Learning Non-Convergent Non-Persistent Short-Run MCMC Toward Energy-Based Model

Erik Nijkamp
UCLA Department of Statistics
enijkamp@ucla.edu

Mitch Hill
UCLA Department of Statistics
mkhill@ucla.edu

Song-Chun Zhu
UCLA Department of Statistics
sczhu@stat.ucla.edu

Ying Nian Wu
UCLA Department of Statistics
ywu@stat.ucla.edu

Abstract

This paper studies a curious phenomenon in learning energy-based model (EBM) using MCMC. In each learning iteration, we generate synthesized examples by running a non-convergent, non-mixing, and non-persistent short-run MCMC toward the current model, always starting from the same initial distribution such as uniform noise distribution, and always running a fixed number of MCMC steps. After generating synthesized examples, we then update the model parameters according to the maximum likelihood learning gradient, as if the synthesized examples are fair samples from the current model. We treat this non-convergent short-run MCMC as a learned generator model or a flow model. We provide arguments for treating the learned non-convergent short-run MCMC as a valid model. We show that the learned short-run MCMC is capable of generating realistic images. More interestingly, unlike traditional EBM or MCMC, the learned short-run MCMC is capable of reconstructing observed images and interpolating between images, like generator or flow models. The code can be found in the Appendix.

1 Introduction

1.1 Learning Energy-Based Model by MCMC Sampling

The maximum likelihood learning of the energy-based model (EBM) [29, 52, 19, 41, 30, 34, 6, 32, 49, 50, 22, 7, 48] follows what Grenander [15] called "analysis by synthesis" scheme. Within each learning iteration, we generate synthesized examples by sampling from the current model, and then update the model parameters based on the difference between the synthesized examples and the observed examples, so that eventually the synthesized examples match the observed examples in terms of some statistical properties defined by the model. To sample from the current EBM, we need to use Markov chain Monte Carlo (MCMC), such as the Gibbs sampler [12], Langevin dynamics, or Hamiltonian Monte Carlo [33]. Recent work that parametrizes the energy function by modern convolutional neural networks (ConvNets) [28, 26] suggests that the "analysis by synthesis" process can indeed generate highly realistic images [49, 11, 21, 10].

Although the "analysis by synthesis" learning scheme is intuitively appealing, the convergence of MCMC can be impractical, especially if the energy function is multi-modal, which is typically the case if the EBM is to approximate the complex data distribution, such as that of natural images. For such EBM, the MCMC usually does not mix, i.e., MCMC chains from different starting points tend to get trapped in different local modes instead of traversing modes and mixing with each other.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Figure 1: **Synthesis by short-run MCMC**: Generating synthesized examples by running 100 steps of Langevin dynamics initialized from uniform noise for CelebA (64×64).

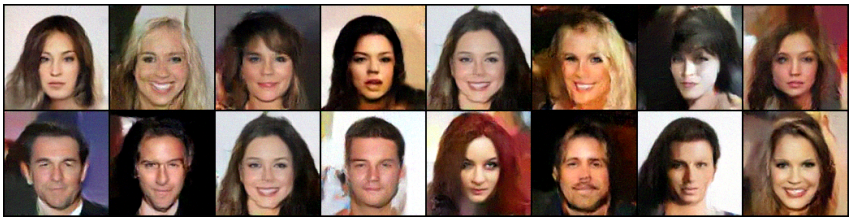

Figure 2: **Synthesis by short-run MCMC**: Generating synthesized examples by running 100 steps of Langevin dynamics initialized from uniform noise for CelebA (128×128).

1.2 Short-Run MCMC as Generator or Flow Model

In this paper, we investigate a learning scheme that is apparently wrong with no hope of learning a valid model. Within each learning iteration, we run a non-convergent, non-mixing and non-persistent short-run MCMC, such as 5 to 100 steps of Langevin dynamics, toward the current EBM. Here, we always initialize the non-persistent short-run MCMC from the same distribution, such as the uniform noise distribution, and we always run the same number of MCMC steps. We then update the model parameters as usual, as if the synthesized examples generated by the non-convergent and non-persistent noise-initialized short-run MCMC are the fair samples generated from the current EBM. We show that, after the convergence of such a learning algorithm, the resulting noise-initialized short-run MCMC can generate realistic images, see Figures 1 and 2.

The short-run MCMC is not a valid sampler of the EBM because it is short-run. As a result, the learned EBM cannot be a valid model because it is learned based on a wrong sampler. Thus we learn a wrong sampler of a wrong model. However, the short-run MCMC can indeed generate realistic images. What is going on?

The goal of this paper is to understand the learned short-run MCMC. We provide arguments that it is a valid model for the data in terms of matching the statistical properties of the data distribution. We also show that the learned short-run MCMC can be used as a generative model, such as a generator model [13, 25] or the flow model [8, 9, 24, 4, 14], with the Langevin dynamics serving as a noise-injected residual network, with the initial image serving as the latent variables, and with the initial uniform noise distribution serving as the prior distribution of the latent variables. We show that unlike traditional EBM and MCMC, the learned short-run MCMC is capable of reconstructing the observed images and interpolating different images, just like a generator or a flow model can do. See Figures 3 and 4. This is very unconventional for EBM or MCMC, and this is due to the fact that the MCMC is non-convergent, non-mixing and non-persistent. In fact, our argument applies to the situation where the short-run MCMC does not need to have the EBM as the stationary distribution.

While the learned short-run MCMC can be used for synthesis, the above learning scheme can be generalized to tasks such as image inpainting, super-resolution, style transfer, or inverse optimal control [53, 1] etc., using informative initial distributions and conditional energy functions.

2 Contributions and Related Work

This paper constitutes a conceptual shift, where we shift attention from learning EBM with unrealistic convergent MCMC to the non-convergent short-run MCMC. This is a break away from the long tradition of both EBM and MCMC. We provide theoretical and empirical evidence that the learned short-run MCMC is a valid generator or flow model. This conceptual shift frees us from the convergence issue of MCMC, and makes the short-run MCMC a reliable and efficient technology.

More generally, we shift the focus from energy-based model to energy-based dynamics. This appears to be consistent with the common practice of computational neuroscience [27], where researchers often directly start from the dynamics, such as attractor dynamics [20, 2, 37] whose express goal is to

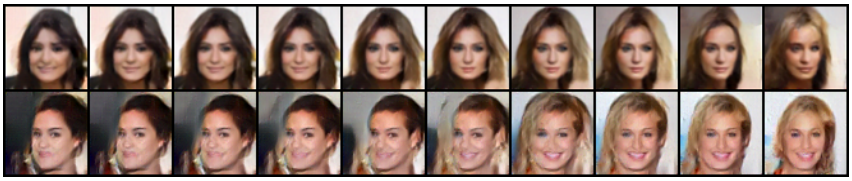

Figure 3: **Interpolation by short-run MCMC resembling a generator or flow model**: The transition depicts the sequence $M_\theta(z_\rho)$ with interpolated noise $z_\rho = \rho z_1 + \sqrt{1-\rho^2} z_2$ where $\rho \in [0,1]$ on CelebA (64×64). Left: $M_\theta(z_1)$. Right: $M_\theta(z_2)$. See Section 3.4.

