Learning Energy-Based Models With Adversarial Training — Supplementary Materials

Xuwang Yin, Shiying Li, and Gustavo K. Rohde

University of Virginia
{xy4cm, sl8jx, gustavo}@virginia.edu

1 Proof of Proposition (1)

Proposition 1. *The optimal solution of* $\max_D \min_{p_T} U(D, p_T)$ *is* $U(D^*, p_T^*) = -\log(4)$*, where* D^* *outputs* $\frac{1}{2}$ *on Supp*(p_{data}) *and* $\leq \frac{1}{2}$ *outside Supp*(p_{data})*, and* p_T^* *is supported in the contour set* $\{D = \frac{1}{2}\}$*.*

Proof. Let

$$p_T^* = \arg\min_{p_T} \mathbb{E}_{\mathrm{x}\sim p_T}[\log(1 - D(x))], \tag{1}$$

then

$$\max_D \min_{p_T} U(D, p_T) = \max_D U(D, p_T^*). \tag{2}$$

We solve $\max_D U(D, p_T^*)$ by first deriving its upper bound. Let $\alpha = \max_{\mathcal{X}} D$, then $\mathbb{E}_{\mathrm{x}\sim p_T^*}[\log(1-D(x))]$ is minimized when p_T^* is supported in $\{x : D(x) = \alpha\}$ With this result, we can derive an upper bound of $U(D, p_T^*)$:

$$\begin{aligned}
&U(D, p_T^*) \\
&= \int_{\mathcal{X}} p_{\text{data}}(x) \log D(x) dx + \int_{\mathcal{X}} p_T^*(x) \log(1 - D(x)) dx \\
&= \int_{\mathcal{X}} p_{\text{data}}(x) \log D(x) dx + \int_{\mathcal{X}} p_T^*(x) \log(1 - \alpha) dx \\
&\leq \int_{\mathcal{X}} p_{\text{data}}(x) \log(\alpha) dx + \int_{\mathcal{X}} p_T^*(x) \log(1 - \alpha) dx \\
&= \log(\alpha) + \log(1 - \alpha) \\
&\leq -\log(4),
\end{aligned} \tag{3}$$

where the last inequality follows from the fact that the function $f(\alpha) = \log(\alpha) + \log(1-\alpha)$ achieves its maximum value of $-\log(4)$ at $\alpha = \frac{1}{2}$. It is not hard to see that equality holds if and only if i) $\max_{\mathcal{X}} D = \frac{1}{2}$, ii) $D = \frac{1}{2}$ on $\text{Supp}(p_{\text{data}})$, and iii) $\text{Supp}(p_T^*) \subseteq \{x : D(x) = \frac{1}{2}\}$. In summary, $\max_D \min_{p_T} U(D, p_T)$ achieves its optimal value of $-\log(4)$ at (D^*, p_T^*) where

$$D^*(x) = \begin{cases} \frac{1}{2} & x \in \text{Supp}(p_{\text{data}}) \\ \leq \frac{1}{2} & x \in \mathcal{X} \setminus \text{Supp}(p_{\text{data}}) \end{cases}, \tag{4}$$

and p_T^* is supported in the contour set $\{D = \frac{1}{2}\}$.

2 Connection to GANs

In this section we provide a comparative analysis of the proposed AT generative model and GANs [8]. The proposed approach learns data distribution by solving the maximin problem

$$\max_{D} \min_{p_T} U(D, p_T) = \mathbb{E}_{\mathrm{x}\sim p_{\mathrm{data}}}[\log D(x)] + \mathbb{E}_{\mathrm{x}\sim p_T}[\log(1 - D(x))], \tag{5}$$

while GANs learn a generator function G by solving the minimax problem

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{\mathrm{x}\sim p_{\mathrm{data}}}[\log D(x)] + \mathbb{E}_{\mathrm{z}\sim p_z}[\log(1 - D(G(z)))]. \tag{6}$$

The generator G implicitly defines a distribution p_g by mapping a prior distribution p_z from a low-dimensional latent space $\mathcal{Z} \subseteq \mathbb{R}^z$ to the high-dimensional data space $\mathcal{X} \subseteq \mathbb{R}^d$. Plugging p_g into Eq. (6), we get:

$$\min_{p_g} \max_{D} U(D, p_g) = \mathbb{E}_{\mathrm{x}\sim p_{\mathrm{data}}}[\log D(x)] + \mathbb{E}_{\mathrm{x}\sim p_g}[\log(1 - D(x))] \tag{7}$$

Comparing Eq. (5) with Eq. (7) we find both problems making use of the standard log-likelihood objective for binary classification, but have a reversed order of minimization and maximization. In fact, both formulations solve a two-player zero-sum game, a mathematical representation of a situation in which one player's gain is balanced by another player's loss. This game can be described by the *payoff function* $f : \mathbb{R}^{p+q} \to \mathbb{R}$, which represents the amount of payment that one player (player 1) makes to the other player (player 2). The goal of player 1 is to choose a strategy $u \in \mathbb{R}^p$ such that the payoff is minimized, while the goal of player 2 is to choose a strategy $u \in \mathbb{R}^q$ such that the payoff is maximized. Depending on the order of maximization and minimization, the best strategies for both players, and the optimal payoff, can be solved via $\min_u \max_v f(u, v)$ or $\max_v \min_u f(u, v)$.

