
Learning Energy-Based Prior Model with Diffusion-Amortized MCMC

Peiyu Yu¹
yupeiyu98@g.ucla.edu

Yaxuan Zhu¹
yaxuanzhu@g.ucla.edu

Sirui Xie²
srxie@ucla.edu

Xiaojian Ma^{2,4}
xiaojian.ma@ucla.edu

Ruiqi Gao³
ruiqig@google.com

Song-Chun Zhu⁴
sczhu@stat.ucla.edu

Ying Nian Wu¹
ywu@stat.ucla.edu

¹UCLA Department of Statistics ²UCLA Department of Computer Science
³Google DeepMind ⁴Beijing Institute for General Artificial Intelligence (BIGAI)

Abstract

Latent space Energy-Based Models (EBMs), also known as energy-based priors, have drawn growing interests in the field of generative modeling due to its flexibility in the formulation and strong modeling power of the latent space. However, the common practice of learning latent space EBMs with non-convergent short-run MCMC for prior and posterior sampling is hindering the model from further progress; the degenerate MCMC sampling quality in practice often leads to degraded generation quality and instability in training, especially with highly multimodal and/or high-dimensional target distributions. To remedy this sampling issue, in this paper we introduce a simple but effective diffusion-based amortization method for long-run MCMC sampling and develop a novel learning algorithm for the latent space EBM based on it. We provide theoretical evidence that the learned amortization of MCMC is a valid long-run MCMC sampler. Experiments on several image modeling benchmark datasets demonstrate the superior performance of our method compared with strong counterparts¹.

1 Introduction

Generative modeling of data distributions has achieved impressive progress with the fast development of deep generative models in recent years [1–9]. It provides a powerful framework that allows successful applications in synthesizing data of different modalities [10–15], extracting semantically meaningful data representation [16–18] as well as other important domains of unsupervised or semi-supervised learning [19–21]. A fundamental and powerful branch of generative modeling is the Deep Latent Variable Model (DLVM). Typically, DLVM assumes that the observation (*e.g.*, a piece of text or images) is generated by its corresponding low-dimensional latent variables via a top-down generator network [1–3]. The latent variables are often assumed to follow a non-informative prior distribution, such as a uniform or isotropic Gaussian distribution. While one can directly learn a deep top-down generator network to faithfully map the non-informative prior distribution to the data distribution, learning an informative prior model in the latent space could further improve the ex-

¹Code and data available at <https://github.com/yuPeiyu98/Diffusion-Amortized-MCMC>.

pressive power of the DLVM with significantly less parameters [22]. In this paper, we specifically consider learning an EBM in the latent space as an informative prior for the model.

Learning energy-based prior can be challenging, as it typically requires computationally expensive Markov Chain Monte Carlo (MCMC) sampling to estimate learning gradients. The difficulty of MCMC-based sampling is non-negligible when the target distribution is highly multi-modal or high-dimensional. In these situations, MCMC sampling can take a long time to converge and perform poorly on traversing modes with limited iterations [23]. Consequently, training models with samples from non-convergent short-run MCMC [24], which is a common choice for learning latent space EBMs [22], often results in malformed energy landscapes [15, 24, 25] and biased estimation of the model parameter. One possible solution is to follow the variational learning scheme [1], which however requires non-trivial extra efforts on model design to deal with problems like posterior collapse [26–28] and limited expressivity induced by model assumptions [1, 29, 30].

To remedy this sampling issue and further unleash the expressive power of the prior model, we propose to shift attention to an economical compromise between unrealistically expensive long-run MCMC and biased short-run MCMC: *we consider learning valid amortization of the potentially long-run MCMC for learning energy-based priors*. Specifically, inspired by the connection between MCMC sampling and denoising diffusion process [7, 8, 31], in this paper we propose a diffusion-based amortization method suitable for long-run MCMC sampling in learning latent space EBMs. The learning algorithm derived from it breaks the long-run chain into consecutive affordable short-run segments that can be iteratively distilled by a diffusion-based sampler. The core idea is simple and can be summarized by a one-liner (Fig. 1). We provide theoretical and empirical evidence that the resulting sampler approximates the long-run chain (see the proof-of-concept toy examples in Appendix E.1), and brings significant performance improvement for learning latent space EBMs on several tasks. We believe that this proposal is a notable attempt to address the learning issues of energy-based priors and is new to the best of our knowledge. We kindly refer to Section 5 for a comprehensive discussion of the related work.

Contributions i) We propose a diffusion-based amortization method for MCMC sampling and develop a novel learning algorithm for the latent space EBM. ii) We provide a theoretical understanding that the learned amortization of MCMC is a valid long-run MCMC sampler. iii) Our experiments demonstrate empirically that the proposed method brings higher sampling quality, a better-learned model and stronger performance on several image modeling benchmark datasets.

2 Background

2.1 Energy-Based Prior Model

We assume that for the observed sample $\mathbf{x} \in \mathbb{R}^D$, there exists $\mathbf{z} \in \mathbb{R}^d$ as its unobserved latent variable vector. The complete-data distribution is

$$p_{\theta}(\mathbf{z}, \mathbf{x}) := p_{\alpha}(\mathbf{z})p_{\beta}(\mathbf{x}|\mathbf{z}), \quad p_{\alpha}(\mathbf{z}) := \frac{1}{Z_{\alpha}} \exp(f_{\alpha}(\mathbf{z})) p_0(\mathbf{z}), \quad (1)$$

where $p_{\alpha}(\mathbf{z})$ is the prior model with parameters α , $p_{\beta}(\mathbf{x}|\mathbf{z})$ is the top-down generation model with parameters β , and $\theta = (\alpha, \beta)$. The prior model $p_{\alpha}(\mathbf{z})$ can be formulated as an energy-based model, which we refer to as the Latent-space Energy-Based Model (LEBM) [22] throughout the paper. In this formulation, $f_{\alpha}(\mathbf{z})$ is parameterized by a neural network with scalar output, Z_{α} is the partition function, and $p_0(\mathbf{z})$ is standard normal as a reference distribution. The prior model in Eq. (1) can be interpreted as an energy-based correction or exponential tilting of the original prior distribution p_0 . The generation model follows $p_{\beta}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(g_{\beta}(\mathbf{z}), \sigma^2 \mathbf{I}_D)$, where g_{β} is the generator network and σ^2 takes a pre-specified value as in VAE [1]. This is equivalent to using l_2 error for reconstruction.

The parameters of LEBM and the generation model can be learned by Maximum Likelihood Estimation (MLE) [22]. To be specific, given the training data \mathbf{x} , the gradients for updating α, β are,

$$\delta_{\alpha}(\mathbf{x}) := \mathbb{E}_{p_{\theta}(\mathbf{z}|\mathbf{x})} [\nabla_{\alpha} f_{\alpha}(\mathbf{z})] - \mathbb{E}_{p_{\alpha}(\mathbf{z})} [\nabla_{\alpha} f_{\alpha}(\mathbf{z})], \quad \delta_{\beta}(\mathbf{x}) := \mathbb{E}_{p_{\theta}(\mathbf{z}|\mathbf{x})} [\nabla_{\beta} \log p_{\beta}(\mathbf{x}|\mathbf{z})]. \quad (2)$$

In practice, one may use the Monte-Carlo average to estimate the expectations in Eq. (2). This involves sampling from the prior $p_{\alpha}(\mathbf{z})$ and the posterior $p_{\theta}(\mathbf{z}|\mathbf{x})$ distribution using MCMC, specifically Langevin Dynamics (LD) [32], to estimate the expectations and hence the gradient. For a target

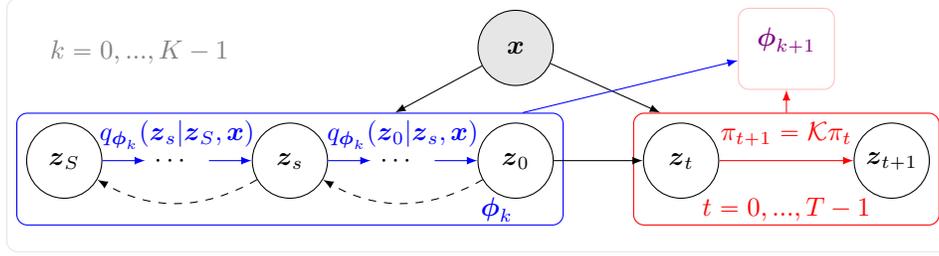


Figure 1: **Learning the DAMC sampler.** The training samples for updating the sampler to ϕ_{k+1} is obtained by T -step short-run LD, initialized with the samples from the current learned sampler ϕ_k . Best viewed in color.

distribution $\pi(z)$, the dynamics iterates

$$z_{t+1} = z_t + \frac{s^2}{2} \nabla_{z_t} \log \pi(z_t) + s w_t, \quad t = 0, 1, \dots, T-1, \quad w_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \quad (3)$$

where s is a small step size. One can draw $z_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ to initialize the chain. For sufficiently small step size s , the distribution of z_t will converge to π as $t \rightarrow \infty$ [32]. However, it is prohibitively expensive to run LD until convergence in most cases, for which we may resort to limited iterations of LD for sampling in practice. This non-convergent short chain yields a moment-matching distribution close to the true $\pi(z)$ but is often biased, which was dubbed as short-run LD [15, 23–25].

2.2 Denoising Diffusion Probabilistic Model

Closely related to EBMs are the Denoising Diffusion Probabilistic Models (DDPMs) [5, 7, 8]. As pointed out in [5, 8], the sampling procedure of DDPM with ϵ -prediction parametrization resembles LD; ϵ (predicted noise) plays a similar role to the gradient of the log density [8].