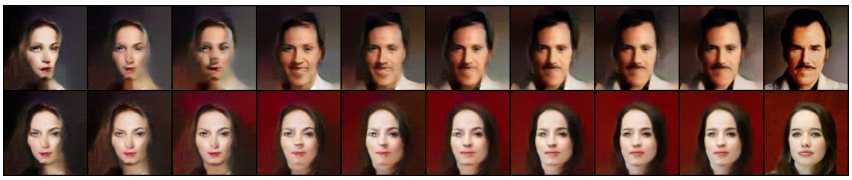

Figure 4: **Reconstruction by short-run MCMC resembling a generator or flow model**: The transition depicts $M_\theta(z_t)$ over time t from random initialization $t = 0$ to reconstruction $t = 200$ on CelebA (64×64). Left: Random initialization. Right: Observed examples. See Section 3.4.

be trapped in a local mode. It is our hope that our work may help to understand the learning of such dynamics. We leave it to future work.

For short-run MCMC, contrastive divergence (CD) [18] is the most prominent framework for theoretical underpinning. The difference between CD and our study is that in our study, the short-run MCMC is initialized from noise, while CD initializes from observed images. CD has been generalized to persistent CD [45]. Compared to persistent MCMC, the non-persistent MCMC in our method is much more efficient and convenient. [35] performs a thorough investigation of various persistent and non-persistent, as well as convergent and non-convergent learning schemes. In particular, the emphasis is on learning proper energy function with persistent and convergent Markov chains. In all of the CD-based frameworks, the goal is to learn the EBM, whereas in our framework, we discard the learned EBM, and only keep the learned short-run MCMC.

Our theoretical understanding of short-run MCMC is based on generalized moment matching estimator. It is related to moment matching GAN [31], however, we do not learn a generator adversarially.

3 Non-Convergent Short-Run MCMC as Generator Model

3.1 Maximum Likelihood Learning of EBM

Let x be the signal, such as an image. The energy-based model (EBM) is a Gibbs distribution

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp(f_\theta(x)), \tag{1}$$

where we assume x is within a bounded range. $f_\theta(x)$ is the negative energy and is parametrized by a bottom-up convolutional neural network (ConvNet) with weights θ. $Z(\theta) = \int \exp(f_\theta(x))dx$ is the normalizing constant.

Suppose we observe training examples $x_i, i = 1, ..., n \sim p_{\text{data}}$, where p_{data} is the data distribution. For large n, the sample average over $\{x_i\}$ approximates the expectation with respect with p_{data}. For notational convenience, we treat the sample average and the expectation as the same.

The log-likelihood is

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x_i) \doteq \mathbb{E}_{p_{\text{data}}}[\log p_\theta(x)]. \tag{2}$$

The derivative of the log-likelihood is

$$L'(\theta) = \mathbb{E}_{p_{\text{data}}}\left[\frac{\partial}{\partial \theta} f_\theta(x)\right] - \mathbb{E}_{p_\theta}\left[\frac{\partial}{\partial \theta} f_\theta(x)\right] \doteq \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial \theta} f_\theta(x_i) - \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial \theta} f_\theta(x_i^-), \tag{3}$$

where $x_i^- \sim p_\theta(x)$ for $i = 1, ..., n$ are the generated examples from the current model $p_\theta(x)$.

The above equation leads to the "analysis by synthesis" learning algorithm. At iteration t, let θ_t be the current model parameters. We generate $x_i^- \sim p_{\theta_t}(x)$ for $i = 1, ..., n$. Then we update $\theta_{t+1} = \theta_t + \eta_t L'(\theta_t)$, where η_t is the learning rate.

3.2 Short-Run MCMC

Generating synthesized examples $x_i^- \sim p_\theta(x)$ requires MCMC, such as Langevin dynamics (or Hamiltonian Monte Carlo) [33], which iterates

$$x_{\tau+\Delta\tau} = x_\tau + \frac{\Delta\tau}{2} f'_\theta(x_\tau) + \sqrt{\Delta\tau} U_\tau, \tag{4}$$

where τ indexes the time, $\Delta\tau$ is the discretization of time, and $U_\tau \sim \mathrm{N}(0, I)$ is the Gaussian noise term. $f'_\theta(x) = \partial f_\theta(x)/\partial x$ can be obtained by back-propagation. If p_θ is of low entropy or low temperature, the gradient term dominates the diffusion noise term, and the Langevin dynamics behaves like gradient descent.

If $f_\theta(x)$ is multi-modal, then different chains tend to get trapped in different local modes, and they do not mix. We propose to give up the sampling of p_θ. Instead, we run a fixed number, e.g., K, steps of MCMC, toward p_θ, starting from a fixed initial distribution, p_0, such as the uniform noise distribution. Let M_θ be the K-step MCMC transition kernel. Define

$$q_\theta(x) = (M_\theta p_0)(z) = \int p_0(z) M_\theta(x|z) dz, \tag{5}$$

which is the marginal distribution of the sample x after running K-step MCMC from p_0.

In this paper, instead of learning p_θ, we treat q_θ to be the target of learning. After learning, we keep q_θ, but we discard p_θ. That is, the sole purpose of p_θ is to guide a K-step MCMC from p_0.

3.3 Learning Short-Run MCMC

The learning algorithm is as follows. Initialize θ_0. At learning iteration t, let θ_t be the model parameters. We generate $x_i^- \sim q_{\theta_t}(x)$ for $i = 1, ..., m$. Then we update $\theta_{t+1} = \theta_t + \eta_t \Delta(\theta_t)$, where

$$\Delta(\theta) = \mathbb{E}_{p_{\text{data}}}\left[\frac{\partial}{\partial\theta} f_\theta(x)\right] - \mathbb{E}_{q_\theta}\left[\frac{\partial}{\partial\theta} f_\theta(x)\right] \approx \sum_{i=1}^m \frac{\partial}{\partial\theta} f_\theta(x_i) - \sum_{i=1}^m \frac{\partial}{\partial\theta} f_\theta(x_i^-). \tag{6}$$

We assume that the algorithm converges so that $\Delta(\theta_t) \to 0$. At convergence, the resulting θ solves the estimating equation $\Delta(\theta) = 0$.