In Eq. (5), $U(D, p_T)$ is the payoff function, and the goal of player p_T is to choose a strategy p_T^* such that the payoff is minimized, whereas the goal of player D is to choose a strategy D^* such that the payoff is maximized. This *maximin* game is played by following such a rule: player D makes the first move by choosing a D; player p_T, after learning that player D has made the move, will choose a p_T to minimize its payment, which results in a payoff of $\min_{p_T} U(D, p_T)$; player D, who is informed of player p_T's strategy, will chooses a D such that the worse case payoff $\min_{p_T} U(D, p_T)$ is maximized, which results in an overall payoff of $\max_D \min_{p_T} U(D, p_T)$. The best strategies of both players and the maximum payoff can be derive from Proposition 1 : In the maximin game $\max_D \min_{p_T} U(D, p_T)$, the best strategy for player D is to choose a D^* that outputs $\frac{1}{2}$ on $\mathrm{Supp}(p_{\mathrm{data}})$ and $\leq \frac{1}{2}$ outside $\mathrm{Supp}(p_{\mathrm{data}})$, the best strategy for player p_T is to choose a p_T^* which is supported in $\{x : D(x) = \frac{1}{2}\}$, and the maximum payoff is $-\log(4)$.

In Eq. (7), $U(D, p_g)$ is the payoff function. Similar to Eq. (5), the goal of player p_g is to minimize the payoff, and the goal of player D is to maximize

the payoff. In contrast to Eq. (5), player p_g makes the first move. The solution to this minimax game is analyzed in [8]: the best strategy of player p_g is to choose a p_g^* which minimizes the Jensen-Shannon divergence (JSD) between p_g and p_{data}: $p_g^* = \arg\min_{p_g} \text{JSD}(p_g \parallel p_{\text{data}}) = p_{\text{data}}$, and the best strategy of player D is to choose $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_g^*(x)} = \frac{1}{2}$. Under these strategies, the payoff function U measures the JSD between p_g and p_{data}: $U(D^*, p_g^*) = -\log(4) + 2 \cdot \text{JSD}(p_g^* \parallel p_{\text{data}}) = -\log(4)$, which coincides with the U solution in the maximin game. Note that in the minimax game, D^* does not need to be defined outside $\text{Supp}(p_g) \cup \text{Supp}(p_{\text{data}})$ [8].

The optimal solutions to these two formulations are summarized in Tab. A1.

The pseudo code for solving the minimax problem is outlined in Algorithm A1. Fig. A1 shows the simulation results in two settings where p_0 data is respectively uniformly distributed (left panel) and concentrated in the lower left corner (right panel). In can be seen that in both cases p_T^* matches p_{data} when the algorithm converges. The right panel shows that when p_0 data is concentrated in the lower left corner, the D solution has undefined outputs outside $\text{Supp}(p_{\text{data}})$.

We find these two formulations giving rise to different applications. The minimax formulation is ideal for learning a generator model that can produce a distribution that matches p_{data}. The discriminator, because of its undefined behavior outside $\text{Supp}(p_{\text{data}})$, may not be very useful for some downstream tasks such as out-of-distribution detection. In the maximin formulation, as we have discussed in the main text, can be used for sample generation, image-to-image translation, image restoration such as denoising and inpainting, and (worst-case) out-of-distribution detection.

Table A1. Optimal solutions to the minimax problem and maximin problem

	Minimax (GANs) Eq. (7)	Maximin (ours) Eq. (5)
p_T^*/p_g^*	$p_g^* = p_{\text{data}}$	p_T^* is supported in $\{x : D(x) = \frac{1}{2}\}$
D^*	$D^*(x) = \frac{1}{2}$ on $\text{Supp}(p_{\text{data}})$, undefined outside $\text{Supp}(p_{\text{data}})$	$D^*(x) = \frac{1}{2}$ on $\text{Supp}(p_{\text{data}})$, $D^*(x) \leq \frac{1}{2}$ outside $\text{Supp}(p_{\text{data}})$
$U(D^*, p_g^*)/U(D^*, p_T^*)$	$-\log(4)$	$-\log(4)$

Algorithm A1 Solving the minimax problem

1: Draw samples $\{x_i\}_{i=1}^m$ from p_{data}, and samples $\{x_i^*\}_{i=1}^m$ from p_0.
2: **repeat**
3: Update D by maximizing $\frac{1}{m}\sum_{i=1}^m \log D(x_i) + \frac{1}{m}\sum_{i=1}^m \log(1 - D(x_i^*))$ (until converge).
4: For each $x \in \{x_i^*\}_{i=1}^m$, update its value by $x \leftarrow x + \lambda \frac{\nabla D(x)}{\|\nabla D(x)\|_2}$ (single step).
5: **until** $\{x_i^*\}_{i=1}^m = \{x_i\}_{i=1}^m$

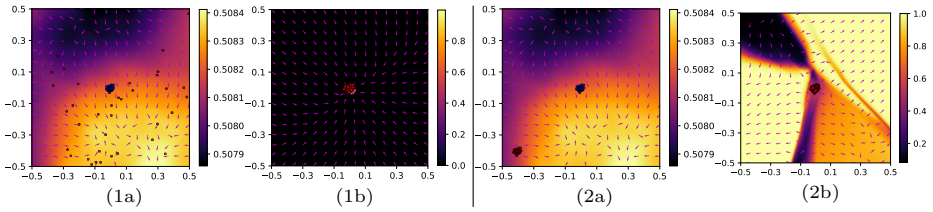


Fig. A1. Plots of contours and (normalized) gradient vector fields of the D functions learned with different p_0 data. Left and right panel respectively show the initial state (1a and 2a) and final state (1b and 2b) of D when p_0 data is respectively uniformly distributed (red points in 1a) and concerntrated in the lower left corner (red points in 2a). p_{data} is a Gaussian distribution centered at $(0, 0)$ (blue points).

3 Experimental Setups

Model Architecture. On CIFAR-10 we use the standard ResNet50 [10] architecture with ReLU activation for the D model. On CelebA-HQ 256, AFHQ-CAT 256, and LSUN-Church 256 we use a customized architecture (Tab. A2) adapted from [3] .

Table A2. Network architecture for the D model used in CelebA-HQ 256, AFHQ-CAT 256, and LSUN-Church 256.