In the formulation proposed by Kingma et al. [33], the DDPM parameterized by ϕ is specified by a noise schedule built upon $\lambda_s = \log[\beta_s^2/\sigma_s^2]$, *i.e.*, the log signal-to-noise-ratio, that decreases monotonically with s . β_s and σ_s^2 are strictly positive scalar-valued functions of s . We use z_0 to denote training data in \mathbb{R}^d . The forward-time diffusion process $q(z|z_0)$ is defined as:

$$q(z_s|z_0) = \mathcal{N}(z_s; \beta_s z_0, \sigma_s^2 \mathbf{I}_d), \quad q(z'_s|z_s) = \mathcal{N}(z'_s; (\beta_{s'}/\beta_s) z_s, \sigma_{s'|s}^2 \mathbf{I}_d), \quad (4)$$

where $0 \leq s < s' \leq S$ and $\sigma_{s'|s}^2 = (1 - e^{\lambda_{s'} - \lambda_s}) \sigma_s^2$. Noticing that the forward process can be reverted as $q(z_s|z_{s'}, z_0) = \mathcal{N}(z_s; \tilde{\mu}_{s|s'}(z_{s'}, z_0), \tilde{\sigma}_{s|s'}^2 \mathbf{I}_d)$, an ancestral sampler $q_\phi(z_s|z_{s'})$ [8] that starts at $z_S \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ can be derived accordingly [33]:

$$\begin{aligned} \tilde{\mu}_{s|s'}(z_{s'}, z_0) &= e^{\lambda_{s'} - \lambda_s} (\alpha_s/\alpha_{s'}) z_{s'} + (1 - e^{\lambda_{s'} - \lambda_s}) \alpha_s z_0, \quad \tilde{\sigma}_{s|s'}^2 = (1 - e^{\lambda_{s'} - \lambda_s}) \sigma_s^2, \\ z_s &= \tilde{\mu}_{s|s'}(z_{s'}, \hat{z}_0) + \sqrt{(\tilde{\sigma}_{s|s'}^2)^{1-\gamma} (\sigma_{s|s'}^2)^\gamma} \epsilon, \end{aligned} \quad (5)$$

where ϵ is standard Gaussian noise, \hat{z}_0 is the prediction of z_0 by the DDPM ϕ , and γ is a hyperparameter that controls the noise magnitude, following [34]. The goal of DDPM is to recover the distribution of z_0 from the given Gaussian distribution. It can be trained by optimizing $\mathbb{E}_{\epsilon, \lambda} [\|\epsilon(z_\lambda) - \epsilon\|_2^2]$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and λ is drawn from a distribution of log noise-to-signal ratio $p(\lambda)$ over uniformly sampled times $s \in [0, S]$. This loss can be justified as a lower bound on the data log-likelihood [8, 33] or as a variant of denoising score matching [7, 35]. We will exploit in this paper the connection between DDPMs and LD sampling of EBMs, based upon which we achieve better sampling performance for LEBM compared with short-run LD.

3 Method

In this section, we introduce the diffusion-based amortization method for long-run MCMC sampling in learning LEBM in Section 3.1. The learning algorithm of LEBM based on the proposed method and details about implementation are then presented in Section 3.2 and Section 3.3, respectively.

3.1 Amortizing MCMC with DDPM

Amortized MCMC Using the same notation as in Section 2, we denote the starting distribution of LD as $\pi_0(\mathbf{z})$, the distribution after t -th iteration as $\pi_t(\mathbf{z})$ and the target distribution as $\pi(\mathbf{z})$. The trajectory of LD in Eq. (3) is typically specified by its transition kernel $\mathcal{K}(\mathbf{z}|\mathbf{z}')$. The process starts with drawing \mathbf{z}_0 from $\pi_0(\mathbf{z})$ and iteratively sample \mathbf{z}_t at the t -th iteration from the transition kernel conditioned on \mathbf{z}_{t-1} , *i.e.*, $\pi_t(\mathbf{z}) = \mathcal{K}\pi_{t-1}(\mathbf{z})$, where $\mathcal{K}\pi_{t-1}(\mathbf{z}) := \int \mathcal{K}(\mathbf{z}|\mathbf{z}')\pi_{t-1}(\mathbf{z}')d\mathbf{z}'$. Recursively, $\pi_t = \mathcal{K}_t\pi_0$, where \mathcal{K}_t denotes the t -step transition kernel. LD can therefore be viewed as approximating a fixed point update in a non-parametric fashion since the target distribution π is a stationary distribution $\pi(\mathbf{z}) := \int \mathcal{K}(\mathbf{z}|\mathbf{z}')\pi(\mathbf{z}')d\mathbf{z}'$, $\forall \mathbf{z}$. This motivates several works for more general approximations of this update [36–38] with the help of neural samplers.

Inspired by [36], we propose to use the following framework for amortizing the LD in learning the LEBM. Formally, let $\mathcal{Q} = \{q_\phi\}$ be the set of amortized samplers parameterized by ϕ . Given the transition kernel \mathcal{K} , the goal is to find a sampler q_{ϕ^*} to closely approximate the target distribution π . This can be achieved by iteratively distill T -step transitions of LD into q_ϕ :

$$q_{\phi_k} \leftarrow \arg \min_{q_\phi \in \mathcal{Q}} \mathcal{D}[q_{\phi_{k-1}, T} || q_\phi], \quad q_{\phi_{k-1}, T} := \mathcal{K}_T q_{\phi_{k-1}}, \quad q_{\phi_0} \approx \pi_0, \quad k = 0, \dots, K-1. \quad (6)$$

where $\mathcal{D}[\cdot || \cdot]$ is the Kullback-Leibler Divergence (KLD) measure between distributions. Concretely, Eq. (6) means that to recover the target distribution π , instead of using long-run LD, we can repeat the following steps: i) employ a T -step short-run LD initialized with the current sampler $q_{\phi_{k-1}}$ to approximate $\mathcal{K}_T q_{\phi_{k-1}}$ as the target distribution of the current sampler, and ii) update the current sampler $q_{\phi_{k-1}}$ to q_{ϕ_k} . The correct convergence of q_ϕ to π with Eq. (6) is supported by the standard theory of Markov chains [39], which suggests that the update in Eq. (6) is monotonically decreasing in terms of KLD, $\mathcal{D}[q_{\phi_k} || \pi] \leq \mathcal{D}[q_{\phi_{k-1}} || \pi]$. We refer to Appendix A.1 for a detailed explanation and discussion of this statement. In practice, one can apply gradient-based methods to minimize $\mathcal{D}[q_{\phi_{k-1}, T} || q_\phi]$ and approximate Eq. (6) for the update from $q_{\phi_{k-1}}$ to q_{ϕ_k} . The above formulation provides a generic and flexible framework for amortizing the potentially long MCMC.

Diffusion-based amortization To avoid clutter, we simply write $q_{\phi_{k-1}, T}$ as q_T . We can see that

$$\arg \min_{q_\phi} \mathcal{D}[q_T || q_\phi] = \arg \min_{q_\phi} -\mathcal{H}(q_T) + \mathcal{H}(q_T, q_\phi) = \arg \min_{q_\phi} -\mathbb{E}_{q_T} [\log q_\phi], \quad (7)$$

where \mathcal{H} represents the entropy of distributions. The selection of the sampler q_ϕ is a matter of design. According to Eq. (7), we may expect the following properties from q_ϕ : i) having analytically tractable expression of the exact value or lower bound of log-likelihood, ii) easy to draw samples from and iii) capable of close approximation to the given distribution $\{q_T\}$. In practice, iii) is important for the convergence of Eq. (6). If q_ϕ is far away from q_T in each iteration, then non-increasing KLD property $\mathcal{D}[q_{\phi_k} || \pi] \leq \mathcal{D}[q_{\phi_{k-1}} || \pi]$ might not hold, and the resulting amortized sampler would not converge to the true target distribution $\pi(\mathbf{z})$.

For the choice of q_ϕ , let us consider distilling the gradient field of q_T in each iteration, so that the resulting sampler is close to the q_T distribution. This naturally points to the DDPMs [8]. To be specific, learning a DDPM with ϵ -prediction parameterization is equivalent to fitting the finite-time marginal of a sampling chain resembling annealed Langevin dynamics [7, 8, 31]. Moreover, it also fulfills i) and ii) of the desired properties mentioned above. We can plug in the objective of DDPM (Section 2.2), which is a lower bound of $\log q_\phi$, to obtain the gradient-based update rule for q_ϕ :

$$\phi_{k-1}^{(i+1)} \leftarrow \phi_{k-1}^{(i)} - \eta \nabla_{\phi} \mathbb{E}_{\epsilon, \lambda} [\|\epsilon(\mathbf{z}_\lambda) - \epsilon\|_2^2], \quad \phi_k^{(0)} \leftarrow \phi_{k-1}^{(M)}, \quad i = 0, 1, \dots, M-1 \quad (8)$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I}_d)$. λ is drawn from a distribution of log noise-to-signal ratio $p(\lambda)$. η is the step size for the update, and M is the number of iterations needed in Eq. (8). In practice, we find that when amortizing the LD sampling chain, a light-weight DDPM q_ϕ updated with very small M , *i.e.*, few steps of Eq. (8) iteration, approximates Eq. (6) well. We provide a possible explanation using the Fisher information by scoping the asymptotic behavior of this update rule in the Appendix A.2. We term the resulting sampler as Diffusion-Amortized MCMC (DAMC).

3.2 Approximate MLE with DAMC

In this section, we show how to integrate the DAMC sampler into the learning framework of LEBM and form a symbiosis between these models. Given a set of N training samples $\{\mathbf{x}_i\}_{i=1}^N$ independently drawn from the unknown data distribution $p_{\text{data}}(\mathbf{x})$, the model p_θ (Section 2.1) can be trained

by maximizing the log-likelihood over training samples $\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \log p_{\theta}(\mathbf{x}_i)$. Doing so typically requires computing the gradients of $\mathcal{L}(\theta)$, where for each \mathbf{x}_i the learning gradient satisfies:

$$\begin{aligned} \nabla_{\theta} \log p_{\theta}(\mathbf{x}_i) &= \mathbb{E}_{p_{\theta}(\mathbf{z}_i|\mathbf{x}_i)} [\nabla_{\theta} \log p_{\theta}(\mathbf{z}_i, \mathbf{x}_i)] \\ &= \underbrace{\left(\mathbb{E}_{p_{\theta}(\mathbf{z}_i|\mathbf{x}_i)} [\nabla_{\alpha} f_{\alpha}(\mathbf{z}_i)] - \mathbb{E}_{p_{\alpha}(\mathbf{z}_i)} [\nabla_{\alpha} f_{\alpha}(\mathbf{z}_i)] \right)}_{\delta_{\alpha}(\mathbf{x}_i)} \underbrace{\mathbb{E}_{p_{\theta}(\mathbf{z}_i|\mathbf{x}_i)} [\nabla_{\beta} \log p_{\beta}(\mathbf{x}_i|\mathbf{z}_i)]}_{\delta_{\beta}(\mathbf{x}_i)}. \end{aligned} \quad (9)$$

Intuitively, based on the discussion in Section 2.1 and Section 3.1, we can approximate the distributions in Eq. (9) by drawing samples from $[\mathbf{z}_i|\mathbf{x}_i] \sim \mathcal{K}_{T, \mathbf{z}_i|\mathbf{x}_i} q_{\phi_k}(\mathbf{z}_i|\mathbf{x}_i)$, $\mathbf{z}_i \sim \mathcal{K}_{T, \mathbf{z}_i} q_{\phi_k}(\mathbf{z}_i)$, to estimate the expectations and hence the learning gradient. Here we learn the DAMC samplers $q_{\phi_k}(\mathbf{z}_i|\mathbf{x}_i)$ and $q_{\phi_k}(\mathbf{z}_i)$ for the posterior and prior sampling chain, respectively. ϕ_k represents the current sampler as in Section 3.1; $\mathcal{K}_{T, \mathbf{z}_i|\mathbf{x}_i}$ and $\mathcal{K}_{T, \mathbf{z}_i}$ are the transition kernels for posterior and prior sampling chain. Equivalently, this means to better estimate the learning gradients we can i) first draw approximate posterior and prior MCMC samples from the current q_{ϕ_k} model, and ii) update the approximation of the prior $p_{\alpha}(\mathbf{z})$ and posterior $p_{\theta}(\mathbf{z}|\mathbf{x})$ distributions with additional T -step LD initialized with q_{ϕ_k} samples. These updated samples are closer to $p_{\theta}(\mathbf{z}_i|\mathbf{x}_i)$ and $p_{\alpha}(\mathbf{z}_i)$ compared with short-run LD samples based on our discussion in Section 2.2. Consequently, the diffusion-amortized LD samples provide a generally better estimation of the learning gradients and lead to better performance, as we will show empirically in Section 4. After updating $\theta = (\alpha, \beta)$ based on these approximate samples with Eq. (9), we can update q_{ϕ_k} with Eq. (8) to distill the sampling chain into $q_{\phi_{k+1}}$. As shown in Fig. 1, we can see that the whole learning procedure iterates between the approximate MLE of p_{θ} and the amortization of MCMC with q_{ϕ} . We refer to Appendix A.3 for an extended discussion of this procedure.