To further improve training, we smooth p_{data} by convolution with a Gaussian white noise distribution, i.e., injecting additive noises $\varepsilon_i \sim \mathrm{N}(0, \sigma^2 I)$ to observed examples $x_i \leftarrow x_i + \varepsilon_i$ [43, 40]. This makes it easy for $\Delta(\theta_t)$ to converge to 0, especially if the number of MCMC steps, K, is small, so that the estimating equation $\Delta(\theta) = 0$ may not have solution without smoothing p_{data}.

The learning procedure in Algorithm 1 is simple. The key to the above algorithm is that the generated $\{x_i^-\}$ are independent and fair samples from the model q_θ.

Algorithm 1: Learning short-run MCMC. See code in Appendix 7.3.

input : Negative energy $f_\theta(x)$, training steps T, initial weights θ_0, observed examples $\{x_i\}_{i=1}^n$, batch size m, variance of noise σ^2, Langevin descretization $\Delta\tau$ and steps K, learning rate η.

output : Weights θ_{T+1}.

for $t = 0 : T$ **do**

1. Draw observed images $\{x_i\}_{i=1}^m$.
2. Draw initial negative examples $\{x_i^-\}_{i=1}^m \sim p_0$.
3. Update observed examples $x_i \leftarrow x_i + \varepsilon_i$ where $\varepsilon_i \sim \mathrm{N}(0, \sigma^2 I)$.
4. Update negative examples $\{x_i^-\}_{i=1}^m$ for K steps of Langevin dynamics (4).
5. Update θ_t by $\theta_{t+1} = \theta_t + g(\Delta(\theta_t), \eta, t)$ where gradient $\Delta(\theta_t)$ is (6) and g is Adam [23].

This is actually quite neat.

3.4 Generator or Flow Model for Interpolation and Reconstruction

We may consider $q_\theta(x)$ to be a generative model,

$$z \sim p_0(z);\ x = M_\theta(z,u), \tag{7}$$

where u denotes all the randomness in the short-run MCMC. For the K-step Langevin dynamics, M_θ can be considered a K-layer noise-injected residual network. z can be considered latent variables, and p_0 the prior distribution of z. Due to the non-convergence and non-mixing, x can be highly dependent on z, and z can be inferred from x. This is different from the convergent MCMC, where x is independent of z. When the learning algorithm converges, the learned EBM tends to have low entropy and the Langevin dynamics behaves like gradient descent, where the noise terms are disabled, i.e., $u = 0$. In that case, we simply write $x = M_\theta(z)$.

We can perform interpolation as follows. Generate z_1 and z_2 from $p_0(z)$. Let $z_\rho = \rho z_1 + \sqrt{1-\rho^2} z_2$. This interpolation keeps the marginal variance of z_ρ fixed. Let $x_\rho = M_\theta(z_\rho)$. Then x_ρ is the interpolation of $x_1 = M_\theta(z_1)$ and $x_2 = M_\theta(z_2)$. Figure 3 displays x_ρ for a sequence of $\rho \in [0,1]$.

For an observed image x, we can reconstruct x by running gradient descent on the least squares loss function $L(z) = \|x - M_\theta(z)\|^2$, initializing from $z_0 \sim p_0(z)$, and iterates $z_{t+1} = z_t - \eta_t L'(z_t)$. Figure 4 displays the sequence of $x_t = M_\theta(z_t)$.

In general, $z \sim p_0(z); x = M_\theta(z,u)$ defines an energy-based dynamics. K does not need to be fixed. It can be a stopping time that depends on the past history of the dynamics. The dynamics can be made deterministic by setting $u = 0$. This includes the attractor dynamics popular in computational neuroscience [20, 2, 37].

4 Understanding the Learned Short-Run MCMC

4.1 Exponential Family and Moment Matching Estimator

An early version of EBM is the FRAME (Filters, Random field, And Maximum Entropy) model [52, 46, 51], which is an exponential family model, where the features are the responses from a bank of filters. The deep FRAME model [32] replaces the linear filters by the pre-trained ConvNet filters. This amounts to only learning the top layer weight parameters of the ConvNet. Specifically, $f_\theta(x) = \langle \theta, h(x) \rangle$, where $h(x)$ are the top-layer filter responses of a pre-trained ConvNet, and θ consists of the top-layer weight parameters. For such an $f_\theta(x)$, $\frac{\partial}{\partial \theta} f_\theta(x) = h(x)$. Then, the maximum likelihood estimator of p_θ is actually a moment matching estimator, i.e., $\mathbb{E}_{p_{\hat\theta_{\text{MLE}}}}[h(x)] = \mathbb{E}_{p_{\text{data}}}[h(x)]$. If we use the short-run MCMC learning algorithm, it will converge (assume convergence is attainable) to a moment matching estimator, i.e., $\mathbb{E}_{q_{\hat\theta_{\text{MME}}}}[h(x)] = \mathbb{E}_{p_{\text{data}}}[h(x)]$. Thus, the learned model $q_{\hat\theta_{\text{MME}}}(x)$ is a valid estimator in that it matches to the data distribution in terms of sufficient statistics defined by the EBM.

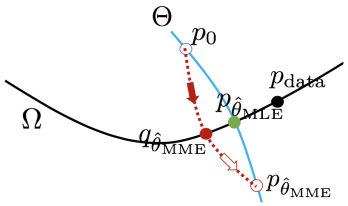

Figure 5: The blue curve illustrates the model distributions corresponding to different values of parameter θ. The black curve illustrates all the distributions that match p_{data} (black dot) in terms of $\mathbb{E}[h(x)]$. The MLE $p_{\hat\theta_{\text{MLE}}}$ (green dot) is the intersection between Θ (blue curve) and Ω (black curve). The MCMC (red dotted line) starts from p_0 (hollow blue dot) and runs toward $p_{\hat\theta_{\text{MME}}}$ (hollow red dot), but the MCMC stops after K-step, reaching $q_{\hat\theta_{\text{MME}}}$ (red dot), which is the learned short-run MCMC.

Consider two families of distributions: $\Omega = \{p : \mathbb{E}_p[h(x)] = \mathbb{E}_{p_{\text{data}}}[h(x)]\}$, and $\Theta = \{p_\theta(x) = \exp(\langle \theta, h(x) \rangle)/Z(\theta), \forall \theta\}$. They are illustrated by two curves in Figure 5. Ω contains all the distributions that match the data distribution in terms of $\mathbb{E}[h(x)]$. Both $p_{\hat\theta_{\text{MLE}}}$ and $q_{\hat\theta_{\text{MME}}}$ belong to Ω, and of course p_{data} also belongs to Ω. Θ contains all the EBMs with different values of the parameter θ. The uniform distribution p_0 corresponds to $\theta = 0$, thus p_0 belongs to Θ.