Layer	Resample	Output shape
Conv1 × 1	-	256 × 256 × 64
ResBlock	AvgPool	128 × 128 × 128
ResBlock	AvgPool	64 × 64 × 256
ResBlock	AvgPool	32 × 32 × 512
ResBlock	AvgPool	16 × 16 × 512
ResBlock	AvgPool	8 × 8 × 512
ResBlock	AvgPool	4 × 4 × 512
LeakyReLU	-	4 × 4 × 512
Conv4 × 4	-	1 × 1 × 512
LeakyReLU	-	1 × 1 × 512
Reshape	-	512
Linear	-	1

Datasets. We evaluate our method on CIFAR-10 [15] (50K training samples), CelebA-HQ 256 [13] (30K training samples), AFHQ-CAT [3] dataset (5153 training samples), and LSUN-Church [21] (126227 training samples). AFHQ [3] is a recently introduced benchmark dataset for image-to-image translation.

Evaluation Metrics. We use Inception Score (IS) [16] and FID score [12] to evaluate the quality of generated samples. We follow [14] and compute the FID score between 50k generated samples and all training samples (IS is also calculated on the generated 50K samples). We use the original code from [16] and [12] to calculate the scores. For OOD detection, we use area under the ROC curve (AUROC) as the evaluation metric.

Training. We use Algorithm 2 to train the models. The training hyperparameters for each task can be found in Tab. A3. We in addition perform perform 5-steps PGD attack, random resized cropping, random horizontal flipping on p_{data} samples to mitigate overfitting. The performance (FID score) of the model is monitored during training and the best-performing model to used to report the final FID score.

The CIFAR-10 worst-case OOD detection model is trained using in- and out-distribution adversarial training [1], where in-distribution AT uses a l^2-ball of radius 0.25 and PGD attacks of steps 10 and step-size 0.1, and out-distribution AT uses a l^2-ball of radius 0.5 and PGD attacks of steps 10 and step-size 0.1. Following [1], we use a batch size of 128 and use the recommended AutoAugment policy from [5]. The model is trained for 400 epochs using a SGD optimizer with a fixed learning rate of 0.1.

Table A3. Training hyperparameters. We use $\beta_1 = 0.0, \beta_2 = 0.99$ for the Adam optimizer.

	CIFAR-10	CelebA-HQ 256	AFHQ-CAT 256	LSUN-Church
Batch size	32	40	40	32
Training iterations	172K	218K	225K	215K
Optimizer	Adam	Adam	Adam	Adam
Learning rate	5e-4	5e-5	5e-5	5e-5
K	0,...,25	0,...,40	0,...,25	0,...,35
Epochs per K	5	5	50	1
PGD attack step-size	0.1	2.0	2.0	2.0
R_1 regularization	0.01	30	100	100

Sample Generation. The generated samples for FID and IS evaluation are produced by performing PGD attacks on 50K samples randomly drawn from the p_0 dataset. The settings for the p_0 dataset and the PGD attack can be found in Table A4.

Table A4. Sample generation setting

Task	p_0 dataset	PGD step size	PGD steps
CIFAR-10	80 million tiny images [18]	0.2	32
CelebA-HQ 256	ImageNet [6]	8.0	20
AFHQ-CAT 256	ImageNet [6]	8.0	14
LSUN-Church 256	ImageNet [6]	8.0	17

4 Extended Experiment Results

4.1 Training and Test Time Sampling Efficiency

Tab. A5 shows that our method has competitive training and test time sampling efficiency to state-of-the-art EBMs. Although VAEBM typically requires much fewer update steps than our method, its per-step efficiency is much worse (Tab. A6), suggesting that its VAE component has considerable computational complexity. We also observe that the quality of our generated samples is not sensitive to the number of sampling steps as long as the overall perturbation (#step $\times$ step-size) remains the same (Tab. A7). This allows us to use a much larger step size than the one used during training to speedup test time sampling in real applications.

Table A5. The number of update steps in the PGD attack (our method) and Langevin dynamics (other methods). "PCD" refers to using a persistent sampling chain.

	Ours	VAEBM [20]	CF-EBM [22]	JEM [9]
CIFAR-10 (train)	25	6 (PCD)	50	20 (PCD)
CIFAR-10 (test)	32	16	50	100
CelebA-HQ 256 (train)	40	6 (PCD)	90	N/A
CelebA-HQ 256 (test)	20	24	90	N/A

Table A6. Number of steps and wall-clock time to generate 50 CIFAR-10 samples. Data of NCSN and VAEBM are from [20].

Model	Steps	Wall-clock time	GPU device
NCSN [17]	1000	107.9 seconds	RTX Titan
VAEBM [20]	16	8.79 seconds	RTX Titan
Ours	32	2.34 seconds	RTX 2080 Ti

Table A7. FID scores of samples generated using different combinations of number of steps and step-size.

	Number of steps × step-size	FID
CIFAR-10	64 × 0.1	13.07
	32 × 0.2	13.21
	16 × 0.4	13.49
CelebA-HQ 256	40 × 4.0	19.19
	20 × 8.0	18.97
	10 × 16.0	19.19

4.2 Extend Results on Worst-Case Out-Of-Distribution Detection

Tab. A8 shows that under a PGD adversary with l^2 radius 7.0 our model exhibits strong out-distribution robustness. (Note that according to [1], a perturbation of 7.0 is already large enough to make undefended models (e.g., OE [11]) fail completely at the OOD detection task). When we further increase the perturbation limit to 100, the AUC scores decrease to near 0, suggesting that obfuscated gradients did not occur.