After learning the models, we can use either DAMC or LEBM for prior sampling. For DAMC, we may draw samples from $q_{\phi}(\mathbf{z}_i)$ with Eq. (5). Prior sampling with LEBM still requires short-run LD initialized from $\mathcal{N}(0, \mathbf{I}_d)$. For posterior sampling, we may sample from $\mathcal{K}_{T, \mathbf{z}_i|\mathbf{x}_i} q_{\phi_k}(\mathbf{z}_i|\mathbf{x}_i)$, *i.e.*, first draw samples from DAMC and then run few steps of LD to obtain posterior samples.

Algorithm 1: Learning algorithm of DAMC.

Input: initial parameters (α, β, ϕ) ; learning rate $\eta = (\eta_{\alpha}, \eta_{\beta}, \eta_{\phi})$; observed examples $\{\mathbf{x}^{(i)}\}_{i=1}^N$; prob. of uncond. training p_{uncond} for the DAMC sampler.

Output: $(\theta^{(K)} = \{\alpha^{(K)}, \beta^{(K)}\}, \phi^{(K)})$.

1 **for** $k = 0 : K - 1$ **do**

2 Sample a minibatch of data $\{\mathbf{x}^{(i)}\}_{i=1}^B$;

3 **Draw DAMC samples:** For each $\mathbf{x}^{(i)}$, draw $\mathbf{z}_+^{(i)}$ and $\mathbf{z}_-^{(i)}$ from $q_{\phi_k}(\mathbf{z}_i|\mathbf{x}_i)$.

4 **Prior LD update:** For each $\mathbf{x}^{(i)}$, update $\mathbf{z}_-^{(i)}$ using Eq. (3), with $\pi(\mathbf{z}_i) = p_{\alpha^{(k)}}(\mathbf{z}_i)$;

5 **Posterior LD update:** For each $\mathbf{x}^{(i)}$, update $\mathbf{z}_+^{(i)}$ using Eq. (3), with $\pi(\mathbf{z}_i) = p_{\beta^{(k)}}(\mathbf{z}_i|\mathbf{x}_i)$;

6 **Update $\theta^{(k)}$:** Update $\alpha^{(k)}$ and $\beta^{(k)}$ using Monte-Carlo estimates (*i.e.*, Monte-Carlo average) of Eq. (9) with $\{\mathbf{z}_+^{(i)}\}_{i=1}^B$ and $\{\mathbf{z}_-^{(i)}\}_{i=1}^B$.

7 **Update $\phi^{(k)}$:** Update $\phi^{(k)}$ using Eq. (8) with p_{uncond} and $\{\mathbf{z}_+^{(i)}\}_{i=1}^B$ as the target.

3.3 Implementation

In order to efficiently model both $q_{\phi_k}(\mathbf{z}_i|\mathbf{x}_i)$ and $q_{\phi_k}(\mathbf{z}_i)$, we follow the method of [40] to train a single network to parameterize both models, where $q_{\phi_k}(\mathbf{z}_i|\mathbf{x}_i)$ can be viewed as a conditional DDPM with the embedding of \mathbf{x}_i produced by an encoder network as its condition, and $q_{\phi_k}(\mathbf{z}_i)$ an unconditional one. For $q_{\phi_k}(\mathbf{z}_i)$, we can input a null token \emptyset as its condition when predicting the noise ϵ . We jointly train both models by randomly nullifying the inputs with the probability $p_{\text{uncond}} = 0.2$. During training, we use samples from $q_{\phi_k}(\mathbf{z}_i|\mathbf{x}_i)$ to initialize both prior and posterior updates for training stability. For the posterior and prior DAMC samplers, we set the number of diffusion steps to 100. The number of iterations in Eq. (8) is set to $M = 6$ throughout the experiments. The LD runs 30 and 60 iterations for posterior and prior updates during training with a step

Table 1: **MSE(↓) and FID(↓) obtained from models trained on different datasets.** The FID scores are computed based on 50k generated images and training images for the first three datasets and 5k images for the CelebA-HQ dataset. The MSEs are computed based on unseen testing images. We highlight our model results in gray color. The best and second-best performances are marked in bold numbers and underlines, respectively; tables henceforth follow this format. *[41] uses a prior model with 4x parameters compared with [22] and ours.

Model	SVHN		CelebA		CIFAR-10		CelebA-HQ	
	MSE	FID	MSE	FID	MSE	FID	MSE	FID
VAE [1]	0.019	46.78	0.021	65.75	0.057	106.37	0.031	180.49
2s-VAE [48]	0.019	42.81	0.021	44.40	0.056	72.90	-	-
RAE [49]	0.014	40.02	0.018	40.95	0.027	74.16	-	-
NCP-VAE [50]	0.020	33.23	0.021	42.07	0.054	78.06	-	-
Adaptive CE* [41]	<u>0.004</u>	26.19	<u>0.009</u>	35.38	0.008	65.01	-	-
ABP [51]	-	49.71	-	51.50	0.018	90.30	0.025	160.21
SRI [24]	0.018	44.86	0.020	61.03	-	-	-	-
SRI (L=5) [24]	0.011	35.32	0.015	47.95	-	-	-	-
LEBM [22]	0.008	29.44	0.013	37.87	0.020	70.15	<u>0.025</u>	133.07
Ours-LEBM	0.002	<u>21.17</u>	0.005	<u>35.67</u>	<u>0.015</u>	<u>60.89</u>	0.023	<u>89.54</u>
Ours-DAMC		18.76		30.83		57.72		85.88

size of $s = 0.1$. For test time sampling from $\mathcal{K}_{T, z_i | x_i} q_{\phi_k}(z_i | x_i)$, $T = 10$ for the additional LD. For a fair comparison, we use the same LEBM and generator as in [22, 41] for all the experiments. We summarize the learning algorithm in Algorithm 1. Please see Appendices B and C for network architecture and further training details, as well as the pytorch-style pseudocode of the algorithm.

4 Experiments

In this section, we are interested in the following questions: (i) How does the proposed method compare with its previous counterparts (*e.g.*, purely MCMC-based or variational methods)? (ii) How is the scalability of this method? (iii) How are the time and parameter efficiencies? (iv) Does the proposed method provide a desirable latent space? To answer these questions, we present a series of experiments on benchmark datasets including MNIST [42], SVHN [43], CelebA64 [44], CIFAR-10 [45], CelebAMask-HQ [46], FFHQ [10] and LSUN-Tower [47]. As to be shown, the proposed method demonstrates consistently better performance in various experimental settings compared with previous methods. We refer to Appendix D for details about the experiments.

4.1 Generation and Inference: Prior and Posterior Sampling

Generation and reconstruction We evaluate the quality of the generated and reconstructed images to examine the sampling quality of DAMC. Specifically, we would like to check i) how well does DAMC fit the seen data, ii) does DAMC provide better MCMC samples for learning LEBM and iii) the generalizability of DAMC on unseen data. We check the goodness of fit of DAMC by evaluating the quality of the images generated with DAMC prior samples. If DAMC does provide better MCMC samples for learning LEBM, we would expect better fitting of data and hence an improved generation quality of LEBM. We evaluate the performance of posterior sampling given unseen testing images by examining the reconstruction error on testing data. We benchmark our model against a variety of previous methods in two groups. The first group covers competing methods that adopt the variational learning scheme, including VAE [1], as well as recent two-stage methods such as 2-stage VAE [48], RAE [49] and NCP-VAE [50], whose prior distributions are learned with posterior samples in a second stage after the generator is trained. The second group includes methods that adopt MCMC-based sampling. It includes Alternating Back-Propogation (ABP) [51], Short-Run Inference (SRI) from [24] and the vanilla learning method of LEBM [22], which relies on short-run LD for both posterior and prior sampling. We also compare our method with the recently proposed Adaptive CE [41]. It learns a series of LEBMs adaptively during training, while these LEBMs are sequentially updated by density ratio estimation instead of MLE. To make fair comparisons, we follow the same evaluation protocol as in [22, 41].

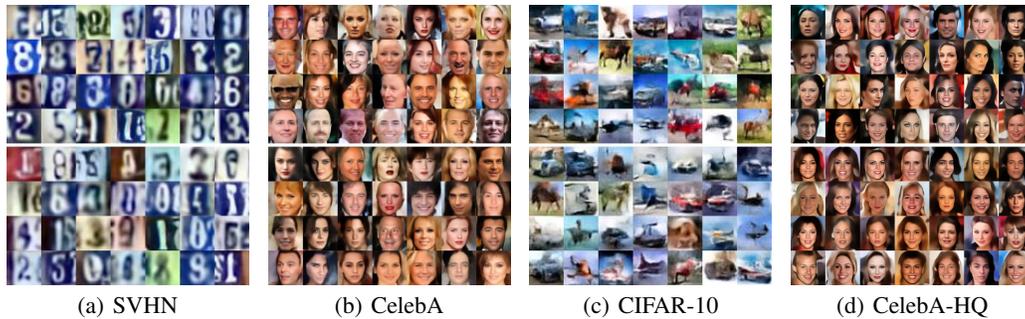


Figure 2: **Samples generated from the DAMC sampler and LEBM** trained on SVHN, CelebA, CIFAR-10 and CelebA-HQ datasets. In each sub-figure, the first four rows are generated by the DAMC sampler. The last four rows are generated by LEBM trained with the DAMC sampler.