Difficult to follow.

The EBM under $\hat{\theta}_{\text{MME}}$, i.e., $p_{\hat{\theta}_{\text{MME}}}$ does not belong to Ω, and it may be quite far from $p_{\hat{\theta}_{\text{MLE}}}$. In general, $\mathbb{E}_{p_{\hat{\theta}_{\text{MME}}}}[h(x)] \neq \mathbb{E}_{p_{\text{data}}}[h(x)]$, that is, the corresponding EBM does not match the data distribution as far as $h(x)$ is concerned. It can be much further from the uniform p_0 than $p_{\hat{\theta}_{\text{MLE}}}$ is from p_0, and thus $p_{\hat{\theta}_{\text{MME}}}$ may have a much lower entropy than $p_{\hat{\theta}_{\text{MLE}}}$.

Figure 5 illustrates the above idea. The red dotted line illustrates MCMC. Starting from p_0, K-step MCMC leads to $q_{\hat{\theta}_{\text{MME}}}(x)$. If we continue to run MCMC for infinite steps, we will get to $p_{\hat{\theta}_{\text{MME}}}$. Thus the role of $p_{\hat{\theta}_{\text{MME}}}$ is to serve as an unreachable target to guide the K-step MCMC which stops at the mid-way $q_{\hat{\theta}_{\text{MME}}}$. One can say that the short-run MCMC is a wrong sampler of a wrong model, but it itself is a valid model because it belongs to Ω.

The MLE $p_{\hat{\theta}_{\text{MLE}}}$ is the projection of p_{data} onto Θ. Thus it belongs to Θ. It also belongs to Ω as can be seen from the maximum likelihood estimating equation. Thus it is the intersection of Ω and Θ. Among all the distributions in Ω, $p_{\hat{\theta}_{\text{MLE}}}$ is the closest to p_0. Thus it has the maximum entropy among all the distributions in Ω.

The above duality between maximum likelihood and maximum entropy follows from the following fact. Let $\hat{p} \in \Theta \cap \Omega$ be the intersection between Θ and Ω. Ω and Θ are orthogonal in terms of the Kullback-Leibler divergence. For any $p_\theta \in \Theta$ and for any $p \in \Omega$, we have the Pythagorean property [36]: $\text{KL}(p|p_\theta) = \text{KL}(p|\hat{p}) + \text{KL}(\hat{p}|p_\theta)$. See Appendix 7.1 for a proof. Thus (1) $\text{KL}(p_{\text{data}}|p_\theta) \geq \text{KL}(p_{\text{data}}|\hat{p})$, i.e., $\hat{p}$ is MLE within Θ. (2) $\text{KL}(p|p_0) \geq \text{KL}(\hat{p}|p_0)$, i.e., $\hat{p}$ has maximum entropy within Ω.

We can understand the learned $q_{\hat{\theta}_{\text{MME}}}$ from two Pythagorean results.

(1) Pythagorean for the right triangle formed by q_0, $q_{\hat{\theta}_{\text{MME}}}$, and $p_{\hat{\theta}_{\text{MLE}}}$,

$$\text{KL}(q_{\hat{\theta}_{\text{MME}}}|p_{\hat{\theta}_{\text{MLE}}}) = \text{KL}(q_{\hat{\theta}_{\text{MME}}}|p_0) - \text{KL}(p_{\hat{\theta}_{\text{MLE}}}|p_0) = H(p_{\hat{\theta}_{\text{MLE}}}) - H(q_{\hat{\theta}_{\text{MME}}}), \tag{8}$$

where $H(p) = -\mathbb{E}_p[\log p(x)]$ is the entropy of p. See Appendix 7.1. Thus we want the entropy of $q_{\hat{\theta}_{\text{MME}}}$ to be high in order for it to be a good approximation to $p_{\hat{\theta}_{\text{MLE}}}$. Thus for small K, it is important to let p_0 be the uniform distribution, which has the maximum entropy.

(2) Pythagorean for the right triangle formed by $p_{\hat{\theta}_{\text{MME}}}$, $q_{\hat{\theta}_{\text{MME}}}$, and $p_{\hat{\theta}_{\text{MLE}}}$,

$$\text{KL}(q_{\hat{\theta}_{\text{MME}}}|p_{\hat{\theta}_{\text{MME}}}) = \text{KL}(q_{\hat{\theta}_{\text{MME}}}|p_{\hat{\theta}_{\text{MLE}}}) + \text{KL}(p_{\hat{\theta}_{\text{MLE}}}|p_{\hat{\theta}_{\text{MME}}}). \tag{9}$$

For fixed θ, as K increases, $\text{KL}(q_\theta|p_\theta)$ decreases monotonically [5]. The smaller $\text{KL}(q_{\hat{\theta}_{\text{MME}}}|p_{\hat{\theta}_{\text{MME}}})$ is, the smaller $\text{KL}(q_{\hat{\theta}_{\text{MME}}}|p_{\hat{\theta}_{\text{MLE}}})$ and $\text{KL}(p_{\hat{\theta}_{\text{MLE}}}|p_{\hat{\theta}_{\text{MME}}})$ are. Thus, it is desirable to use large K as long as we can afford the computational cost, to make both $q_{\hat{\theta}_{\text{MME}}}$ and $p_{\hat{\theta}_{\text{MME}}}$ close to $p_{\hat{\theta}_{\text{MLE}}}$.

4.2 General ConvNet-EBM and Generalized Moment Matching Estimator

For a general ConvNet $f_\theta(x)$, the learning algorithm based on short-run MCMC solves the following estimating equation: $\mathbb{E}_{q_\theta}\left[\frac{\partial}{\partial\theta} f_\theta(x)\right] = \mathbb{E}_{p_{\text{data}}}\left[\frac{\partial}{\partial\theta} f_\theta(x)\right]$, whose solution is $\hat{\theta}_{\text{MME}}$, which can be considered a generalized moment matching estimator that in general solves the following estimating equation: $\mathbb{E}_{q_\theta}[h(x,\theta)] = \mathbb{E}_{p_{\text{data}}}[h(x,\theta)]$, where we generalize $h(x)$ in the original moment matching estimator to $h(x,\theta)$ that involves both x and θ. For our learning algorithm, $h(x,\theta) = \frac{\partial}{\partial\theta} f_\theta(x)$. That is, the learned $q_{\hat{\theta}_{\text{MME}}}$ is still a valid estimator in the sense of matching to the data distribution. The above estimating equation can be solved by Robbins-Monro's stochastic approximation [39], as long as we can generate independent fair samples from q_θ.