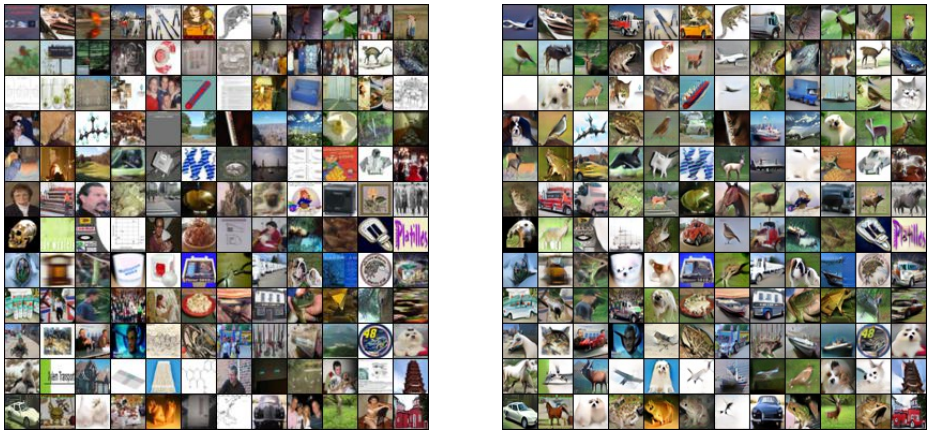
Table A8. OOD detection results on 256 × 256 datasets. Each entry shows the AUC score on clean OOD samples (left value) and AUC score on adversarial OOD samples (right value). Adversarial OOD samples are computed by maximizing the model output in a l^2-ball of radius 7.0 or 100.0 around OOD samples via Auto-PGD [4] with 100 steps and 5 random restarts. Results are computed using 1024 in-distribution samples and 1024 out-distribution samples.

Threat model	Out-distribution dataset	In-distribution dataset		
		CelebA-HQ 256	AFHQ-CAT 256	LSUN-Church 256
$\epsilon = 7.0$	Uniform noise	1.0 / 1.0	1.0 / 1.0	0.9476 / 0.9331
	SVHN	0.9967 / 0.9930	0.9944 / 0.9889	0.9668 / 0.9541
	CIFAR-10	0.9978 / 0.9985	0.9930 / 0.9902	0.9081 / 0.8707
	ImageNet validation set	0.9986 / 0.9988	0.9971 / 0.9945	0.9409 / 0.9218
	AFHQ-CAT 256	0.9984 / 0.9971	N/A	0.9691 / 0.9595
	CelebA-HQ 256	N/A	0.9900 / 0.9810	0.9794 / 0.9691
	LSUN-Church56	0.9999 / 1.0000	0.9900 / 0.9810	N/A
$\epsilon = 100$	Uniform noise	1.0000 / 0.0930	1.0 / 0.0041	0.9476 / 0.0422
	SVHN	0.9955 / 0.0487	0.9944 / 0.0039	0.9668 / 0.0656
	CIFAR-10	0.9978 / 0.0732	0.9930 / 0.0042	0.9081 / 0.0300
	ImageNet validation set	0.9967 / 0.0406	0.9971 / 0.0131	0.9409 / 0.1342
	AFHQ-CAT 256	0.9984 / 0.0843	N/A	0.9691 / 0.0536
	CelebA-HQ 256	N/A	0.9900 / 0.0023	0.9794 / 0.0957
	LSUN-Church56	0.9999 / 0.0951	0.9997 / 0.0080	N/A

4.3 Extended Results on Generation

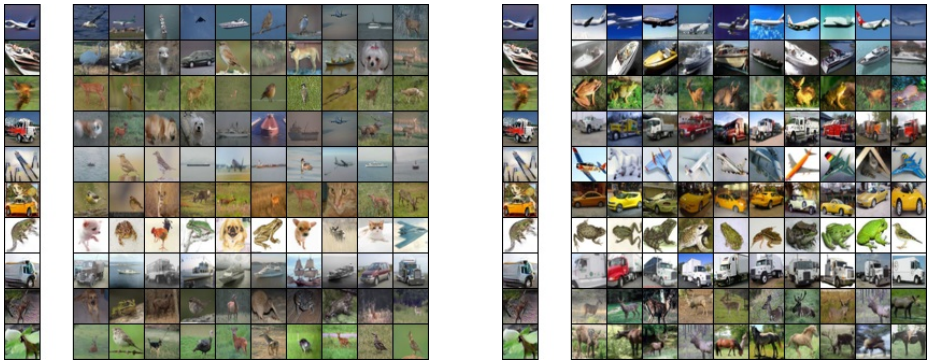
Additional results are summarized below:

- **Uncurated generation samples.** Fig. A2, Fig. A4, Fig. A6, and Fig. A7 show the uncurated generated samples on CIFAR-10, CelebA-HQ 256, AFHQ-CAT 256, and LSUN-Church 256. Note that we have used the same seed images (Fig. A18) to generated these results. We find that some generated images contain artifacts. By first applying Gaussian smoothing ($\sigma = 10$) to the source images (p_0 data), we are able to obtain more visually pleasing results (Fig. A5). The generated samples contain less artifacts, but have a slightly worse FID. The smoothing filters out high frequency components, and seems to be playing a similar role as reduced-temperature sampling [19,20] and the "truncation trick" [2], where better-looking results (typically with reduced diversity) can be generated from latent noise sampled from the high density area of the latent space.
- **Nearest Neighbor Analysis.** Fig. A3, Fig. A8 and Fig. A9 show the pixel space and inception feature space nearest neighbors of the generated samples on CIFAR-10, CelebA-HQ 256, and AFHQ-CAT 256. Note that none of the nearest neighbors resemble the generated samples, suggesting that the models have not memorized the training data.
- **Interpolation.** Fig. A10 and Fig. A11 show the interpolation results on CelebA-HQ 256 and AFHQ-CAT 256. The interpolation works reasonable well even on AFHQ-CAT where only about 5000 training images are available.
- **Intermediate Generation Results.** Fig. A12 and Fig. A13 show the intermediate generation results. It can be seen that the model is capable of transforming natural images into valid images of the target data distribution. In addition, when the number of PGD attack steps is too large, the generated samples become saturated, which suggests that the model, like many EBMs trained with short-run MCMC, do not have a valid steady-state that reflects the distribution of target data.
- **Compositional Visual Generation.** Fig. A14 shows that our model can be composed like regular EBMs [7].
- **Denosing and Inpainting.** Fig. A15 and Fig. A16 show uncurated denoising and inpainting results on CelebA-HQ 256 and AFHQ-CAT 256.
- **Image Translation.** Fig. A17 shows uncurated image translation results on CelebA-HQ 256 and AFHQ-CAT 256.



