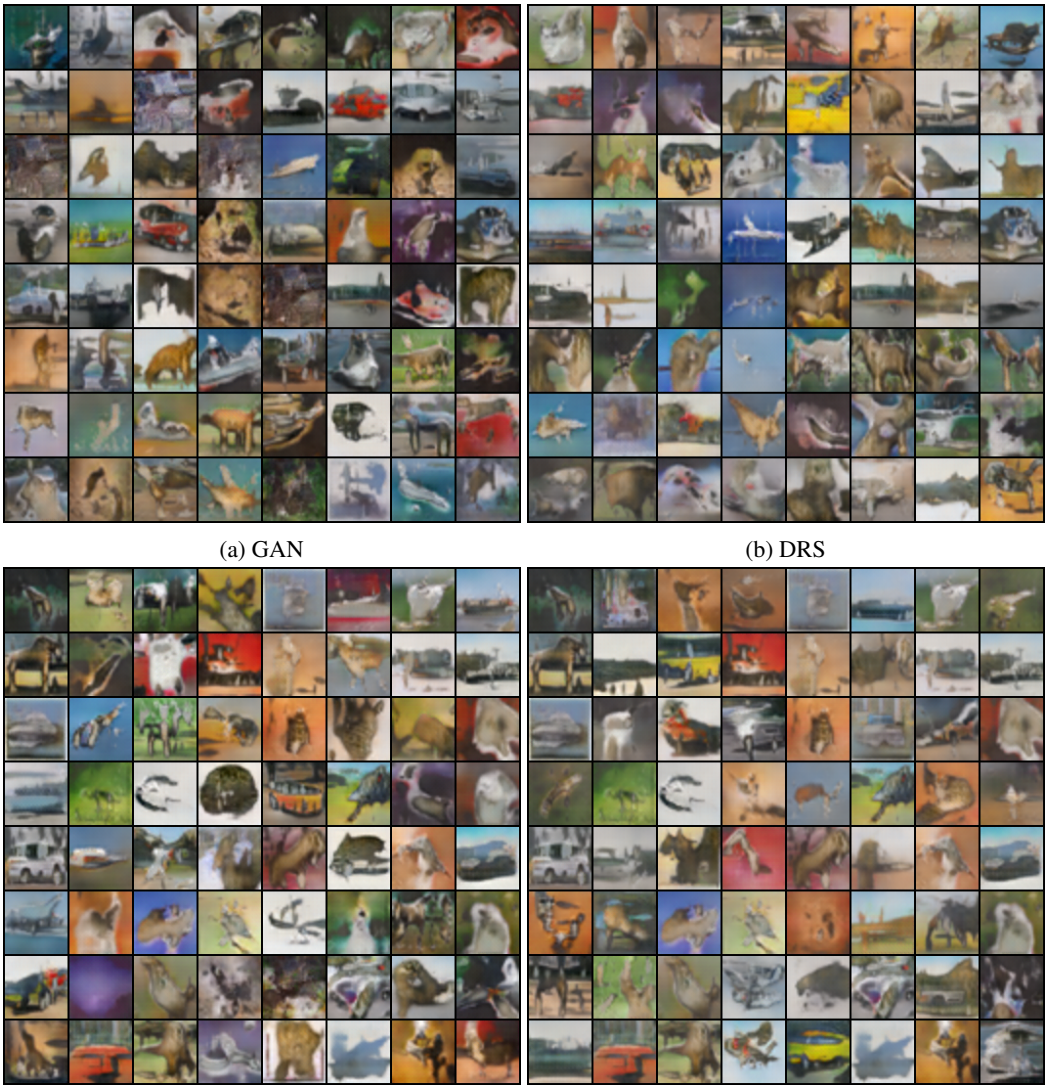
	DCGAN				WGAN			
	CIFAR-10	p	CelebA	p	CIFAR-10	p	CelebA	p
GAN	2.8789	–	2.3317	–	3.0734	–	2.7876	–
DRS	2.977(77)	0.0131	2.511(50)	<0.0001				
DRS (cal)	3.073(80)	<0.0001	2.869(67)	<0.0001	3.137(64)	0.0497	2.861(66)	0.0277
MH-GAN	3.113(69)	<0.0001	2.682(50)	<0.0001				
MH-GAN (cal)	**3.379**(66)	<0.0001	**3.106**(64)	<0.0001	**3.305**(83)	<0.0001	**2.889**(89)	0.0266

In Table 1, we summarize performance (Inception score) across all experiments, running MCMC with $K = 640$ iterations in all cases. Behavior is qualitatively similar to that in Figure 5a. While DRS improves on a direct GAN, MH-GAN improves Inception score more in every case. Calibration helps in every case. There was not a substantial difference between different calibration methods, but we found a slight advantage for isotonic regression. Results are computed at epoch 60, and as in Figure 5a, error bars and p-values are computed using a paired t-test across Inception score batches. All results are significantly better than the baseline GAN at $p < 0.05$.