In classical statistics, we often assume that the model is correct, i.e., p_{data} corresponds to a $q_{\theta_{\text{true}}}$ for some true value θ_{true}. In that case, the generalized moment matching estimator $\hat{\theta}_{\text{MME}}$ follows an asymptotic normal distribution centered at the true value θ_{true}. The variance of $\hat{\theta}_{\text{MME}}$ depends on the choice of $h(x,\theta)$. The variance is minimized by the choice $h(x,\theta) = \frac{\partial}{\partial\theta}\log q_\theta(x)$, which corresponds to the maximum likelihood estimate of q_θ, and which leads to the Cramer-Rao lower bound and Fisher information. See Appendix 7.2 for a brief explanation.

$\frac{\partial}{\partial\theta}\log p_\theta(x) = \frac{\partial}{\partial\theta} f_\theta(x) - \frac{\partial}{\partial\theta}\log Z(\theta)$ is not equal to $\frac{\partial}{\partial\theta}\log q_\theta(x)$. Thus the learning algorithm will not give us the maximum likelihood estimate of q_θ. However, the validity of the learned q_θ does

How strong are these baselines?

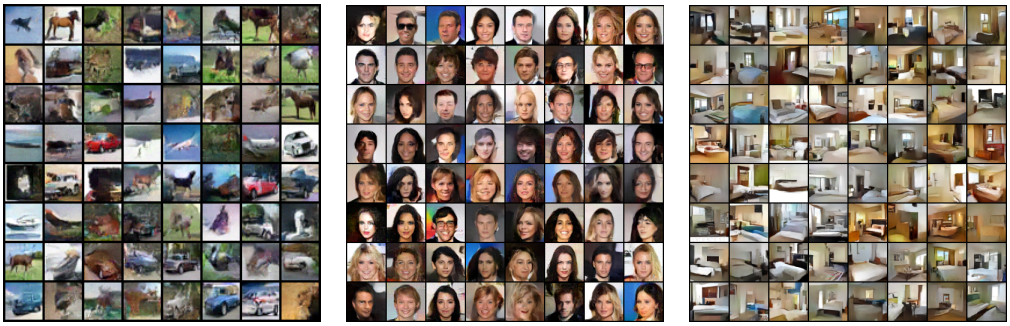

Figure 6: Generated samples for $K = 100$ MCMC steps. From left to right: (1) CIFAR-10 (32×32), (2) CelebA (64×64), (3) LSUN Bedroom (64×64).

Model	CIFAR-10 IS	CelebA FID	LSUN Bedroom FID
VAE [25]	4.28	79.09	183.18
DCGAN [38]	6.16	32.71	54.17
Ours	**6.21**	**23.02**	**44.16**

(a) IS and FID scores for generated examples.

Model	CIFAR-10 MSE	CelebA MSE	LSUN Bedroom MSE
VAE [25]	0.0421	0.0341	0.0440
DCGAN [38]	0.0407	0.0359	0.0592
Ours	**0.0387**	**0.0271**	**0.0272**

(b) Reconstruction error (MSE per pixel).

Table 1: Quality of synthesis and reconstruction for CIFAR-10 (32×32), CelebA (64×64), and LSUN Bedroom (64×64). The number of features n_f is 128, 64, and 64, respectively, and $K = 100$.

not require $h(x, \theta)$ to be $\frac{\partial}{\partial \theta} \log q_\theta(x)$. In practice, one can never assume that the model is true. As a result, the optimality of the maximum likelihood may not hold, and there is no compelling reason that we must use MLE.

The relationship between p_{data}, $q_{\hat{\theta}_{\text{MME}}}$, $p_{\hat{\theta}_{\text{MME}}}$, and $p_{\hat{\theta}_{\text{MLE}}}$ may still be illustrated by Figure 5, although we need to modify the definition of Ω.

5 Experimental Results

In this section, we will demonstrate (1) realistic synthesis, (2) smooth interpolation, (3) faithful reconstruction of observed examples, and, (4) the influence of hyperparameters. K denotes the number of MCMC steps in equation (4). n_f denotes the number of output features maps in the first layer of f_θ. See Appendix for additional results.

We emphasize the simplicity of the algorithm and models, see Appendix 7.3 and 7.4, respectively.

5.1 Fidelity

We evaluate the fidelity of generated examples on various datasets, each reduced to $40,000$ observed examples. Figure 6 depicts generated samples for various datasets with $K = 100$ Langevin steps for both training and evaluation. For CIFAR-10 we set the number of features $n_f = 128$, whereas for CelebA and LSUN we use $n_f = 64$. We use $200,000$ iterations of model updates, then gradually decrease the learning rate η and injected noise $\varepsilon_i \sim \text{N}(0, \sigma^2 I)$ for observed examples. Table 1 (a) compares the Inception Score (IS) [42, 3] and Fréchet Inception Distance (FID) [17] with Inception v3 classifier [44] on $40,000$ generated examples. Despite its simplicity, short-run MCMC is competitive.

5.2 Interpolation

We demonstrate interpolation between generated examples. We follow the procedure outlined in Section 3.4. Let $x_\rho = M_\theta(z_\rho)$ where M_θ to denotes the K-step gradient descent with $K = 100$. Figure 3 illustrates x_ρ for a sequence of $\rho \in [0, 1]$ on CelebA. The interpolation appears smooth and the intermediate samples resemble realistic faces. The interpolation experiment highlights that the short-run MCMC does not mix, which is in fact an advantage instead of a disadvantage. The interpolation ability goes far beyond the capacity of EBM and convergent MCMC.

	K					
	5	10	25	50	75	100
σ	0.15	0.1	0.05	0.04	0.03	**0.03**
FID	213.08	182.5	92.13	68.28	65.37	**63.81**
IS	2.06	2.27	4.06	4.82	4.88	**4.92**
$\|\frac{\partial}{\partial x} f_\theta(x)\|_2$	7.78	3.85	1.76	0.97	0.65	**0.49**

Table 2: Influence of number of MCMC steps K on models with $n_f = 32$ for CIFAR-10 (32×32).

	σ					
	0.10	0.08	0.06	0.05	0.04	0.03
FID	132.51	117.36	94.72	83.15	65.71	**63.81**
IS	4.05	4.20	4.63	4.78	4.83	**4.92**

(a) Influence of additive noise $\varepsilon_i \sim \mathrm{N}(0, \sigma^2 I)$.

	n_f		
	32	64	128
FID	63.81	46.61	**44.50**
IS	4.92	5.49	**6.21**

(b) Influence of model complexity n_f.

Table 3: Influence of noise and model complexity with $K = 100$ for CIFAR-10 (32×32).