Source images Generated images

Fig. A2. Uncurated CIFAR-10 generated samples.



Generated samples (left panel) and pixel space nearest neighbors (right panel).

Generated samples (left panel) and Inception feature space nearest neighbors (right panel).

Fig. A3. Nearest neighbors of generated samples on CIFAR-10.

Fig. A4. Uncurated generated samples on CelebAHQ-256. Source images are in Fig. A18.

Fig. A5. Uncurated generated samples on CelebAHQ-256. The source images used to generate these samples are obtained by applying Gaussian blur ($\sigma = 10$) to the images in Fig. A18.

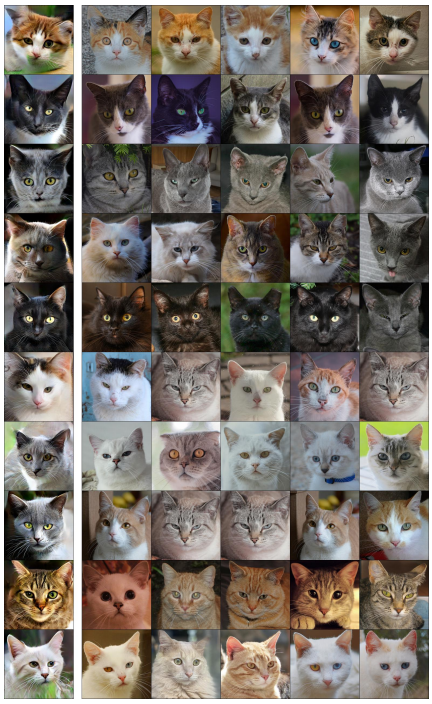
Fig. A6. Uncurated generated samples on AFHQ-CAT 256. Source images are in Fig. A18.

Fig. A7. Uncurated generated samples on LSUN-Church 256. Source images are in Fig. A18.

Generated samples (left panel) and pixel space nearest neighbors (right panel

Generated samples (left panel) and Inception feature space nearest neighbors (right panel).

Fig. A8. Nearest neighbors of generated samples on CelebA-HQ 256.



Generated samples (left panel) and pixel space nearest neighbors (right panel

Generated samples (left panel) and Inception feature space nearest neighbors (right panel).

Fig. A9. Nearest neighbors of generated samples on AFHQ-CAT 256.

Fig. A10. Interpolation results on CelebA-HQ 256. Intermediate images are generated by performing PGD attacks on linear interpolations between the source images used to generate the leftmost and rightmost samples.

Fig. A11. Interpolation results on AFHQ-CAT 256. Intermediate images are generated by performing PGD attacks on linear interpolations between the source images used to generate the leftmost and rightmost samples.

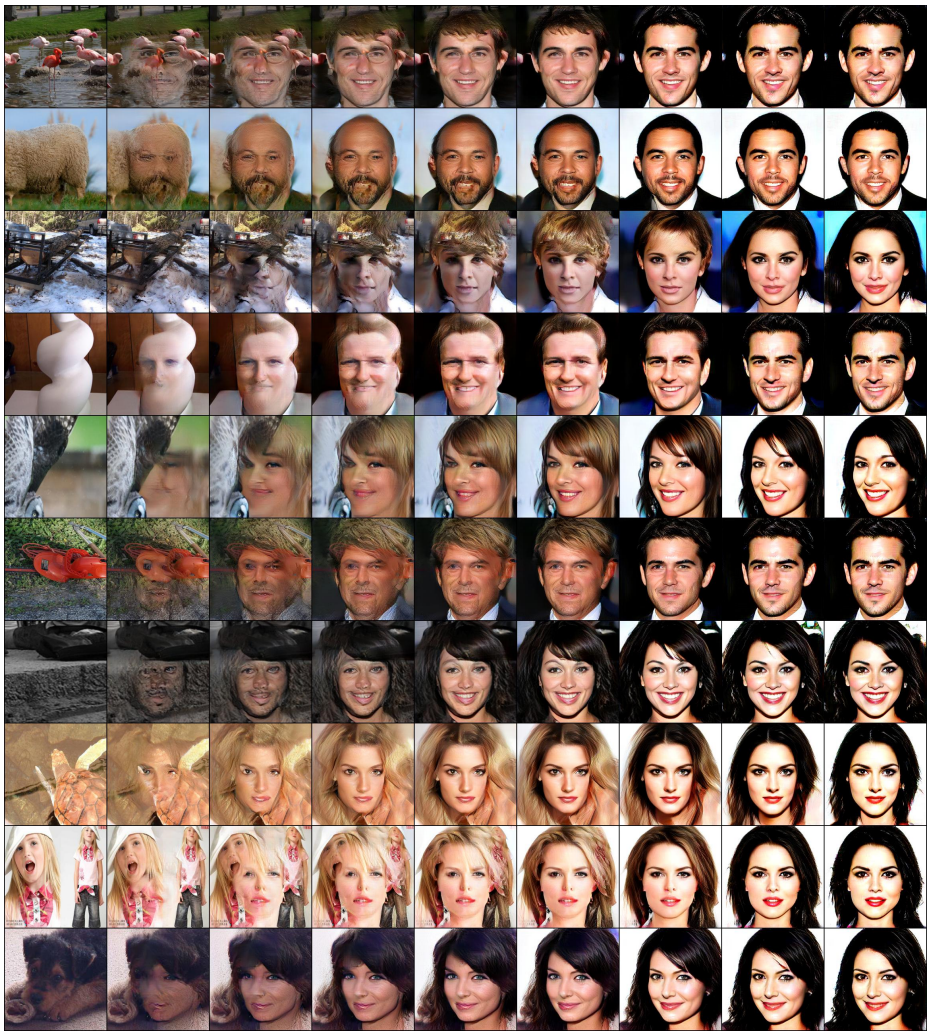


Fig. A12. CelebA-HQ 256 intermediate generation results. The PGD attack steps for column 1-9 are [0, 4, 8, 12, 16, 20, 30, 40, 50] (steps 20 has the best FID score).