For generation, we report the FID scores [52] in Table 1. We observe that i) the DAMC sampler, denoted as Ours-DAMC, provides superior generation performance compared to baseline models, and ii) the LEBM learned with samples from DAMC, denoted as Ours-LEBM, demonstrates significant performance improvement compared with the LEBM trained with short-run LD, denoted as LEBM. These results confirm that DAMC is a reliable sampler and indeed partly addresses the learning issue of LEBM caused by short-run LD. We would like to point out that the improvement is clearer on the CelebAMask-HQ dataset, where the input data is of higher dimension (256×256) and contains richer details compared with other datasets. This illustrates the superiority of DAMC sampler over short-run LD when the target distribution is potentially highly multi-modal. We show qualitative results of generated samples in Fig. 2, where we observe that our method can generate diverse, sharp and high-quality samples. For reconstruction, we compare our method with baseline methods in terms of MSE in Table 1. We observe that our method demonstrates competitive reconstruction error, if not better, than competing methods do. Additional qualitative results of generation and reconstruction are presented in Appendices E.2 and E.3.

GAN inversion We have examined the scalability of our method on the CelebAMask-HQ dataset. Next, we provide more results on high-dimensional and highly multi-modal data by performing GAN inversion [54] using the proposed method. Indeed, we may regard GAN inversion as an inference problem and a special case of posterior sampling. As a suitable testbed, the StyleGAN structure [10] is specifically considered as our generator in the experiments: [53] points out that to effectively infer the latent representation of a given image, the GAN inversion method needs to consider an extended latent space of StyleGAN, consisting of 14 different 512-dimensional latent vectors. We attempt to use the DAMC sampler for GAN inversion. We benchmark our method against i) learning an encoder that maps a given image to the latent space [16], which relates to the variational methods for posterior inference, ii) optimizing a random initial latent code by minimizing the reconstruction error and perceptual loss [53], which can be viewed as a variant of LD sampling, and iii) optimizing the latent code by minimizing both the objectives used in ii) and the energy score provided by LEBM. We use the pretrained weights provided by [18] for the experiments. Both the DAMC sampler and the encoder-based method are augmented with 100 post-processing optimization iterations. We refer to Appendix D for more experiment details. We test LEBM-based inversion with different optimization iterations. To be specific, 1x, 2x, and 4x represent 100, 200, and 400 iterations respectively. We can see in Table 2 that DAMC performs better than all the baseline methods on the unseen testing data, which supports the efficacy of our method in high-dimensional settings. We provide qualitative results in Fig. 3.

Parameter efficiency and sampling time One potential disadvantage of our method is its parameter inefficiency for introducing an extra DDPM. Fortunately, our models are in the latent space

Model	FFHQ		LSUN-T	
	MSE	FID	MSE	FID
Opt. [53]	0.055	149.39	0.080	240.11
Enc. [16]	<u>0.028</u>	<u>62.32</u>	0.079	132.41
[22] w/ 1x	0.054	149.21	0.072	239.51
[22] w/ 2x	0.039	101.59	0.066	163.20
[22] w/ 4x	0.032	84.64	<u>0.059</u>	<u>111.53</u>
Ours	0.025	52.85	0.059	80.42

Table 2: **MSE(↓) and FID(↓) for GAN inversion on different datasets.** Opt. and Enc. denotes the optimization-based and encoder-based methods.

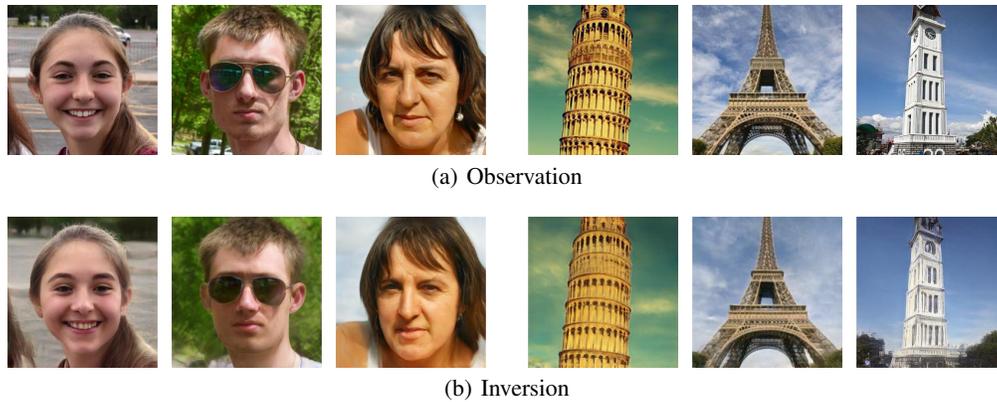


Figure 3: **Qualitative results of StyleGAN inversion using the DAMC sampler.** In each sub-figure, the left panel contain samples from the FFHQ dataset, and the right panel contains samples from the LSUN-T dataset.

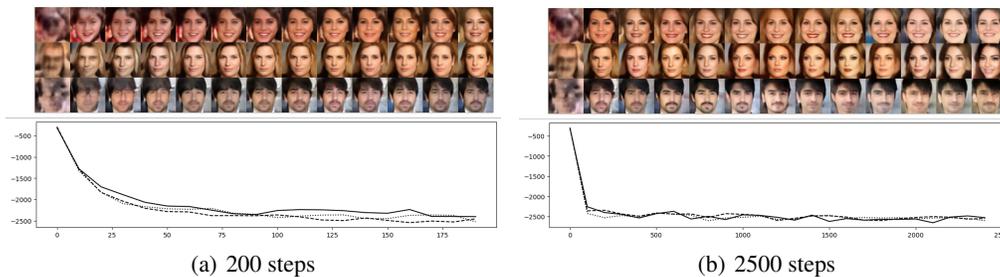


Figure 4: **Transition of Markov chains initialized from $\mathcal{N}(0, \mathbf{I}_d)$ towards $p_\alpha(z)$.** We present results by running LD for 200 and 2500 steps. In each sub-figure, the top panel displays the trajectory in the data space uniformly sampled along the chain. The bottom panel shows the energy score $f_\alpha(z)$ over the iterations.

so the network is lightweight. To be specific, on CIFAR-10 dataset the number of parameters in the DDPM is only around 10% (excluding the encoder) of those in the generator. The method has competitive time efficiency. With the batch size of 64, the DAMC prior sampling takes 0.3s, while 100 steps of short-run LD with LEBM takes 0.2s. The DAMC posterior sampling takes 1.0s, while LEBM takes 8.0s. Further discussions about the limitations can be found in the Appendix G.1.

4.2 Analysis of Latent Space

Long-run langevin transition In this section, we examine the energy landscape induced by the learned LEBM. We expect that a well-trained $p_\alpha(z)$ fueled by better prior and posterior samples from the DAMC sampler would lead to energy landscape with regular geometry. In Fig. 4, we visualize the transition of LD initialized from $\mathcal{N}(0, \mathbf{I}_d)$ towards $p_\alpha(z)$ on the model trained on the CelebA dataset. Additional visualization of transitions on SVHN and CIFAR-10 datasets can be found in the Appendix E.4. The LD iterates for 200 and 2500 steps, which is longer than the LD within each training iteration (60 steps). For the 200-step set-up, we can see that the generation quality quickly improves by exploring the local modes (demonstrating different facial features, e.g., hairstyle, facial expression and lighting). For the 2500-step long-run set-up, we can see that the LD produces consistently valid results without the oversaturating issue of the long-run chain samples [23]. These observations provide empirical evidence that the LEBM is well-trained.

Anomaly detection We further evaluate how the LEBM learned by our method could benefit the anomaly detection task. With properly learned models, the posterior $p_{\theta, \phi}(z|\mathbf{x})$ could form a discriminative latent space that has separated probability densities for in-distribution (normal) and out-of-distribution (anomalous) data. Given the testing sample \mathbf{x} , we use un-normalized log joint density $p_{\theta, \phi}(z|\mathbf{x}) \propto p_{\theta, \phi}(\mathbf{x}, z) \approx p_{\beta}(\mathbf{x}|z)p_{\alpha}(z)|_{z \sim \mathcal{K}_{T, z|\mathbf{x}} q_{\phi}(z|\mathbf{x})}$ as our decision function. This means that we draw samples from $\mathcal{K}_{T, z|\mathbf{x}} q_{\phi}(z|\mathbf{x})$ and compare the corresponding reconstruction errors and energy scores. A higher value of log joint density indicates a higher probability of the test sample being a normal sample. To make fair comparisons, we follow the experimental settings in [22, 41, 55, 56] and train our model on MNIST with one class held out as an anomalous class. We consider the

Table 3: **AUPRC(\uparrow) scores for unsupervised anomaly detection on MNIST.** Numbers are taken from [41]. Results of our model are averaged over the last 10 trials to account for variance.

Heldout Digit	1	4	5	7	9
VAE [1]	0.063	0.337	0.325	0.148	0.104
ABP [51]	0.095 ± 0.03	0.138 ± 0.04	0.147 ± 0.03	0.138 ± 0.02	0.102 ± 0.03
MEG [55]	0.281 ± 0.04	0.401 ± 0.06	0.402 ± 0.06	0.290 ± 0.04	0.342 ± 0.03
BiGAN- σ [56]	0.287 ± 0.02	0.443 ± 0.03	0.514 ± 0.03	0.347 ± 0.02	0.307 ± 0.03
LEBM [22]	0.336 ± 0.01	0.630 ± 0.02	0.619 ± 0.01	0.463 ± 0.01	0.413 ± 0.01
Adaptive CE [41]	0.531 ± 0.02	0.729 ± 0.02	0.742 ± 0.01	0.620 ± 0.02	0.499 ± 0.01
Ours	0.684 ± 0.02	0.911 ± 0.01	0.939 ± 0.02	0.801 ± 0.01	0.705 ± 0.01

Table 4: **Ablation study on CIFAR-10 dataset.** VI denotes learning LEBM using variational methods. SR denotes learning LEBM with short-run LD. DAMC-G replaces the LEBM in DAMC-LEBM with a standard Gaussian distribution. NALR denotes the non-amortized DDPM setting. For each set-up, we provide results using the vanilla sampling method, denoted as V, and the ones using the DAMC sampler, denoted as D.

Model	VI-LEBM		SR-LEBM		DAMC-G		NALR-LEBM		DAMC-LEBM	
	V.	D.	V.	D.	V.	D.	V.	D.	V.	D.
MSE	0.054	-	0.020	-	0.018	0.015	0.028	0.016	0.021	0.015
FID	78.06	-	70.15	-	90.30	66.93	68.52	64.38	60.89	57.72

baseline models that employ MCMC-based or variational inferential mechanisms. Table 3 shows the results of AUPRC scores averaged over the last 10 trials. We observe significant improvements in our method over the previous counterparts.

4.3 Ablation Study

In this section, we conduct ablation study on several variants of the proposed method. Specifically, we would like to know: i) what is the difference between the proposed method and directly training a DDPM in a fixed latent space? ii) What is the role of LEBM in this learning scheme? iii) Does DAMC effectively amortize the sampling chain? We use CIFAR-10 dataset for the ablative experiments to empirically answer these questions. More ablation studies can be found in Appendix F.