5.3 Reconstruction

We demonstrate reconstruction of observed examples. For short-run MCMC, we follow the procedure outlined in Section 3.4. For an observed image x, we reconstruct x by running gradient descent on the least squares loss function $L(z) = \|x - M_\theta(z)\|^2$, initializing from $z_0 \sim p_0(z)$, and iterates $z_{t+1} = z_t - \eta_t L'(z_t)$. For VAE, reconstruction is readily available. For GAN, we perform Langevin inference of latent variables [16, 47]. Figure 4 depicts faithful reconstruction. Table 1 (b) illustrates competitive reconstructions in terms of MSE (per pixel) for $1,000$ observed leave-out examples. Again, the reconstruction ability of the short-run MCMC is due to the fact that it is not mixing.

5.4 Influence of Hyperparameters

MCMC Steps. Table 2 depicts the influence of varying the number of MCMC steps K while training on synthesis and average magnitude $\|\frac{\partial}{\partial x} f_\theta(x)\|_2$ over K-step Langevin (4). We observe: (1) the quality of synthesis decreases with decreasing K, and, (2) the shorter the MCMC, the colder the learned EBM, and the more dominant the gradient descent part of the Langevin. With small K, short-run MCMC fails "gracefully" in terms of synthesis. A choice of $K = 100$ appears reasonable.

Injected Noise. To stabilize training, we smooth p_{data} by injecting additive noises $\varepsilon_i \sim \mathrm{N}(0, \sigma^2 I)$ to observed examples $x_i \leftarrow x_i + \varepsilon_i$. Table 3 (a) depicts the influence of σ^2 on the fidelity of negative examples in terms of IS and FID. That is, when lowering σ^2, the fidelity of the examples improves. Hence, it is desirable to pick smallest σ^2 while maintaining the stability of training. Further, to improve synthesis, we may gradually decrease the learning rate η and anneal σ^2 while training.

Model Complexity. We investigate the influence of the number of output features maps n_f on generated samples with $K = 100$. Table 3 (b) summarizes the quality of synthesis in terms of IS and FID. As the number of features n_f increases, so does the quality of the synthesis. Hence, the quality of synthesis may scale with n_f until the computational means are exhausted.

6 Conclusion

Despite our focus on short-run MCMC, we do not advocate abandoning EBM all together. On the contrary, we ultimately aim to learn valid EBM [35]. Hopefully, the non-convergent short-run MCMC studied in this paper may be useful in this endeavor. It is also our hope that our work may help to understand the learning of attractor dynamics popular in neuroscience.

Acknowledgments

The work is supported by DARPA XAI project N66001-17-2-4029; ARO project W911NF1810296; and ONR MURI project N00014-16-1-2007; and XSEDE grant ASC170063. We thank Prof. Stu Geman, Prof. Xianfeng (David) Gu, Diederik P. Kingma, Guodong Zhang, and Will Grathwohl for helpful discussions.

References

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, 2004.

[2] Daniel J. Amit. *Modeling brain function: the world of attractor neural networks, 1st Edition*. Cambridge Univ. Press, 1989.

[3] Shane T. Barratt and Rishi Sharma. A note on the inception score. *CoRR*, abs/1801.01973, 2018.

[4] Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 573–582, 2019.

[5] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006.

[6] Jifeng Dai, Yang Lu, and Ying Nian Wu. Generative modeling of convolutional neural networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[7] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard H. Hovy, and Aaron C. Courville. Calibrating energy-based generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[8] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[10] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *CoRR*, abs/1903.08689, 2019.

[11] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning generative convnets via multi-grid modeling and sampling. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9155–9164, 2018.

[12] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741, 1984.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[14] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[15] Ulf Grenander and Michael I Miller. *Pattern theory: from representation to inference*. Oxford University Press, 2007.

[16] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 1976–1984, 2017.

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6626–6637, 2017.

[18] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[19] Geoffrey E. Hinton, Simon Osindero, Max Welling, and Yee Whye Teh. Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive Science*, 30(4):725–731, 2006.

[20] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the national academy of sciences*, volume 79, pages 2554–2558. National Acad Sciences, 1982.

[21] Long Jin, Justin Lazarow, and Zhuowen Tu. Introspective learning for discriminative classification. In *Advances in Neural Information Processing Systems*, 2017.

[22] Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *CoRR*, abs/1606.03439, 2016.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[24] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 10236–10245, 2018.

[25] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012.

[27] Dmitry Krotov and John J. Hopfield. Unsupervised learning by competing hidden units. *Proc. Natl. Acad. Sci. U.S.A.*, 116(16):7723–7731, 2019.

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[30] Honglak Lee, Roger B. Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pages 609–616, 2009.

[31] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2203–2213, 2017.

[32] Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Learning FRAME models using CNN filters. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1902–1910, 2016.

[33] Radford M Neal. MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

[34] Jiquan Ngiam, Zhenghao Chen, Pang Wei Koh, and Andrew Y. Ng. Learning deep energy models. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1105–1112, 2011.

[35] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of MCMC-based maximum likelihood learning of energy-based models. *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[36] Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. Inducing features of random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4):380–393, 1997.

[37] Bruno Poucet and Etienne Save. Attractors in memory. *Science*, 308(5723):799–800, 2005.

[38] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[39] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[40] Kevin Roth, Aurélien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2018–2028, 2017.

[41] Ruslan Salakhutdinov and Geoffrey E. Hinton. Deep boltzmann machines. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, pages 448–455, 2009.

[42] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234, 2016.

[43] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised MAP inference for image super-resolution. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[44] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826, 2016.

[45] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 1064–1071, 2008.

[46] Ying Nian Wu, Song Chun Zhu, and Xiuwen Liu. Equivalence of julesz ensembles and FRAME models. *International Journal of Computer Vision*, 38(3):247–265, 2000.

[47] Jianwen Xie, Yang Lu, Ruiqi Gao, and Ying Nian Wu. Cooperative learning of energy-based model and latent variable model via MCMC teaching. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4292–4301, 2018.

[48] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of descriptor and generator networks. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 2018.

[49] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative convnet. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2635–2644, 2016.

[50] Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. Energy-based generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[51] Song Chun Zhu and David Mumford. Grade: Gibbs reaction and diffusion equations. In *Computer Vision, 1998. Sixth International Conference on*, pages 847–854, 1998.

[52] Song Chun Zhu, Ying Nian Wu, and David Mumford. Filters, random fields and maximum entropy (FRAME): towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.

[53] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1433–1438, 2008.