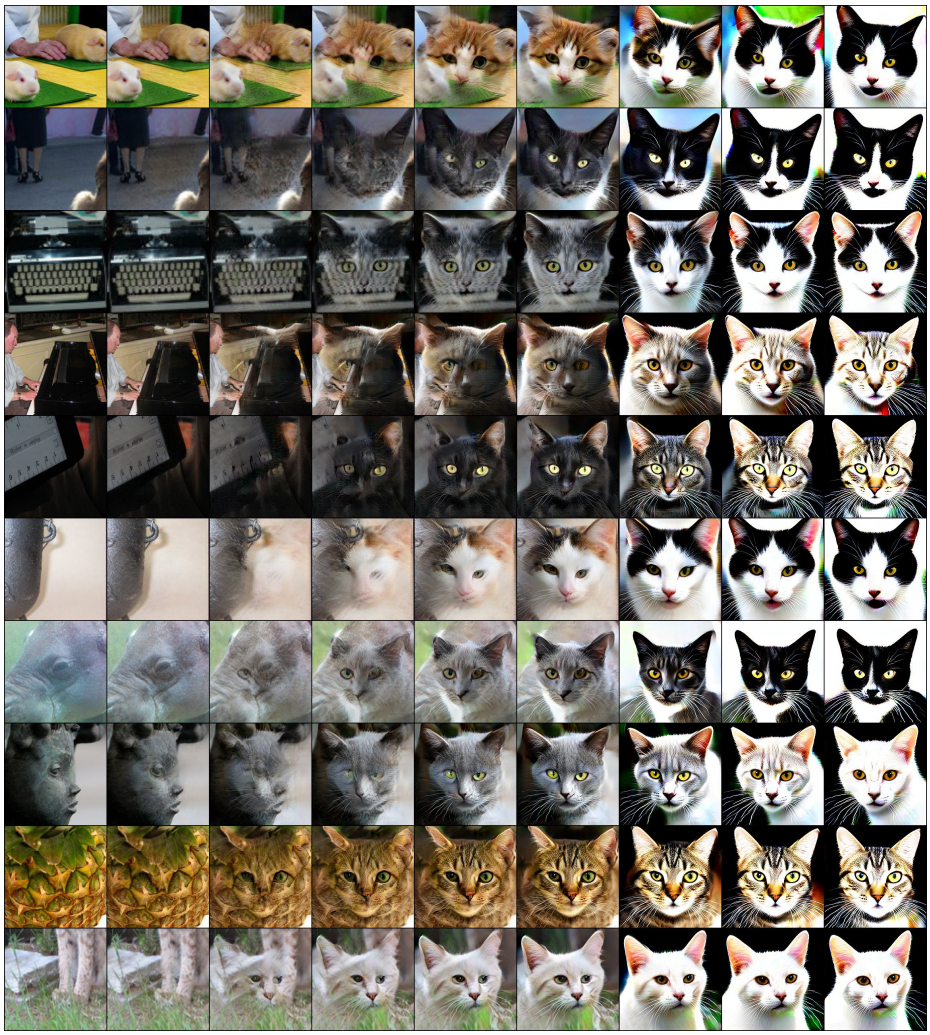


Fig. A13. AFHQ-CAT 256 intermediate generation results. The PGD attack steps for column 1-9 are [0, 2, 5, 8, 11, 14, 30, 50, 100] (steps 14 has the best FID score).

