A Limitations

A major limitation of our method is the computational cost of MCMC sampling. Updating the EBM weights requires about 50 to 100 MCMC steps, and each step requires a backward pass to compute the Langevin gradient. Despite this costly step, training our model requires substantially less resources than GANs that achieve similar quality results (see Appendix C). The joint version of the Hat EBM further increases computational requirements compared to standard EBM because of the need for dual Langevin updates and the need to backprop through the generator. The conditional Hat EBM only requires a backprop through the hat network and forward pass through the generator, which has similar runtime to standard EBMs.

Another limitation of our work is that we still rely on noise-initialized shortrun sampling for our retrofit experiments, which requires that the initialization distribution in the latent space is a reasonable starting point for obtaining good latent space samples. This is done in our work by enforcing that latent vectors corresponding to images lie on a sphere of radius $\sqrt{m}$, so that sampling from a m-dimensional Gaussian is roughly aligned with the target states. In future work, we hope to develop a better latent space initialization method, perhaps by adapting cooperative learning to give latent space initialization rather than image space initialization.

B Potential Negative Impacts

Like many works in generative modeling, our work has the potential to contribute to the development of harmful images, which could take the form of images that spread misinformation, explicit content, or images that perpetuate negatives biases and stereotypes.

C Computational Requirements

Our computational resources were primarily 5 TPUv2-8 and 5 TPUv3-8 devices. For our large-scale ImageNet experiments, we used a TPUv3-32 device. Our large scale ImageNet experiment can be run on a TPUv3-32 in approximately 60 hours, coming to a total of $32 \times 60 = 1920$ TPU hours. We were also able to run our largest scale experiment on a TPUv3-8 in approximately 130 hours, coming to a total of $8 \times 130 = 1040$ TPU hours for the same experiment. Since we only get about a $2\times$ speed-up when parallelizing from 8 to 32 cores, the overall compute is lower for the TPUv3-8, although the actual runtime is longer. This runtime compares favorably with the state-of-the-art Self-Supervised GAN (SSGAN) [6] for unconditional ImageNet 128×128 synthesis, which reports a runtime of about 1.5 days using a TPUv3-128 and a cost of $36 \times 128 = 4608$ TPU hours. Despite the costly step of MCMC sampling, our relatively lightweight networks and parallelization of Langevin sampling make EBM learning feasible.

To estimate our total amount of compute, we use the following calculation. We performed experiments for this work over the course of 3 months. We estimate that on average, we had approximately 4 TPU-8 machines (either v2 or v3) running at any given time. This work done by the TPU-8 machines comes to about $8 \times 4 \times 3 \times 730 = 70080$ TPU hours. We additionally performed about 15 experiments using the TPUv3-32, which comes to about $15 \times 1920 = 28800$ TPU hours. In total, we used approximately 100K TPU hours across all experiments.

D Validity of Learning Procedures and Assumptions

Our primary theoretical claim is the validity of the joint Hat EBM and conditional Hat EBM learning procedures. The validity of the learning procedures relies on two key assumptions: correct model specification for the observed data, and the convergence of MCMC samples during the training to ensure that negative samples represent samples from the current model.

Our assumptions about the distribution of the observed data X are as follows. For the joint Hat EBM, we assume that $X \sim Y + G(Z)$ for a joint distribution (Y, Z) that can be parameterized as $U(y, z; \theta) = H(Y + G(Z); \theta)$. Since Y is unconstrained, it seems reasonable that for a suitably flexible function H, one could always learn a residual that corrects the appearance of any generator, even a poor generator. Thus the model is well-specified since $Y + G(Z)$ should be able to represent an arbitrary appearance regardless of the generator G if H has sufficient capacity. For the conditional

Hat EBM, the observed data X are assumed to follow $X = Y + G(Z)$ where $Z \sim N(0, I)$ and $U(Y|Z;\theta) = H(Y + G(Z);\theta)$. Again, as long as H has sufficient capacity, it should be able to learn residuals Y that correct the appearance of any generator.