7 Appendix

7.1 Proof of Pythagorean Identity

For $p \in \Omega$, let $\mathbb{E}_p[h(x)] = \mathbb{E}_{p_{\text{data}}}[h(x)] = \hat{h}$.

$$\begin{aligned} \text{KL}(p|p_\theta) &= \mathbb{E}_p[\log p(x) - \langle \theta, h(x) \rangle + \log Z(\theta)] && (10) \\ &= -H(p) - \langle \theta, \hat{h} \rangle + \log Z(\theta), && (11) \end{aligned}$$

where $H(p) = -\mathbb{E}_p[\log p(x)]$ is the entropy of p.

For $\hat{p} \in \Omega \cap \Theta$,

$$\text{KL}(\hat{p}|p_\theta) = -H(\hat{p}) - \langle \theta, \hat{h} \rangle + \log Z(\theta). \quad (12)$$

$$\begin{aligned} \text{KL}(p|\hat{p}) &= \mathbb{E}_p[\log p(x)] - \mathbb{E}_p[\log \hat{p}(x)] && (13) \\ &= \mathbb{E}_p[\log p(x)] - \mathbb{E}_{\hat{p}}[\log \hat{p}(x)] && (14) \\ &= -H(p) + H(\hat{p}). && (15) \end{aligned}$$

Thus $\text{KL}(p|p_\theta) = \text{KL}(p|\hat{p}) + \text{KL}(\hat{p}|p_\theta)$.

7.2 Estimating Equation and Cramer-Rao Theory

For a model q_θ, we can estimate θ by solving the estimating equation $\mathbb{E}_{q_\theta}[h(x,\theta)] = \frac{1}{n}\sum_{i=1}^n h(x_i,\theta)$. Assume the solution exists and let it be $\hat{\theta}$. Assume there exists θ_{true} so that $p_{\text{data}} = q_{\theta_{\text{true}}}$. Let $c(\theta) = \mathbb{E}_{q_\theta}[h(x,\theta)]$. We can change $h(x,\theta) \leftarrow h(x,\theta) - c(\theta)$. Then $\mathbb{E}_{q_\theta}[h(x,\theta)] = 0 \; \forall \theta$, and the estimating equation becomes $\frac{1}{n}\sum_{i=1}^n h(x_i,\theta) = 0$. A Taylor expansion around θ_{true} gives us the asymptotic linear equation $\frac{1}{n}\sum_{i=1}^n [h(x_i,\theta_{\text{true}}) + h'(x_i,\theta_{\text{true}})(\theta - \theta_{\text{true}})] = 0$, where $h'(x,\theta) = \frac{\partial}{\partial\theta} h(x,\theta)$. Thus the estimate $\hat{\theta} = \theta_{\text{true}} - \left[\frac{1}{n}\sum_{i=1}^n h'(x_i,\theta_{\text{true}})\right]^{-1} \left[\frac{1}{n}\sum_{i=1}^n h(x_i,\theta_{\text{true}})\right]$, i.e., one-step Newton-Raphson update from θ_{true}. Since $\mathbb{E}_{q_\theta}[h(x,\theta)] = 0$ for any θ, including θ_{true}, the estimator $\hat{\theta}$ is asymptotically unbiased. The Cramer-Rao theory establishes that $\hat{\theta}$ has an asymptotic normal distribution, $\sqrt{n}(\hat{\theta} - \theta_{\text{true}}) \sim \text{N}(0, V)$, where $V = \mathbb{E}_{\theta_{\text{true}}}[h'(x,\theta_{\text{true}}]^{-1} \text{Var}_{\theta_{\text{true}}}[h(x;\theta_{\text{true}})] \mathbb{E}_{\theta_{\text{true}}}[h'(x,\theta_{\text{true}}]^{-1}$. V is minimized if we take $h(x,\theta) = \frac{\partial}{\partial\theta} \log q_\theta(x)$, which leads to the maximum likelihood estimating equation, and the corresponding $V = I(\theta_{\text{true}})^{-1}$, where I is the Fisher information.

7.3 Code

Note, in the code we parametrize the energy as $-f_\theta(x)/T$, for an a priori chosen small temperature T, for convenience, so that the Langevin dynamics becomes $x_{t+1} = x_t + f'_\theta(x_t)/2 + N(0,T)$.

```
import torch as t, torch.nn as nn
import torchvision as tv, torchvision.transforms as tr

seed = 1
im_sz = 32
sigma = 3e-2 # decrease until training is unstable
n_ch = 3
m = 8**2
K = 100
n_f = 64 # increase until compute is exhausted
n_i = 10**5

t.manual_seed(seed)
if t.cuda.is_available():
    t.cuda.manual_seed_all(seed)

device = t.device('cuda' if t.cuda.is_available() else 'cpu')

class F(nn.Module):
    def __init__(self, n_c=n_ch, n_f=n_f, l=0.2):
        super(F, self).__init__()
        self.f = nn.Sequential(
            nn.Conv2d(n_c, n_f, 3, 1, 1),
            nn.LeakyReLU(l),
            nn.Conv2d(n_f, n_f * 2, 4, 2, 1),
            nn.LeakyReLU(l),
            nn.Conv2d(n_f * 2, n_f * 4, 4, 2, 1),
            nn.LeakyReLU(l),
            nn.Conv2d(n_f * 4, n_f * 8, 4, 2, 1),
            nn.LeakyReLU(l),
            nn.Conv2d(n_f * 8, 1, 4, 1, 0))

    def forward(self, x):
        return self.f(x).squeeze()

f = F().to(device)

transform = tr.Compose([tr.Resize(im_sz), tr.ToTensor(), tr.Normalize((.5, .5, .5), (.5, .5, .5))])
p_d = t.stack([x[0] for x in tv.datasets.CIFAR10(root='data/cifar10', transform=transform)]).to(device)
noise = lambda x: x + sigma * t.randn_like(x)
def sample_p_d():
    p_d_i = t.LongTensor(m).random_(0, p_d.shape[0])
    return noise(p_d[p_d_i]).detach()

sample_p_0 = lambda: t.FloatTensor(m, n_ch, im_sz, im_sz).uniform_(-1, 1).to(device)
def sample_q(K=K):
    x_k = t.autograd.Variable(sample_p_0(), requires_grad=True)
    for k in range(K):
        f_prime = t.autograd.grad(f(x_k).sum(), [x_k], retain_graph=True)[0]
        x_k.data += f_prime + 1e-2 * t.randn_like(x_k)
    return x_k.detach()

sqrt = lambda x: int(t.sqrt(t.Tensor([x])))
plot = lambda p, x: tv.utils.save_image(t.clamp(x, -1., 1.), p, normalize=True, nrow=sqrt(m))

optim = t.optim.Adam(f.parameters(), lr=1e-4, betas=[.9, .999])

for i in range(n_i):
    x_p_d, x_q = sample_p_d(), sample_q()
    L = f(x_p_d).mean() - f(x_q).mean()
    optim.zero_grad()
    (-L).backward()
    optim.step()

    if i % 100 == 0:
        print('{:>6d} f(x_p_d)={:>14.9f} f(x_q)={:>14.9f}'.format(i, f(x_p_d).mean(), f(x_q).mean()))
        plot('x_q_{:>06d}.png'.format(i), x_q)
```

7.4 Model Architecture

We use the following notation. Convolutional operation $conv(n)$ with n output feature maps and bias term. Leaky-ReLU nonlinearity *LReLU* with default leaky factor 0.2. We set $n_f \in \{32, 64, 128\}$.