Score distribution In Figure 5c, we visualize what G' does to the distribution on discriminator scores. MCMC shifts the distribution of the fakes to match the distribution on true images.

Calibration results Figure 6 shows the results per epoch for both CIFAR-10 and CelebA. It shows that the raw discriminator is highly miscalibrated, but can be fixed with any of the standard calibration methods. The Z statistic for the raw discriminator D (DCGAN on CIFAR-10) varies from -77.57 to 48.98 in the first 60 epochs; even after Bonferroni correction at N=60, we expect $|Z| < 3.35$ with 95% confidence for a calibrated classifier. The calibrated discriminator varies from -2.91 to 3.60, which shows it is almost perfectly calibrated. Accordingly, it is unsurprising that the calibrated discriminator significantly boosts performance in the MH-GAN.

Visual results Finally, we also show example images from the CIFAR-10 and CelebA setups in Figures 7 and 8. The selectors (such as MH-GAN) result in a wider spread of probability mass across background colors. For CIFAR-10, it enhances modes with animal-like outlines and vehicles.



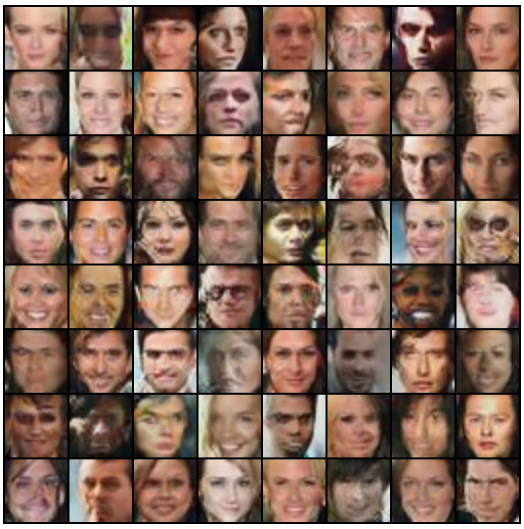
(a) GAN (b) DRS

(c) MH-GAN (d) MH-GAN (cal)

Figure 7: Example images on CIFAR-10 for different GAN setups. The different selectors (MH-GAN and DRS) are run on the same batch of images. Meaning, the same images may appear for both generators. The calibrated MH-GAN shows a greater preference for animal-like images with four legs.

(a) GAN

(b) DRS



(c) MH-GAN

(d) MH-GAN (cal)

Figure 8: Example images on CelebA for different GAN setups. Like Figure 7, the same batch of images goes into each selector.

6 Conclusions

We have shown how to incorporate the knowledge in the discriminator D into an improved generator G'. Our method is based on the premise that D is better at density ratio estimation than G is at sampling data, which is inherently a harder task. The principled MCMC setup selects among samples from G to correct biases in G. This is the only method in the literature which has the property that given a perfect D one can recover G such that $p_G = p_{\text{data}}$. We have shown the raw discriminators in GANs and DRS are poorly calibrated. To our knowledge, this is the first work to evaluate the discriminator in this way and to rigorously show the poor calibration of the discriminator. The MH-GAN has great potential for extension.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the International Conference on Machine Learning*, volume 70, pages 214–223, 2017.

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

[3] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

[4] M. K. Cowles, G. O. Roberts, and J. S. Rosenthal. Possible biases induced by MCMC convergence diagnostics. *Journal of Statistical Computation and Simulation*, 64(1):87–104, 1999.

[5] A. P. Dawid. Prequential analysis. *Encyclopedia of Statistical Sciences*, 1:464–470, 1997.

[6] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

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

[8] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. Introducing Markov chain Monte Carlo. *Markov chain Monte Carlo in practice*, 1:19, 1996.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems*, pages 2672–2680. Curran Associates, Inc., 2014.

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028v3*, 2017.

[11] K. Kempinska and J. Shawe-Taylor. Adversarial sequential Monte Carlo. In *Bayesian Deep Learning (NIPS Workshop)*, 2017.

[12] M. Kull, T. S. Filho, and P. Flach. Beta calibration: A well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 54, pages 623–631, 2017.

[13] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.

[14] I. Murray and R. Salakhutdinov. Notes on the KL-divergence between a Markov chain and its equilibrium distribution. 2008.

[15] K. Neklyudov, P. Shvechikov, and D. Vetrov. Metropolis-Hastings view on variational inference and adversarial training. *arXiv preprint arXiv:1810.07151*, 2018.

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

[17] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242. Curran Associates, Inc., 2016.

[18] N. Shibuya. Understanding generative adversarial networks. `https://towardsdatascience.com/understanding-generative-adversarial-networks-4dafc963f2ef`, 2017.

[19] J. Song, S. Zhao, and S. Ermon. A-NICE-MC: Adversarial training for MCMC. In *Proceedings of Advances in Neural Information Processing Systems*, pages 5140–5150. Curran Associates, Inc., 2017.

[20] M. Sugiyama, T. Suzuki, and T. Kanamori. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, 2012.

[21] L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4): 1701–1728, 1994.

[22] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.