The second assumption requires that MCMC samples converge to their steady-state to update the Hat Network. It is known from previous work that the predominant outcome of most MCMC sampling during EBM training is very far from convergence [28]. Our work operates exclusively in the non-convergent regime, and we acknowledge that the second assumption is violated. Prior work has identified that realistic synthesis is much more easily achieved in the non-convergent regime, and violating the convergence assumption is standard practice in EBM learning with the goal of synthesis.

E Architectures

All architectures except for the retrofit generator use a standard SN-GAN discriminator or generator architecture. The hat network uses a discriminator structure with spectral norm layers removed. Batch norm is set to test mode for all generators. The scaled ImageNet experiments simply double the scaling factor for channel dimensions of the EBM and Generator from 1024 to 2048 for the 128×128 SN-GAN architecture. The generator used in the CIFAR-10 retrofit experiment has a image-style latent space with dimension [16, 16, 1]. This network has the same architecture has the SN-GAN generator except that the it removes the fully connected layer at the base of the generator and replaces it with a convolutional layer, and the first two upsampling residual blocks in the original SN-GAN are converted to residual blocks that do not upsample.

F Historical Generator Update

A key aspect of our self-contained learning procedure using the Conditional Hat EBM is the use of historical EBM samples to update the generator, rather than samples from the current EBM. This dramatically improves the quality of synthesized samples. We hypothesize that using only current EBM samples to update the generator fails because the short MCMC trajectories cannot significantly change the appearance of the initial generator samples. This means that a lack of diversity in the initial generator samples will also lead to a lack of diversity of samples used to update the EBM. Lack of sample diversity can easily cause EBM instability because the EBM will rapidly change its landscape to try to cover modes that are not contained in the negative samples, causing it to forget previous modes. Once it has forgotten a previous mode, it will once again experience a rapid update to recover the forgotten mode, at the expense of forgetting another mode. Stable EBM learning requires that the initial states for MCMC sampling have reasonable diversity so that learning in many modes can take place simultaneously. We find the same problem occurs for the original cooperative learning formulation [39].

We visualize the importance of our historical update in Figure 4. This figure compares cooperative learning [39] with and without batch normalization to Hat EBM synthesis for CIFAR-10. One Hat EBM experiment uses only the current EBM to update the generator, while the other uses the historical approach we outline in the text. Neither of our Hat EBM experiments use batch norm. The Hat EBM using historical updates is by far the most successful synthesis method early in training, and we find that this advantage is maintained throughout learning. We also find that batch norm is an essential part of the original cooperative learning method because it prevents the generator from collapsing early in training.

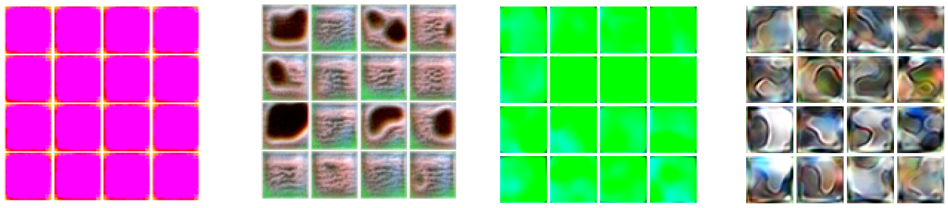
This historical update can be viewed as performing maximum likelihood where the data distribution (X, Z) is defined as the marginal of the joint distribution of (t_ℓ, X, Z) where $X = Y_K + G(Z;\phi_{t_\ell})$ and $Y_K|Z$ is sampled from

$$p(Y|Z, t_\ell) \propto -\exp\{H(Y + G(Z;\phi_{t_\ell});\theta_{t_\ell})\}.$$

This holds because

$$\begin{aligned} E_{(t_\ell,Z,X)}[-\log p_G(X, Z;\phi)] &= E_{(t_\ell,Z,X)}\left[\frac{1}{2\tau^2}\|G(Z;\phi) - X\|_2^2\right] + C \\ &\approx \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2\tau^2}\|G(Z_i;\phi) - (Y_{K,i} + G(Z_i;\phi_{t_{\ell_i}}))\|_2^2 + C \end{aligned}$$

for n samples $\{(t_{\ell_i}, Z_i, X_i)\}_{i=1}^n$.