Non-Amortized DDPM vs. DAMC We term directly training a DDPM in a fixed latent space as the non-amortized DDPM. To analyze the difference between non-amortized DDPM and DAMC, we first train a LEBM model with persistent long-run chain sampling [57] and use the trained model to obtain persistent samples for learning the non-amortized DDPM. In short, the non-amortized DDPM can be viewed as directly distilling the long-run MCMC sampling process, instead of progressively amortizing the chain. We present the FID and MSE of the non-amortized model (NALR) in Table 4. We observe that directly amortizing the long-run chain leads to degraded performance compared with the proposed method. The results are consistently worse for both posterior and prior sampling and the learned LEBMs, which verify the effectiveness of the proposed iterative learning scheme.

The contribution of LEBM One may argue that since we have the DAMC as a powerful sampler, it might not be necessary to jointly learn LEBM in the latent space. To demonstrate the necessity of this joint learning scheme, we train a variant of DAMC by replacing the LEBM with a Gaussian prior. The results are presented in Table 4. We observe that models trained with non-informative Gaussian prior obtain significantly worse generation results. It suggests that LEBM involved in the learning iteration provides positive feedback to the DAMC sampler. Therefore, we believe that it is crucial to jointly learn the DAMC and LEBM.

Vanilla sampling vs. DAMC We compare the vanilla sampling process of each model with DAMC. The vanilla sampling typically refers to short-run or long-run LD initialized with $\mathcal{N}(0, I_d)$. We also provide results of learning LEBM using variational methods for comparison. We can see in Table 4 that sampling with DAMC shows significantly better scores of the listed models, compared with vanilla sampling. The result is even better than that of the persistent chain sampler (V. of NALR-LEBM). This indicates that DAMC effectively amortizes the sampling chain. Comparing DAMC sampler with the variational sampler also indicates that DAMC is different from general variational approximation: it benefits from its connection with LD and shows better expressive power.

5 Related Work

Energy-based prior model EBMs [23, 24, 58–60] play an important role in generative modeling. Pang et al. [22] propose to learn an EBM as a prior model in the latent space of DLVMs; it greatly improves the model expressivity over those with non-informative priors and brings strong performance on downstream tasks, *e.g.*, image segmentation, text modeling, molecule generation, and trajectory prediction [12, 15, 61, 62]. However, learning EBMs or latent space EBMs requires MCMC sampling to estimate the learning gradients, which needs numerous iterations to converge when the target distributions are high-dimensional or highly multi-modal. Typical choices of sampling with non-convergent short-run MCMC [24] in practice can lead to poor generation quality, malformed energy landscapes [15, 24, 25], biased estimation of the model parameter and instability in training [23, 25, 59, 60]. In this work, we consider learning valid amortization of the long-run MCMC for energy-based priors; the proposed model shows reliable sampling quality in practice.

Denosing diffusion probabilistic model DDPMs [7, 8, 31], originating from [5], learn the generative process by recovering the observed data from a sequence of noise-perturbed versions of the data. The learning objective can be viewed as a variant of the denoising score matching objective [35]. As pointed out in [5, 8], the sampling procedure of DDPM with ϵ -prediction parameterization resembles LD of an EBM; ϵ (predicted noise) plays a similar role to the gradient of the log density [8]. To be specific, learning a DDPM with ϵ -prediction parameterization is equivalent to fitting the finite-time marginal of a sampling chain resembling annealed Langevin dynamics [7, 8, 31]. Inspired by this connection, we propose to amortize the long-run MCMC in learning energy-based prior by iteratively distilling the short-run chain segments with a DDPM-based sampler. We show empirically and theoretically that the learned sampler is valid for long-run chain sampling.

Amortized MCMC The amortized MCMC technique is formally brought up by Li et al. [36], which incorporates feedback from MCMC back to the parameters of the amortizer distribution q_ϕ . It is concurrently and independently proposed by Xie et al. [37] as the MCMC teaching framework. Methods under this umbrella term [36–38, 63–66] generally learn the amortizer by minimizing the divergence (typically the KLD) between the improved distribution and its initialization, *i.e.*, $D[\mathcal{K}_T q_{\phi_{k-1}} || q_\phi]$, where \mathcal{K}_T represents T -step MCMC transition kernel and $q_{\phi_{k-1}}$ represents the current amortizer. The diffusion-based amortization proposed in this work can be viewed as an instantiation of this framework, while our focus is on learning the energy-based prior. Compared with previous methods, our method i) specifically exploits the connection between EBMs and DDPMs and is suitable for amortizing the prior and posterior sampling MCMC of energy-based prior, and ii) resides in the lower-dimensional latent space and enables faster sampling and better convergence.

More methods for learning EBM Several techniques other than short-run MCMC have been proposed to learn the EBM. In the seminal work, Hinton [67] proposes to initialize Markov chains using real data and run several steps of MCMC to obtain samples from the model distribution. Tieleman [57] proposes to start Markov chains from past samples in the previous sampling iteration, known as Persistent Contrastive Divergence (PCD) or persistent chain sampling, to mimic the long-run sampling chain. Nijkamp et al. [23] provide comprehensive discussions about tuning choices for LD such as the step size s and sampling steps T to obtain stable long-run samples for persistent training. [59, 60] employ a hybrid of persistent chain sampling and short-run sampling by maintaining a buffer of previous samples. The methods draw from the buffer or initialize the short-run chain with noise distribution with some pre-specified probability. Another branch of work, stemmed from [68], considers discriminative contrastive estimation to avoid MCMC sampling. Gao et al. [69] use a normalizing flow [4] as the base distribution for contrastive estimation. Aneja et al. [50] propose to estimate the energy-based prior model based on the prior of a pre-trained VAE [70] by noise contrastive estimation. More recently, Xiao and Han [41] learn a sequence of EBMs in the latent space with adaptive multi-stage NCE to further improve the expressive power of the model.

6 Conclusion

In this paper, we propose the DAMC sampler and develop a novel learning algorithm for LEBM based on it. We provide theoretical and empirical evidence for the effectiveness of our method. We notice that our method can be applied to amortizing MCMC sampling of unnormalized continuous densities in general. It can also be applied to sampling posterior distributions of continuous latent variables in general latent variable models. We would like to explore these directions in future work.

Acknowledgements

Y. N. Wu was supported by NSF DMS-2015577. We would like to thank the anonymous reviewers for their constructive comments.

References

- [1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [2] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2014.
- [3] 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 (NeurIPS)*, 2014.
- [4] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.
- [5] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015.
- [6] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 2019.
- [7] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [9] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2020.
- [10] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.
- [11] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.
- [12] Peiyu Yu, Sirui Xie, Xiaojian Ma, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. Unsupervised foreground extraction via deep region competition. *Advances in Neural Information Processing Systems*, 2021.
- [13] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [14] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [15] Peiyu Yu, Sirui Xie, Xiaojian Ma, Baoxiong Jia, Bo Pang, Ruigi Gao, Yixin Zhu, Song-Chun Zhu, and Ying Nian Wu. Latent diffusion energy-based model for interpretable text modeling. *arXiv preprint arXiv:2206.05895*, 2022.
- [16] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.

- [17] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 2017.
- [18] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.
- [19] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [20] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations (ICLR)*, 2020.
- [21] Weili Nie, Arash Vahdat, and Anima Anandkumar. Controllable and compositional generation with latent-space energy-based models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [22] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [23] 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. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2020.
- [24] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [25] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. In *International Conference on Learning Representations (ICLR)*, 2020.
- [26] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, 2016.
- [27] Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. *arXiv preprint arXiv:1901.03416*, 2019.
- [28] James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don’t blame the elbo! a linear vae perspective on posterior collapse. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [29] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016.
- [30] Jakub Tomczak and Max Welling. Vae with a vampprior. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.
- [31] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [32] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of International Conference on Machine Learning (ICML)*, 2011.
- [33] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [34] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021.
- [35] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

- [36] Yingzhen Li, Richard E Turner, and Qiang Liu. Approximate inference with amortised mcmc. *arXiv preprint arXiv:1702.08343*, 2017.
- [37] 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 AAAI Conference on Artificial Intelligence (AAAI)*, 2018.
- [38] Francisco Ruiz and Michalis Titsias. A contrastive divergence for combining variational inference and mcmc. In *Proceedings of International Conference on Machine Learning (ICML)*, 2019.
- [39] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.
- [40] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.
- [41] Zhisheng Xiao and Tian Han. Adaptive multi-stage density ratio estimation for learning latent space energy-based model. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [42] Yann LeCun. The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>, 1998.
- [43] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.
- [44] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.
- [45] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.
- [46] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [47] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.
- [48] Bin Dai and David Wipf. Diagnosing and enhancing vae models. *International Conference on Learning Representations (ICLR)*, 2019.
- [49] Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. *International Conference on Learning Representations (ICLR)*, 2020.
- [50] Jyoti Aneja, Alex Schwing, Jan Kautz, and Arash Vahdat. A contrastive learning approach for training variational autoencoder priors. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [51] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2017.
- [52] 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. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [53] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.
- [54] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

- [55] Rithesh Kumar, Sherjil Ozair, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *arXiv preprint arXiv:1901.08508*, 2019.
- [56] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient gan-based anomaly detection. *Proceedings of the 20th IEEE International Conference on Data Mining (ICDM)*, 2018, 2018.
- [57] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of International Conference on Machine Learning (ICML)*, 2008.
- [58] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016.
- [59] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.
- [60] Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021.
- [61] Bo Pang, Tian Han, and Ying Nian Wu. Learning latent space energy-based prior model for molecule generation. *arXiv preprint arXiv:2010.09351*, 2020.
- [62] Bo Pang, Tianyang Zhao, Xu Xie, and Ying Nian Wu. Trajectory prediction with latent belief energy-based model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [63] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Generative voxelnet: learning energy-based models for 3d shape synthesis and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2468–2484, 2020.
- [64] Jianwen Xie, Zilong Zheng, and Ping Li. Learning energy-based model with variational auto-encoder as amortized sampler. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2021.
- [65] Jianwen Xie, Yaxuan Zhu, Jun Li, and Ping Li. A tale of two flows: cooperative learning of langevin flow and normalizing flow toward energy-based model. *arXiv preprint arXiv:2205.06924*, 2022.
- [66] Yaxuan Zhu, Jianwen Xie, Yingnian Wu, and Ruiqi Gao. Learning energy-based models by cooperative diffusion recovery likelihood. *arXiv preprint arXiv:2309.05153*, 2023.
- [67] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.
- [68] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.
- [69] Ruiqi Gao, Erik Nijkamp, Diederik P Kingma, Zhen Xu, Andrew M Dai, and Ying Nian Wu. Flow contrastive estimation of energy-based models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [70] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 2020.
- [71] JM Bernardo, MJ Bayarri, JO Berger, AP Dawid, D Heckerman, AFM Smith, M West, et al. The variational bayesian em algorithm for incomplete data: with application to scoring graphical model structures. *Bayesian statistics*, 7(453-464):210, 2003.
- [72] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

- [73] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.
- [74] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [75] Alexander Li and Deepak Pathak. Functional regularization for reinforcement learning via learned fourier features. *Advances in Neural Information Processing Systems*, 2021.
- [76] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.
- [77] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [78] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [79] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [80] Shohei Taniguchi, Yusuke Iwasawa, Wataru Kumagai, and Yutaka Matsuo. Langevin autoencoders for learning deep latent variable models. *Advances in Neural Information Processing Systems*, 35:13277–13289, 2022.