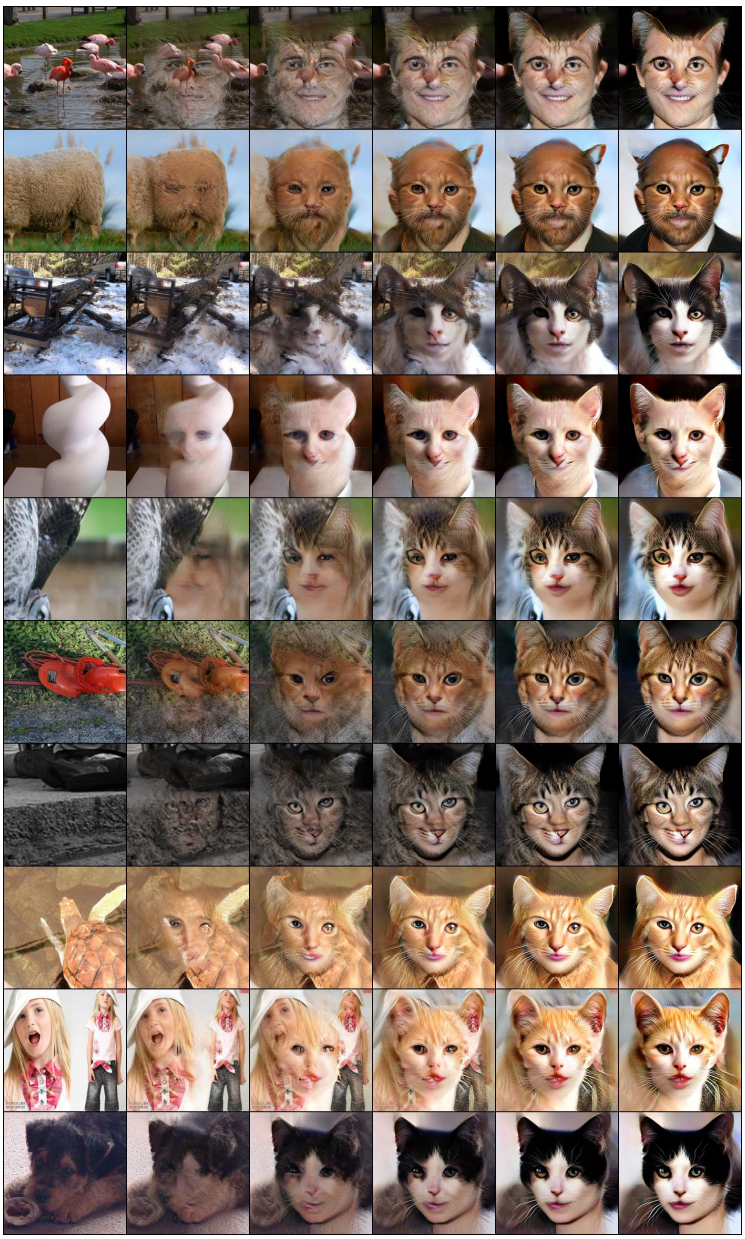
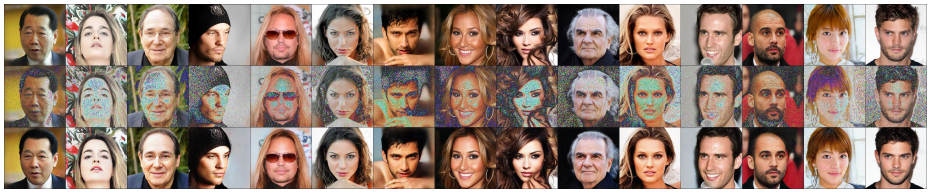
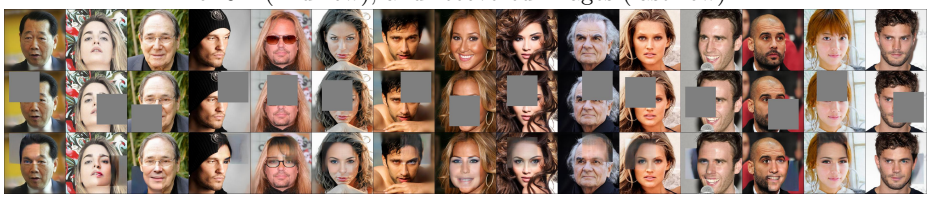


Fig. A14. Concept conjunction [7] using the CelebA-HQ model and AFHQ-CAT model. The generated samples have both human face features and cat face features.



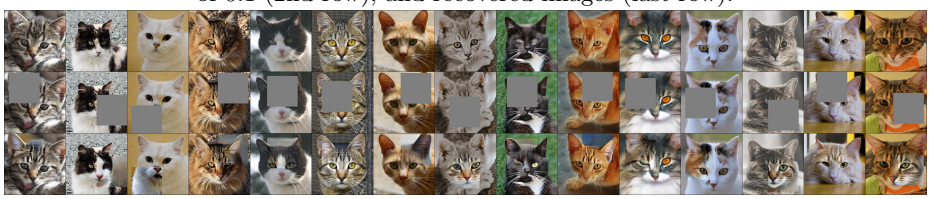
Original images (1st row), images with additive Gaussian noise of standard deviation of 0.1 (2nd row), and recovered images (last row).



Original image (1st row), occluded images (2nd row), and recovered images (last row).

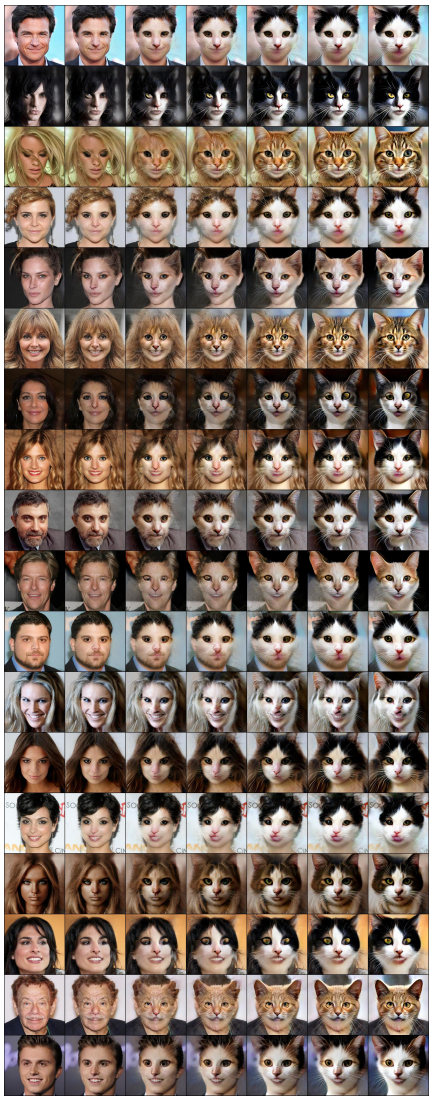
Fig. A15. Uncurated denoising and inpainting results on CelebA-HQ 256.

Original images (1st row), images with additive Gaussian noise of standard deviation of 0.1 (2nd row), and recovered images (last row).



Original image (1st row), occluded images (2nd row), and recovered images (last row).

Fig. A16. Uncurated denoising and inpainting results on AFHQ-CAT 256.



Uncurated image translation results on CelebA-HQ 256.

Uncurated image translation results on AFHQ-CAT 256.

Fig. A17. Uncurated image translation samples.