Coop. (no batch norm) Coop. (batch norm) Hat EBM (current) Hat EBM (historical)

Figure 4: Samples after 500 weight updates for different EBM learning methods. Cooperative learning fails without batch norm because the tether between the EBM and generator leads to lack of diversity and instability. Including batch norm in cooperative learning helps add some diversity, but samples can still remain very visually similar for many updates. The Hat EBM has similar problem as cooperative learning when current EBM samples are used to update the generator. This problem is greatly alleviated by instead using historical samples to update the generator, because several past updates of the EBM have much higher diversity than a single snapshot.

G Algorithm for Tandem Training of Conditional Hat EBM

Algorithm 1 Training a Conditional Hat EBM for Image Synthesis

Require: Natural images $\{x_m^+\}_{m=1}^M$, EBM $U(x;\theta)$, generator $G(z;\phi)$ Langevin noise ε, number of shortrun steps K, EBM optimizer h_U, generator optimizer h_G, random initial weights θ_0 and ϕ_0, number of training iterations T, bank size N.
Ensure: Learned weights θ_T for EBM and ϕ_T for generator.
Initialize bank of random latent states $\{Z_i\}_{i=1}^N$ i.i.d. from the Gaussian $N(0, I)$.
Initialize image bank $\{X_i^-\}_{i=1}^N$ from generator using $X_i^- = g(Z_i;\phi_0)$
for $1 \leq t \leq T$ **do**

Steps to Update EBM
Select batch $\{\tilde{X}_b^+\}_{b=1}^B$ from data samples $\{x_m^+\}_{m=1}^M$.
Draw latent samples $\{\tilde{Z}_b\}_{b=1}^B$ i.i.d. from the Gaussian $N(0, I)$.
Initialize residual images $\{\tilde{Y}_{b,0}^-\}_{b=1}^B$ from the image with all pixels set to 0.
Update residual images $\{\tilde{Y}_{b,0}^-\}_{b=1}^B$ with K Langevin steps of Equation 6 to obtain $\{\tilde{Y}_{b,K}^-\}_{b=1}^B$. Keep $\tilde{Z}_b$ fixed.
Sum generated image and residual image using $\tilde{X}_b^- = G(\tilde{Z}_b, \phi_{t-1}) + \tilde{Y}_{b,K}^-$ to obtain negative samples $\{\tilde{X}_b^-\}_{b=1}^B$.
Get learning gradient $\Delta_U^{(t)}$ using Equation 8 with samples $\{\tilde{X}_b^+\}_{b=1}^B$ and $\{\tilde{X}_b^-\}_{b=1}^B$.
Update θ_t using gradient $\Delta_U^{(t)}$ and optimizer h_U.

Steps to Update Generator
Randomly choose unique indices $\{i_1, \ldots, i_B\} \subset \{1, \ldots, N\}$.
Get paired batches $\{Z_{i_b}\}_{b=1}^B$ and $\{X_{i_b}^-\}_{b=1}^B$ from $\{Z_i\}_{i=1}^N$ and $\{X_i^-\}_{i=1}^N$.
Get learning gradient $\Delta_G^{(t)}$ using Equation 11 with samples $\{Z_{i_b}\}_{b=1}^B$ and $\{X_{i_b}^-\}_{b=1}^B$.
Update ϕ_t using gradient $\Delta_G^{(t)}$ and optimizer h_G.
Overwrite old states $\{Z_{i_b}\}_{b=1}^B$ and $\{X_{i_b}^-\}_{b=1}^B$ in bank with update $Z_{i_b} \leftarrow \tilde{Z}_b$ and $X_{i_b}^- \leftarrow \tilde{X}_b^-$.
end for

H Importance of Residual Image for Stability

Throughout our experiments with different versions of the Hat EBM, we find the inclusion of the residual image Y essential for stability. In particular, one could consider an alternate version of the Hat EBM where

$$U(z;\theta) = H(G(z);\theta) \tag{13}$$

without a residual state. As long as the training data is of the form $X^+ = G(Z^+)$ for a latent state Z^+, one could learn the hat network using the same procedure as the Hat EBM without the residual Y. In practice, it is usually not possible to exactly invert the generator. In other words, real images X^+ never lie exactly on the generator output manifold, although they might be close by. Nonetheless, one could bend the rules and use X^+ to train the potential (13) with the justification that there is some Z^+ such that $X^+ \approx G(Z^+)$. In practice, this leads to instability as shown in Figure 5. Even when it is nearly invisible, the residual state Y is still needed for the hat network to balance the energy of positive and negative samples and achieve stable learning.