A Theoretical Discussion

A.1 Monotonically Decreasing KLD

We state in the main text that $\mathcal{D}[q_{\phi_k}|\pi] \leq \mathcal{D}[q_{\phi_{k-1}}|\pi]$, where π is the stationary distribution. To show this, we first provide a proof of $\mathcal{D}[\pi_{t+T}|\pi] \leq \mathcal{D}[\pi_t|\pi]$, where π_t and π_{t+T} are the distributions of \mathbf{z} at t -th and $(t+T)$ -th iteration, respectively. This is a known result from [39], and we include it here for completeness.

$$\begin{aligned} \mathcal{D}[\pi_t|\pi] &= \mathbb{E}_{\pi_t(\mathbf{z}_t)} \left[\log \frac{\pi_t(\mathbf{z}_t)}{\pi(\mathbf{z}_t)} \right] = \mathbb{E}_{\pi_t(\mathbf{z}_t), \mathcal{K}(\mathbf{z}_{t+1}|\mathbf{z}_t)} \left[\log \frac{\pi_t(\mathbf{z}_t)\mathcal{K}(\mathbf{z}_{t+1}|\mathbf{z}_t)}{\pi(\mathbf{z}_t)\mathcal{K}(\mathbf{z}_{t+1}|\mathbf{z}_t)} \right] \\ &\stackrel{(i)}{=} \mathbb{E}_{\pi_{t+1}(\mathbf{z}_{t+1}), \mathcal{K}'_{\pi_{t+1}}(\mathbf{z}_t|\mathbf{z}_{t+1})} \left[\log \frac{\mathcal{K}'_{\pi_{t+1}}(\mathbf{z}_t|\mathbf{z}_{t+1})\pi_{t+1}(\mathbf{z}_{t+1})}{\mathcal{K}'_{\pi}(\mathbf{z}_t|\mathbf{z}_{t+1})\pi(\mathbf{z}_{t+1})} \right] \\ &= \mathcal{D}[\pi_{t+1}|\pi] + \mathbb{E}_{\pi_{t+1}(\mathbf{z}_{t+1})} \mathcal{D}[\mathcal{K}'_{\pi_{t+1}}(\mathbf{z}_t|\mathbf{z}_{t+1})|\mathcal{K}'_{\pi}(\mathbf{z}_t|\mathbf{z}_{t+1})] \\ &\stackrel{(ii)}{\geq} \mathcal{D}[\pi_{t+1}|\pi], \end{aligned} \quad (10)$$

where we denote the forward-time transition kernel as \mathcal{K} and the reverse-time kernel as \mathcal{K}' . (i) holds because we are just re-factorizing the joint density of $[\mathbf{z}_t, \mathbf{z}_{t+1}]$: $\pi_t(\mathbf{z}_t)\mathcal{K}(\mathbf{z}_{t+1}|\mathbf{z}_t) = \mathcal{K}'_{\pi_{t+1}}(\mathbf{z}_t|\mathbf{z}_{t+1})\pi_{t+1}(\mathbf{z}_{t+1})$ and $\pi(\mathbf{z}_t)\mathcal{K}(\mathbf{z}_{t+1}|\mathbf{z}_t) = \mathcal{K}'_{\pi}(\mathbf{z}_t|\mathbf{z}_{t+1})\pi(\mathbf{z}_{t+1})$. (ii) holds because the KLD is non-negative. We can see that $\mathcal{D}[\pi_{t+T}|\pi] \leq \mathcal{D}[\pi_t|\pi]$ is a direct result from Eq. (10), and that $\pi_t \rightarrow \pi$ as $t \rightarrow \infty$ under proper conditions [32].

In the main text, we describe the update rule of the sampler q_{ϕ} as follows:

$$q_{\phi_k} \leftarrow \arg \min_{q_{\phi} \in \mathcal{Q}} \mathcal{D}[q_{\phi_{k-1}, T}|q_{\phi}], \quad q_{\phi_{k-1}, T} := \mathcal{K}_T q_{\phi_{k-1}}, \quad q_{\phi_0} \approx \pi_0, \quad k = 0, \dots, K-1. \quad (11)$$

In the ideal case, we can assume that the objective in Eq. (6) is properly optimized and that $\{q_{\phi}\}$ is expressive enough to parameterize each $q_{\phi_{k-1}, T}$. With $q_{\phi_0} \approx \pi_0$ and $q_{\phi_k} \approx \mathcal{K}_T q_{\phi_{k-1}}$, we can conclude that $\mathcal{D}[q_{\phi_k}|\pi] \leq \mathcal{D}[q_{\phi_{k-1}}|\pi]$ for each $k = 1, \dots, K$ according to Eq. (10). We will discuss in the following section the scenario where we apply gradient-based methods to minimize $\mathcal{D}[q_{\phi_{k-1}, T}|q_{\phi}]$ and approximate Eq. (6) for the update from $q_{\phi_{k-1}}$ to q_{ϕ_k} .

A.2 Discussion about Diffusion-Based Amortization

We can see that the statement in Appendix A.1 holds when q_{ϕ_k} is a close approximation of $q_T := \mathcal{K}_T q_{\phi_{k-1}}$. This motivates our choice of employing DDPM to amortize the LD transition, considering its capability of close approximation to the given distribution $\{q_T\}$. Based on the derivation of DDPM learning objective in [1], we know that

$$\begin{aligned} \arg \min_{q_{\phi}} \mathcal{D}[q_T|q_{\phi}] &= \arg \min_{q_{\phi}} -\mathcal{H}(q_T) + \mathcal{H}(q_T, q_{\phi}) = \arg \min_{q_{\phi}} -\mathbb{E}_{q_T} [\log q_{\phi}] \\ &\leq \arg \min_{q_{\phi}} \mathbb{E}_{\epsilon, \lambda} [\|\epsilon_{\phi}(\mathbf{z}_{\lambda}) - \epsilon\|_2^2] \approx \arg \min_{q_{\phi}} \frac{1}{N} \sum_{j=1}^N [\|\epsilon_{\phi}(\mathbf{z}_{j, \lambda_j}) - \epsilon_j\|_2^2], \end{aligned} \quad (12)$$

where \mathbf{z}_{λ} is draw from $q(\mathbf{z}_{\lambda}|\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_{\lambda}; \beta_{\lambda}\mathbf{z}_0, \sigma_{\lambda}^2\mathbf{I}_d)$. $\mathbf{z}_0 \sim q_T$. $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and λ is drawn from a distribution of log noise-to-signal ratio $p(\lambda)$. In practice, we use Monte-Carlo average to approximate the objective and employ a gradient-based update rule for q_{ϕ} :

$$\phi_{k-1}^{(i+1)} \leftarrow \phi_{k-1}^{(i)} - \eta \nabla_{\phi} \frac{1}{N} \sum_{j=1}^N [\|\epsilon_{\phi}(\mathbf{z}_{j, \lambda_j}) - \epsilon_j\|_2^2], \quad \phi_k^{(0)} \leftarrow \phi_{k-1}^{(M)}, \quad i = 0, 1, \dots, M-1. \quad (13)$$

This can be viewed as a M-estimation of ϕ . Recall that $q_T = \mathcal{K}_T q_{\phi_{k-1}}$, we can construct $\phi_k = (\phi_{k-1}, \tilde{\phi})$ to minimize the KLD, where $\tilde{\phi}$ models the transition kernel \mathcal{K}_T . Therefore, initializing q_{ϕ} to be optimized with $q_{\phi_{k-1}}$, we are effectively maximizing $\mathcal{L}(\tilde{\phi}) = \frac{1}{N} \sum_{j=1}^N \log p_{\tilde{\phi}}(\hat{\mathbf{z}}_j|\mathbf{z}_j)$,

where $\{z_j\}$ are from $q_{\phi_{k-1}}$ and $\{\hat{z}_j\}$ are from $\mathcal{K}_T q_{\phi_{k-1}}$. Let $\tilde{\phi}$ be the result of M-estimation and $\tilde{\phi}^*$ be the true target parameter. Then based on the derivation in [25], asymptotically we have

$$\sqrt{N} \left(\tilde{\phi} - \tilde{\phi}^* \right) \rightarrow \mathcal{N} \left(0, \mathcal{I} \left(\tilde{\phi}^* \right)^{-1} \right), \quad (14)$$

where $\mathcal{I} \left(\tilde{\phi}^* \right) = \mathbb{E}_{\hat{z}, z} \left[-\nabla^2 \log p_{\tilde{\phi}}(\hat{z}|z) \right]$ is the Fisher information matrix. This interpretation tells us that i) when the sample size N is large, the estimation $\tilde{\phi}$ is asymptotically unbiased, and ii) if we want to obtain the estimation $\tilde{\phi}$ with a few gradient-based updates, then the eigenvalues of the Fisher information matrix would be relatively small but non-zero. ii) suggests that $\mathcal{K}_T q_{\phi_{k-1}}$ should be significantly different from $q_{\phi_{k-1}}$, which is confirmed by [38] and our preliminary experiments, but it should not be too far away because that would require more gradient-based updates. We find that setting $T = 30$ and $M = 6$ works well in the experiments.