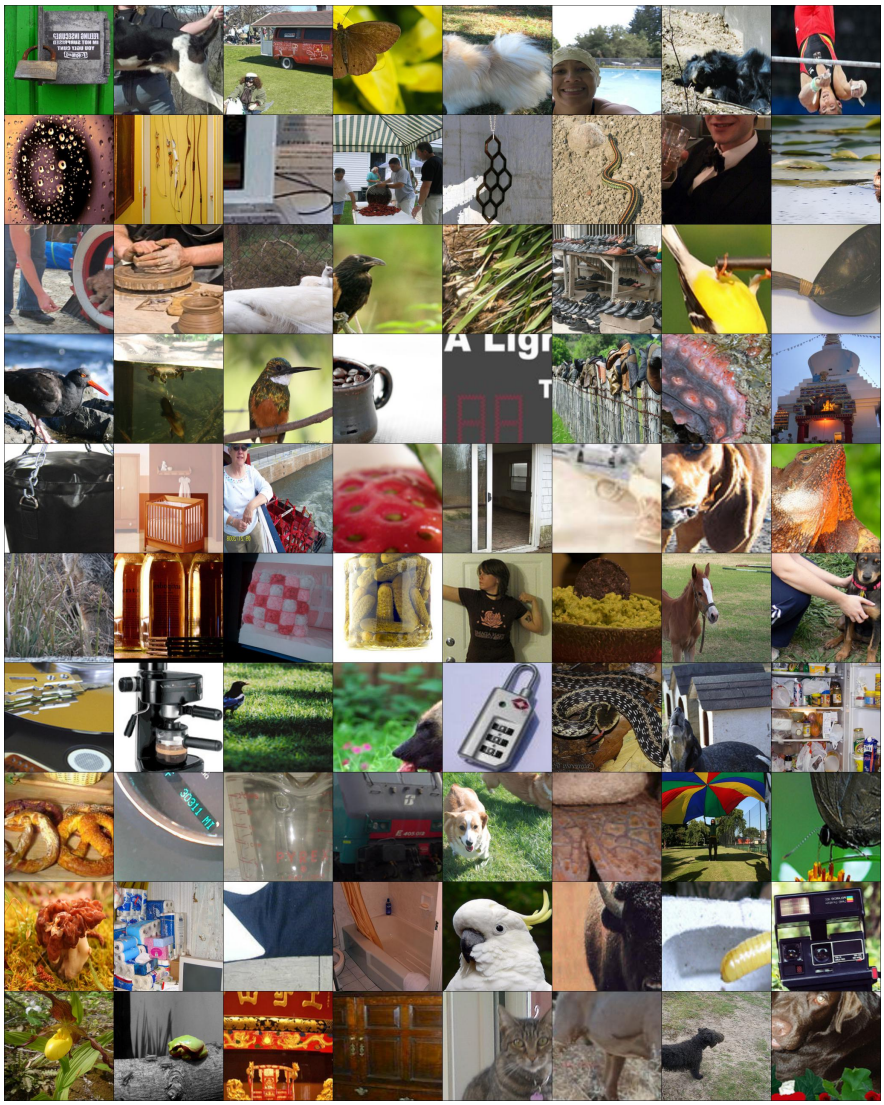


Fig. A18. Seed images used to generate samples in Fig. A4, Fig. A5, Fig. A6, and Fig. A7.

References

1. Augustin, M., Meinke, A., Hein, M.: Adversarial robustness on in-and out-distribution improves explainability. In: European Conference on Computer Vision. pp. 228–245. Springer (2020) 5, 7
2. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=B1xsqj09Fm 8
3. Choi, Y., Uh, Y., Yoo, J., Ha, J.W.: Stargan v2: Diverse image synthesis for multiple domains. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8188–8197 (2020) 4, 5
4. Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: ICML (2020) 7
5. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation strategies from data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 113–123 (2019) 5
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) 6
7. Du, Y., Li, S., Mordatch, I.: Compositional visual generation with energy based models. Advances in Neural Information Processing Systems **33**, 6637–6647 (2020) 8, 20
8. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014) 2, 3
9. Grathwohl, W., Wang, K.C., Jacobsen, J.H., Duvenaud, D., Norouzi, M., Swersky, K.: Your classifier is secretly an energy based model and you should treat it like one. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=Hkxzx0NtDB 6
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 4
11. Hendrycks, D., Mazeika, M., Dietterich, T.: Deep anomaly detection with outlier exposure. arXiv preprint arXiv:1812.04606 (2018) 7
12. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in neural information processing systems. pp. 6626–6637 (2017) 5
13. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017) 5
14. Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., Aila, T.: Training generative adversarial networks with limited data. In: Advances in Neural Information Processing Systems. vol. 33, pp. 12104–12114 (2020), https://proceedings.neurips.cc/paper/2020/file/8d30aa96e72440759f74bd2306c1fa3d-Paper.pdf 5
15. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009) 5
16. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. Advances in neural information processing systems **29**, 2234–2242 (2016) 5

17. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. In: Advances in Neural Information Processing Systems. vol. 32 (2019), https://proceedings.neurips.cc/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf 6
18. Torralba, A., Fergus, R., Freeman, W.T.: 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) 6
19. Vahdat, A., Kautz, J.: Nvae: A deep hierarchical variational autoencoder. In: Advances in Neural Information Processing Systems. vol. 33 (2020), https://proceedings.neurips.cc/paper/2020/file/e3b21256183cf7c2c7a66be163579d37-Paper.pdf 8
20. Xiao, Z., Kreis, K., Kautz, J., Vahdat, A.: Vaebm: A symbiosis between variational autoencoders and energy-based models. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=5m3SEczOV8L 6, 8
21. Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015) 5
22. Zhao, Y., Xie, J., Li, P.: Learning energy-based generative models via coarse-to-fine expanding and sampling. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=aD1_5zowqV 6