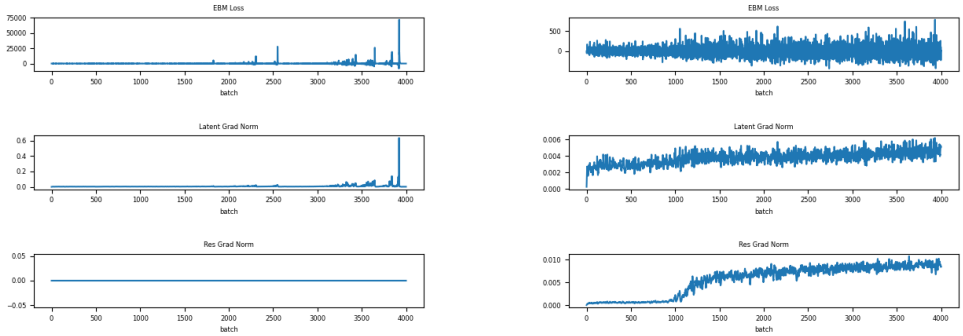


Figure 5: Unstable learning using the energy (13) *(left)* and stable learning using the joint Hat EBM *(right)*. Both settings replicate the refinement experiment using CIFAR-10 with a pretrained SN-GAN generator. Even though the appearance of the residual image Y is nearly invisible, including the residual is essential both from a theoretical perspective and for practical stability.

I Discussion of EBM Synthesis Methods

This appendix provides further discussion of related EBM methods, including methods presented in Table 2 of the main paper, and draws relevant comparisons between the Hat EBM and other EBMs.

One branch of EBM works uses MCMC-based Maximum Likelihood with persistent initialization of MCMC states. Persistent initialization uses samples of prior short run EBM trajectories to initialize the current sampling trajectory. This approach is introduced by Persistent Contrastive Divergence (PCD) [37]. The IGEBM [9] is trained using a bank with 10,000 images to hold persistent states. States are rejuvenated from a Gaussian or uniform noise image with of between 0.5% and 5% probability before being returned to the image bank. The Improved CD EBM [8] builds on these results by including an approximate KL divergence term in EBM learning to minimize the difference between the data distribution and the sampled distribution, and by rejuvenating MCMC trajectories using data augmentation instead of resetting states with noise. The Joint Energy Model (JEM) [13] trains an unconditional EBM and a classifier model simultaneously with the same network using persistent initialization with noise rejuvenation. The use of persistent states in our work differs from prior work because we use persistent states to update only the generator while the EBM is updated by states generated from scratch in the current iteration. This is done to increase the diversity of samples used to update the generator, which is essential for enabling the generator to create distinct appearances for different Z early in training (see Appendix F).

Another branch of EBM works trains a generator network in tandem with the energy network. Most works use the standard EBM update or a close variant to train the energy network, as we do. In some works, the generators produce the final samples and no MCMC is used, while other works use the generator to initialize samples and then refine the samples with MCMC driven by the energy network. Our work adopts the second strategy. To our knowledge, the first work that explores the idea jointly training an energy network and generator network is by Kim & Bengio [19]. This work suggests using the generator samples directly as negative samples without use of MCMC, and updating the generator network to decrease the energy of the generator samples. The EGAN [7] builds on [19] by introducing a entropy maximization term which is needed for a valid Maximum Likelihood objective and which prevents generator collapse. The entropy term is estimated by neighborhood methods and variational methods. MEG [21] and VERA [14] build on [7] by introducing more sophisticated methods of entropy maximization. The GEBM [3] uses an approach similar to [7], with the major differences being use of a generalized log likelihood objective that bridges the gap between the support of the generator output and the full image space distribution of the data, and a novel approximate KL bound for learning the generator. Like the Hat EBM, none of these methods require the log determinant of the generator Jacobian or inference of latent states for data. Unlike the Hat EBM, the probability models from these methods lie in the latent space (or the restricted image space given by the generator outputs) instead of the full image space. The methods are also incompatible with non-probabilistic generators, unlike Hat EBM. None of the works above use MCMC during training, although some use MCMC during synthesis [14, 3]. Cooperative learning [39] uses Maximum Likelihood learning described in Section 3.4. This requires MCMC sampling for both image and latent states. The conditional Hat EBM for synthesis requires sampling for image states but not latent states.