A.3 Further Discussion about the Learning Algorithm

For completeness, we first derive the learning gradients for updating θ .

$$\begin{aligned} \nabla_{\theta} \log p_{\theta}(\mathbf{x}) &= \frac{1}{p_{\theta}(\mathbf{x})} \nabla_{\theta} \int_{\mathbf{z}} p_{\theta}(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int_{\mathbf{z}} \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})} \nabla_{\theta} \log p_{\theta}(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\ &= \mathbb{E}_{p_{\theta}(\mathbf{z}|\mathbf{x})} \left[\nabla_{\theta} \log p_{\theta}(\mathbf{z}, \mathbf{x}) \right] \\ &= \left(\mathbb{E}_{p_{\theta}(\mathbf{z}|\mathbf{x})} \left[\nabla_{\alpha} \log p_{\alpha}(\mathbf{z}) \right], \mathbb{E}_{p_{\theta}(\mathbf{z}|\mathbf{x})} \left[\nabla_{\beta} \log p_{\beta}(\mathbf{x}|\mathbf{z}) \right] \right) \\ &= \underbrace{\left(\mathbb{E}_{p_{\theta}(\mathbf{z}|\mathbf{x})} \left[\nabla_{\alpha} f_{\alpha}(\mathbf{z}) \right] - \mathbb{E}_{p_{\alpha}(\mathbf{z})} \left[\nabla_{\alpha} f_{\alpha}(\mathbf{z}) \right] \right)}_{\delta_{\alpha}(\mathbf{x})}, \underbrace{\mathbb{E}_{p_{\theta}(\mathbf{z}|\mathbf{x})} \left[\nabla_{\beta} \log p_{\beta}(\mathbf{x}|\mathbf{z}) \right]}_{\delta_{\beta}(\mathbf{x})}. \end{aligned} \quad (15)$$

We can see that θ is estimated in a MLE-by-EM style. The learning gradient is the same as that of directly maximizing the observed data likelihood, while we need to approximate the expectations in Eq. (15). Estimating the expectations is like the E-step, and update θ with Eq. (15) is like the M-step in the EM algorithm. The proposed diffusion-based amortization brings better estimation of the expectations in the E-step, and incorporate the feedback from the M-step by running prior and posterior sampling LD as follows

$$\begin{aligned} \mathbf{z}_{t+1} &= \mathbf{z}_t + \frac{s^2}{2} \nabla_{\mathbf{z}_t} \underbrace{\left(f_{\alpha}(\mathbf{z}_t) - \frac{1}{2} \|\mathbf{z}_t\|_2^2 \right)}_{\log p_{\alpha}(\mathbf{z}_t)} + s \mathbf{w}_t, \\ \mathbf{z}_{t+1} &= \mathbf{z}_t + \frac{s^2}{2} \nabla_{\mathbf{z}_t} \underbrace{\left(-\frac{\|\mathbf{x} - g_{\beta}(\mathbf{z}_t)\|_2^2}{2\sigma^2} + f_{\alpha}(\mathbf{z}_t) - \frac{1}{2} \|\mathbf{z}_t\|_2^2 \right)}_{\log p_{\theta}(\mathbf{z}|\mathbf{x}) = \log p_{\theta}(\mathbf{x}, \mathbf{z}) + C} + s \mathbf{w}_t, \end{aligned} \quad (16)$$

to obtain training data. Here $t = 0, 1, \dots, T$. $\mathbf{z}_0 \sim q_{\phi}(\mathbf{z}|\mathbf{x})$ for posterior sampling and $\mathbf{z}_0 \sim q_{\phi}(\mathbf{z})$ for prior sampling. $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Note that we plug-in $p_{\theta}(\mathbf{x}, \mathbf{z})$ for the target distribution of posterior sampling LD. This is because given the observed data \mathbf{x} , by Bayes' rule we know that $p_{\theta}(\mathbf{z}|\mathbf{x}) \propto p_{\theta}(\mathbf{x}, \mathbf{z}) = p_{\beta}(\mathbf{x}|\mathbf{z}) p_{\alpha}(\mathbf{z})$. The whole learning iteration can be viewed as a variant of the variational EM algorithm [71].

B Network Architecture and Training Details

Architecture The energy score network f_α uses a simple fully-connected structure throughout the experiments. We describe the architecture in details in Table S1. The generator network has a simple deconvolution structure similar to DCGAN [72] shown in Table S2. The denoising diffusion model is implemented by a light-weight MLP-based U-Net [73] structure (Table S3). The encoder network to embed the observed images has a fully convolutional structure [74], as shown in Table S4.

Table S1: **Network structures of the energy score network.** LReLU denotes the Leaky ReLU activation function. The slope in Leaky ReLU is set to 0.2. For SVHN and CelebA datasets, we use $nz=100$. For the CIFAR-10 and CelebA-HQ datasets, we use $nz=128$. We use $nz=8$ for anomaly detection on the MNIST dataset, and $nz=7168$ for GAN inversion. We use $ndf=512$ for GAN inversion and $ndf=200$ for the rest experiments.

Layers	Out Size
Input: z	$nz \in \{8, 100, 128, 7168\}$
Linear, LReLU	$ndf \in \{200, 512\}$
Linear, LReLU	$ndf \in \{200, 512\}$
Linear	1

Table S2: **Network structures of the generator networks** used for the SVHN, CelebA, CIFAR-10, CelebA-HQ and MNIST (from top to bottom) datasets. For GAN inversion, we use the StyleGAN [10] structure as our generator network. ConvT(n) indicates a transposed convolutional operation with n output channels. We use $ngf=64$ for the SVHN dataset and $ngf=128$ for the rest. LReLU indicates the Leaky-ReLU activation function. The slope in Leaky ReLU is set to be 0.2.

Layers	Out Size	Stride
Input: z	1x1x100	-
4x4 ConvT($ngf \times 8$), LReLU	4x4x($ngf \times 8$)	1
4x4 ConvT($ngf \times 4$), LReLU	8x8x($ngf \times 4$)	2
4x4 ConvT($ngf \times 2$), LReLU	16x16x($ngf \times 2$)	2
4x4 ConvT(3), Tanh	32x32x3	2
Layers	Out Size	Stride
Input: z	1x1x100	-
4x4 ConvT($ngf \times 8$), LReLU	4x4x($ngf \times 8$)	1
4x4 ConvT($ngf \times 4$), LReLU	8x8x($ngf \times 4$)	2
4x4 ConvT($ngf \times 2$), LReLU	16x16x($ngf \times 2$)	2
4x4 ConvT($ngf \times 1$), LReLU	32x32x($ngf \times 1$)	2
4x4 ConvT(3), Tanh	64x64x3	2
Layers	Out Size	Stride
Input: z	1x1x128	-
8x8 ConvT($ngf \times 8$), LReLU	8x8x($ngf \times 8$)	1
4x4 ConvT($ngf \times 4$), LReLU	16x16x($ngf \times 4$)	2
4x4 ConvT($ngf \times 2$), LReLU	32x32x($ngf \times 2$)	2
3x3 ConvT(3), Tanh	32x32x3	1
Layers	Out Size	Stride
Input: z	1x1x128	-
4x4 ConvT($ngf \times 16$), LReLU	4x4x($ngf \times 16$)	1
4x4 ConvT($ngf \times 8$), LReLU	8x8x($ngf \times 8$)	2
4x4 ConvT($ngf \times 4$), LReLU	16x16x($ngf \times 4$)	2
4x4 ConvT($ngf \times 4$), LReLU	32x32x($ngf \times 4$)	2
4x4 ConvT($ngf \times 2$), LReLU	64x64x($ngf \times 2$)	2
4x4 ConvT($ngf \times 1$), LReLU	128x128x($ngf \times 1$)	2
4x4 ConvT(3), Tanh	256x256x3	2
Layers	Out Size	Stride
Input: z	1x1x8	-
7x7 ConvT($ngf \times 8$), LReLU	7x7x($ngf \times 8$)	1
4x4 ConvT($ngf \times 4$), LReLU	14x14x($ngf \times 4$)	2
4x4 ConvT($ngf \times 2$), LReLU	28x28x($ngf \times 2$)	2
3x3 ConvT(1), Tanh	28x28x1	1

Table S3: **Network structure of the denoising diffusion network.** a) We use the sinusoidal embedding to embed the time index as in [25, 8]. b) We use the learned fourier feature module [75] to embed the input z . c) The merged time embedding and context embedding is used to produce a pair of bias and scale terms to shift and scale the embedding of input z . nz is the input dimension, as in Table S1. $nemb$ is the dimension of image embedding as in Table S4.

Layers	Out size	Note
Time Embedding		
Input: t	1	time index
Sin. emb. ^a	128	
Linear, SiLU	128	
Linear	128	
Input Embedding		
Input: z	nz	
Fr. emb. ^b	$2x(nz)$	Fourier feature
Basic Block		
Input: z, z_{ctx}, z_t	$nzf, nemb, 128$	$z, ctx.$ and t emb.
Cat, SiLU	$nemb + 128$	merge $ctx.$ & t emb.
Linear, SiLU	nout	
Linear	nout	Input emb.
Scale-shift ^c	nout	scale-shift z emb. w/ merged emb.
Add z	nout	skip connection from z
Denoising Diffusion Network		
Input: z, z_{ctx}, t	$nz, nemb, 128$	Input
Embedding	$2x(nz), 128$	Input & t emb.
Basic Block	128	Encoding
Basic Block	256	
Basic Block	256	
Basic Block	256	Intermediate
Basic Block	256	Cat & Decoding
Basic Block	128	
Basic Block	nz	Output

Hyperparameters and training details As mentioned in the main text, for the posterior and prior DAMC samplers, we set the number of diffusion steps to 100. The number of iterations in Eq. (8) is set to $M = 6$ for the experiments. The LD runs $T = 30$ and $T = 60$ iterations for posterior and prior updates during training with a step size of $s = 0.1$. For test time sampling from $\mathcal{K}_{T, z_i | x_i} q_{\phi_k}(z_i | x_i)$, we set $T = 10$ for the additional LD. For test time prior sampling of LEBM with LD, we follow [22, 41] and set $T = 100$. To further stabilize the training procedure, we i) perform gradient clipping by setting the maximal gradient norm as 100, ii) use a separate target diffusion network which is the EMA of the current diffusion network to initialize the prior and posterior updates and iii) add noise-initialized prior samples for the prior updates. These set-ups are identical across different datasets.

The parameters of all the networks are initialized with the default pytorch methods [77]. We use the Adam optimizer [78] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ to train the generator network and the energy score network. We use the AdamW optimizer [79] with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and $weight_decay=1e-4$ to train the diffusion network. The initial learning rates of the generator and diffusion networks are $2e-4$, and $1e-4$ for the energy score network. The learning rates are decayed with a factor of 0.99 every 1K training iterations, with a minimum learning rate of $1e-5$. We run the experiments on a A6000 GPU with the batch size of 128. For GAN inversion, we reduce the batch size to 64. Training typically converges within 200K iterations on all the datasets.

Table S4: **Network structures of the encoder networks** used for the SVHN, CelebA, CIFAR-10, CelebA-HQ and MNIST (from top to bottom) datasets. For GAN inversion, the encoder network structure is the same as in [18]. Conv(n)Norm indicates a convolutional operation with n output channels followed by the Instance Normalization [76]. We use nif=64 and nemb=1024 for all the datasets. LReLU indicates the Leaky-ReLU activation function. The slope in Leaky ReLU is set to be 0.2.