Energy-based Model $(32 \times 32 \times 3)$		
Layers	In-Out Size	Stride
Input	$32 \times 32 \times 3$	
3×3 conv(n_f), LReLU	$32 \times 32 \times n_f$	1
4×4 conv($2 * n_f$), LReLU	$16 \times 16 \times (2 * n_f)$	2
4×4 conv($4 * n_f$), LReLU	$8 \times 8 \times (4 * n_f)$	2
4×4 conv($8 * n_f$), LReLU	$4 \times 4 \times (8 * n_f)$	2
4×4 conv(1)	$1 \times 1 \times 1$	1

Table 4: Network structures $(32 \times 32 \times 3)$.

Energy-based Model $(64 \times 64 \times 3)$		
Layers	In-Out Size	Stride
Input	$64 \times 64 \times 3$	
3×3 conv(n_f), LReLU	$64 \times 64 \times n_f$	1
4×4 conv($2 * n_f$), LReLU	$32 \times 32 \times (2 * n_f)$	2
4×4 conv($4 * n_f$), LReLU	$16 \times 16 \times (4 * n_f)$	2
4×4 conv($8 * n_f$), LReLU	$8 \times 8 \times (8 * n_f)$	2
4×4 conv($8 * n_f$), LReLU	$4 \times 4 \times (8 * n_f)$	2
4×4 conv(1)	$1 \times 1 \times 1$	1

Table 5: Network structures $(64 \times 64 \times 3)$.

Energy-based Model $(128 \times 128 \times 3)$		
Layers	In-Out Size	Stride
Input	$128 \times 128 \times 3$	
3×3 conv(n_f), LReLU	$128 \times 128 \times n_f$	1
4×4 conv($2 * n_f$), LReLU	$64 \times 64 \times (2 * n_f)$	2
4×4 conv($4 * n_f$), LReLU	$32 \times 32 \times (4 * n_f)$	2
4×4 conv($8 * n_f$), LReLU	$16 \times 16 \times (8 * n_f)$	2
4×4 conv($8 * n_f$), LReLU	$8 \times 8 \times (8 * n_f)$	2
4×4 conv($8 * n_f$), LReLU	$4 \times 4 \times (8 * n_f)$	2
4×4 conv(1)	$1 \times 1 \times 1$	1

Table 6: Network structures $(128 \times 128 \times 3)$.

7.5 Computational Cost

Each iteration of short-run MCMC requires computing K derivatives of $f_\theta(x)$ which is in the form of a convolutional neural network. As an example, $100,000$ model parameter updates for 64×64 CelebA with $K = 100$ and $n_f = 64$ on 4 Titan Xp GPUs take 16 hours.

7.6 Varying K for Training and Sampling

We may interpret short-run MCMC as a noise-injected residual network of variable depth. Hence, the number of MCMC steps while training K_1 and sampling K_2 may differ. We train the model on CelebA with K_1 and test the trained model by running MCMC with K_2 steps. Figure 7 below depicts training with $K_1 = 100$ and varied K_2 for sampling. Note, over-saturation occurs for $K_2 > K_1$.

Figure 7: Transition with $K_1 = 100$ for training and varying K_2 for sampling.

7.7 Toy Examples in 1D and 2D

In Figures 8 and 9, we plot the density and log-density of the true model, the learned EBM, and the kernel density estimate (KDE) of the MCMC samples. The density of the MCMC samples matches the true density closely. The learned energy captures the modes of the true density, but is of a much bigger scale, so that the learned EBM density is of much lower entropy or temperature (so that the density focuses on the global energy minimum). This is consistent with our theoretical understanding.

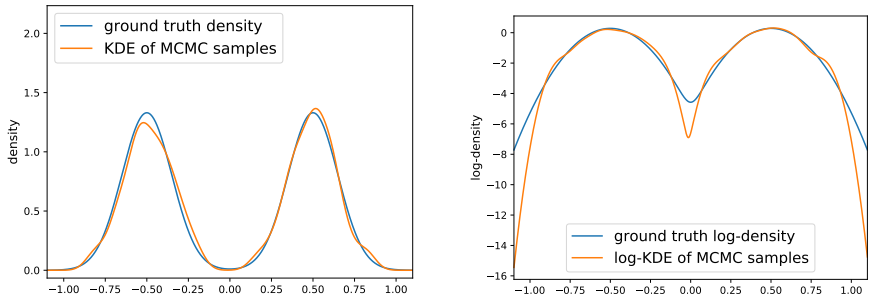

Figure 8: Ground-truth density and KDE of MCMC samples in 1D.

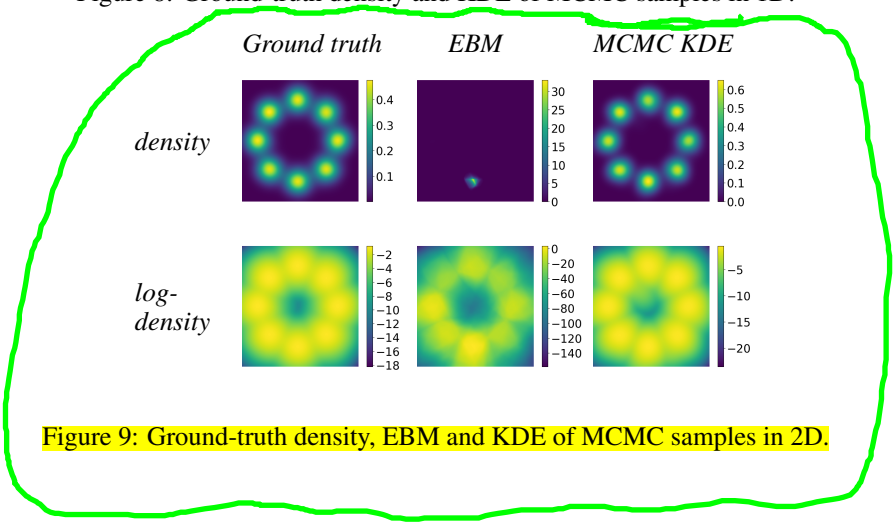

Figure 9: Ground-truth density, EBM and KDE of MCMC samples in 2D.