A third branch of EBM methods initialize MCMC sampling from a noise distribution and use a fixed length MCMC trajectory to generate states without a generator network. This branch differs from persistent methods because no persistent bank is used and negative samples to update the EBM are created from scratch each time the EBM weights are updated. It differs from generator methods because realistic synthesis is achieved through pure

MCMC without initial realistic states from the generator. The Multigrid EBM [10] has a MCMC-based training method where images are synthesized and sampled at multiple resolutions. Multiple EBMs are learned in parallel at different resolutions, and generated images from low resolution EBMs are passed to high resolution EBMs to initialize MCMC sampling. Generation can be performed by trivial sampling (uniform, Gaussian Mixture, KDE, etc.) at a single-pixel resolution and passing the generated MCMC states along from the single-pixel EBM to the full-size EBM. The short run initialization method [29] starts sampling from a uniform image distribution and runs 50 to 100 MCMC steps to generate images during each EBM update, bypassing the need for persistent banks. Our retrofit Hat EBM training is a variation of the short run method where both the Y and Z are initialized from uniform noise. Since the generator is non-probabilistic, the short run trajectories of Z must move from uniform latent samples that represent noisy images to tuned latent samples whose generated images match the data appearance.

J Hyperparameters

Synthesis Training

Dataset	Celeb-A	CIFAR-10	ImageNet
Training Steps	50000	75000	300000
Batch Size	128	128	128
Data Epsilon	1e-3	1e-3	1e-3
EBM LR	1e-4	1e-4	1e-4
EBM Optimizer	Adam	Adam	Adam
EBM Gradient Clip	None	None	50
Langevin Epsilon	5e-4	5e-4	5e-4
MCMC Steps	50	50	50
MCMC Temperature	1e-8	1e-3	1e-8
Persistent Bank Size	10000	10000	10000
Generator LR	1e-4	1e-4	5e-5
Generator Optimizer	Adam	Adam	Adam

Retrofit Training

Dataset	CIFAR-10
Training Steps	30000
Batch Size	128
Data Epsilon	1e-3
EBM LR	1e-4
EBM Optimizer	Adam
Image Space Epsilon	5e-4
Latent Epsilon	1e-3
MCMC Steps	100
MCMC Temperature	1e-3
Prior σ	0.1

Refinement Training

Dataset	Celeb-A	CIFAR-10
Training Steps	20000	20000
Batch Size	128	128
Data Epsilon	1e-3	1e-3
EBM LR	1e-5	1e-5
EBM Optimizer	Adam	Adam
Image Space Epsilon	1e-4	1e-4
Latent Epsilon	5e-3	5e-3
MCMC Steps	100	250
MCMC Temperature	1e-6	1e-3
Prior σ	None	0.25

K Visualization of Synthesis Results

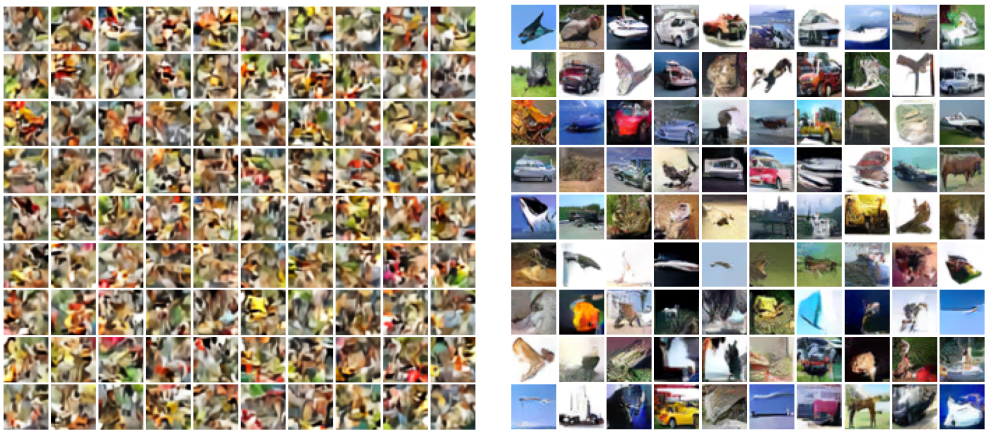


Figure 6: Initial image states *(left)* and sampled image states *(right)* for retrofit Hat EBM that uses a pretrained generator from a deterministic autoencoder. The training dataset is CIFAR-10.

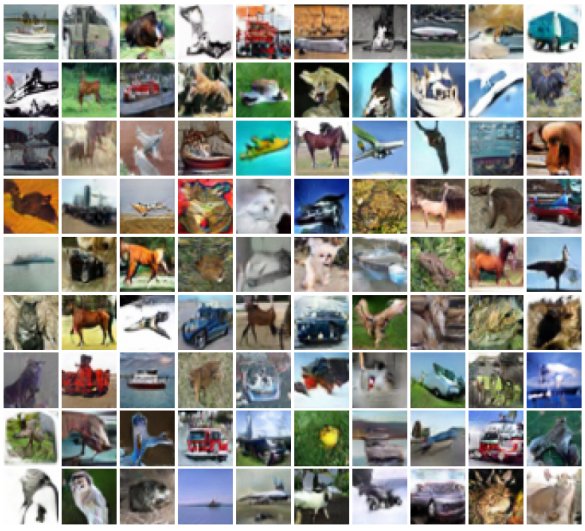


Figure 7: Uncurated Hat EBM samples for unconditional CIFAR-10 at resolution 32×32.

Figure 8: Uncurated Hat EBM samples for Celeb-A at resolution 64×64.

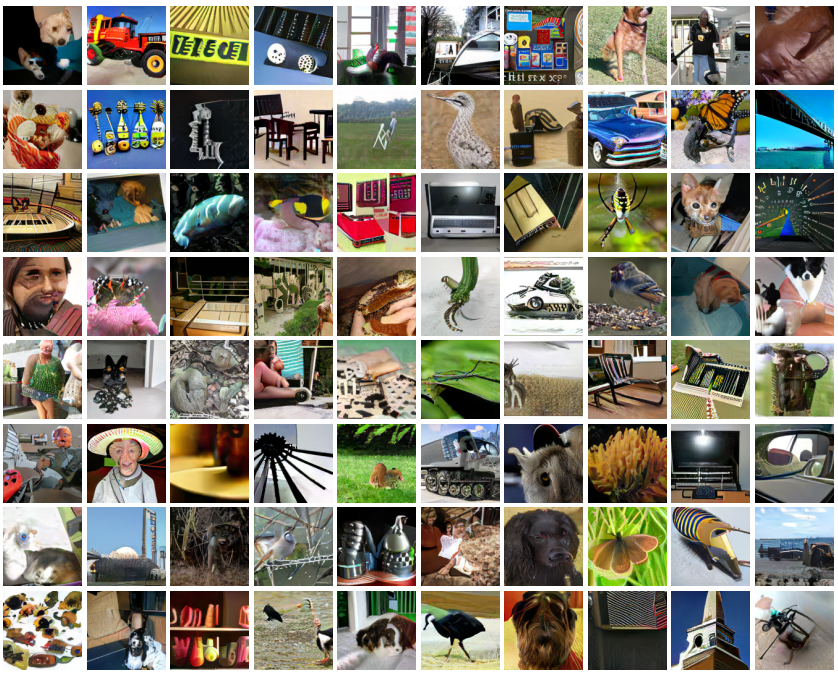


Figure 9: Uncurated Hat EBM samples for unconditional ImageNet at resolution 128×128.