Layers	Out Size	Stride
Input: \mathbf{x}	32x32x3	-
3x3 Conv(nif x 1)Norm, LReLU	32x32x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	16x16x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	8x8x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nemb)Norm, LReLU	1x1x(nemb)	1
Layers	Out Size	Stride
Input: \mathbf{x}	64x64x3	-
3x3 Conv(nif x 1)Norm, LReLU	64x64x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	32x32x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	16x16x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	8x8x(nif x 8)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nemb)Norm, LReLU	1x1x(nemb)	1
Layers	Out Size	Stride
Input: \mathbf{x}	32x32x3	-
3x3 Conv(nif x 1)Norm, LReLU	32x32x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	16x16x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	8x8x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nemb)Norm, LReLU	1x1x(nemb)	1
Layers	Out Size	Stride
Input: \mathbf{x}	256x256x3	-
3x3 Conv(nif x 1)Norm, LReLU	256x256x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	128x128x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	64x64x(nif x 4)	2
4x4 Conv(nif x 4)Norm, LReLU	32x32x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	16x16x(nif x 8)	2
4x4 Conv(nif x 8)Norm, LReLU	8x8x(nif x 8)	2
4x4 Conv(nif x 8)Norm, LReLU	4x4x(nif x 8)	2
4x4 Conv(nemb)Norm, LReLU	1x1x(nemb)	1
Layers	Out Size	Stride
Input: \mathbf{x}	28x28x3	-
3x3 Conv(nif x 1)Norm, LReLU	28x28x(nif x 1)	1
4x4 Conv(nif x 2)Norm, LReLU	14x14x(nif x 2)	2
4x4 Conv(nif x 4)Norm, LReLU	7x7x(nif x 4)	2
4x4 Conv(nif x 8)Norm, LReLU	3x3x(nif x 8)	2
3x3 Conv(nemb)Norm, LReLU	1x1x(nemb)	1

C Pytorch-style Pseudocode

We provide pytorch-style pseudocode to help understand the proposed method. We denote the generator network as G , the energy score network as E and the diffusion network as Q . The first page sketches the prior and posterior sampling process. The second page outlines the learning procedure.

Listing 1: Prior and posterior LD sampling.

```
def sample_langevin_prior_z(z, netE):
    s = step_size

    for i in range(n_steps):
        en = netE(z).sum()
        z_norm = 1.0 / 2.0 * torch.sum(z**2)
        z_grad = torch.autograd.grad(en + z_norm, z)[0]
        w = torch.randn_like(z)

        # Prior LD Update
        z.data = z.data - 0.5 * (s ** 2) * z_grad + s * w

    return z.detach()

def sample_langevin_posterior_z(z, x, netG, netE):
    s = step_size
    sigma_inv = 1.0 / (2.0 * sigma ** 2)

    for i in range(n_steps):
        x_hat = netG(z)
        g_log_lkhd = sigma_inv * torch.sum((x_hat - x) ** 2)

        z_n = 1.0 / 2.0 * torch.sum(z**2)
        en = netE(z).sum()

        total_en = g_log_lkhd + en + z_n
        z_grad = torch.autograd.grad(total_en, z)[0]
        w = torch.randn_like(z)

        # Posterior LD Update
        z.data = z.data - 0.5 * (s ** 2) * z_grad + s * w

    return z.detach()
```

Listing 2: Learning LEBM with DAMC.

```

for x in dataset:
    # mask for unconditional learning of DAMC
    z_mask_prob = torch.rand((len(x),), device=x.device)
    z_mask = torch.ones(len(x), device=x.device)
    z_mask[z_mask_prob < 0.2] = 0.0
    z_mask = z_mask.unsqueeze(-1)

    # draw DAMC samples
    z0 = Q(x)
    zk_pos, zk_neg = z0.detach().clone(), z0.detach().clone()

    # prior and posterior updates
    zk_pos = sample_langevin_posterior_z(
        z=zk_pos, x=x, netG=G, netE=E)
    zk_neg = sample_langevin_prior_z(
        z=torch.cat(
            [zk_neg, torch.randn_like(zk_neg)], dim=0),
        netE=E)

    # update Q
    for __ in range(6):
        Q_optimizer.zero_grad()
        Q_loss = Q.calculate_loss(
            x=x, z=zk_pos, mask=z_mask).mean()
        Q_loss.backward()
        Q_optimizer.step()

    # update G
    G_optimizer.zero_grad()
    x_hat = G(zk_pos)
    g_loss = torch.sum((x_hat - x) ** 2, dim=[1,2,3]).mean()
    g_loss.backward()
    G_optimizer.step()

    # update E
    E_optimizer.zero_grad()
    e_pos, e_neg = E(zk_pos), E(zk_neg)
    E_loss = e_pos.mean() - e_neg.mean()
    E_loss.backward()
    E_optimizer.step()

```

D Dataset and Experiment Settings

Datasets We include the following datasets to study our method: SVHN ($32 \times 32 \times 3$), CIFAR-10 ($32 \times 32 \times 3$), CelebA ($64 \times 64 \times 3$), CelebAMask-HQ ($256 \times 256 \times 3$) and MNIST ($28 \times 28 \times 1$). Following Pang et al. [22], we use the full training set of SVHN (73,257) and CIFAR-10 (50,000), and take 40,000 samples of CelebA as the training data. We take 29,500 samples from the CelebAMask-HQ dataset as the training data, and test the model on 500 held-out samples. For anomaly detection on MNIST dataset, we follow the experimental settings in [22, 41, 55, 56] and use 80% of the in-domain data to train the model. The images are scaled to $[-1, 1]$ and randomly horizontally flipped with a prob. of .5 for training.

GAN inversion settings We attempt to use the DAMC sampler for GAN version on the FFHQ ($256 \times 256 \times 3$) and LSUN-Tower ($256 \times 256 \times 3$) datasets. We take 69,500 samples from the CelebAMask-HQ dataset as the training data, and use the held-out 500 samples for testing. We follow the default data splits of the LSUN dataset.

For the LEBM-based inversion method, we train a LEBM in the $14 \times 512 = 7168$ dim. latent space. During training, we add l_2 -regularization on the energy score of LEBM to stabilize training, as suggested in [59]. For the DAMC sampler and the encoder-based inversion method [16], we generate initial posterior samples of the training data using [18] and train the DAMC sampler and the encoder-based method for 5K iterations using these samples as a warm-up step. The encoder-based method is trained by minimizing the l_2 distance between the encoder output and the target samples. After that, these two methods are trained with the default learning algorithms.

E Additional Qualitative Results

E.1 Toy Examples

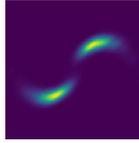


Figure S1: The 2-arm pinwheel-shaped prior distribution used in the toy example.

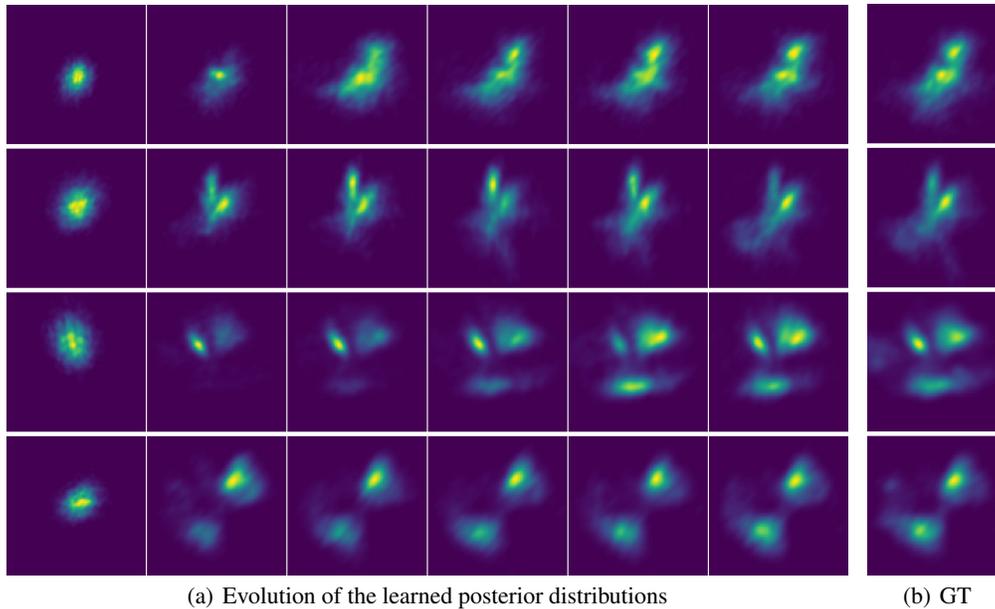


Figure S2: **Evolution of the posterior distributions learned by the DAMC sampler.** In each row, we display from left to right the evolution of the learned posterior distributions from the DAMC sampler through training iterations. The last column shows the corresponding ground-truth posterior distribution obtained by running 1K-3K steps of Langevin Dynamics until convergence for posterior sampling.

As the proof-of-concept toy examples, we implement the neural likelihood experiments following the same set-up mentioned in Section 5.1 and B.1 in [80]. We choose to use a more complex prior distribution, i.e., 2-arm pinwheel-shaped prior distribution (shown in Fig. S1) instead of a standard normal one to make sure that the true posterior distributions are multimodal. The ground-truth posterior distributions are obtained by performing long-run LD sampling until convergence. We visualize the convergence trajectory of the posterior distributions learned by our model to the ground-truth ones. In Fig. S2, we can see that our model can faithfully reproduce the ground-truth distributions with sufficient training iterations, which indicates that the learned sampler successfully amortizes the long-run sampling chain with the length of 1,000-3,000 iterations.

E.2 Generation

We provide additional generated samples from our models trained on SVHN (Fig. S3), CelebA (Fig. S4), CIFAR-10 (Fig. S5) and CelebA-HQ (Fig. S6).

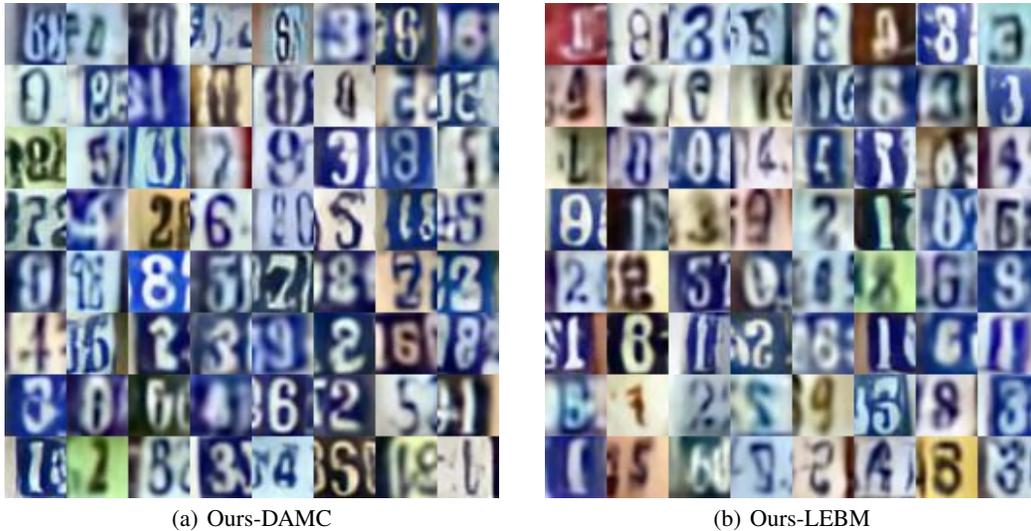


Figure S3: Samples generated from the DAMC sampler and LEBM trained on the SVHN dataset.

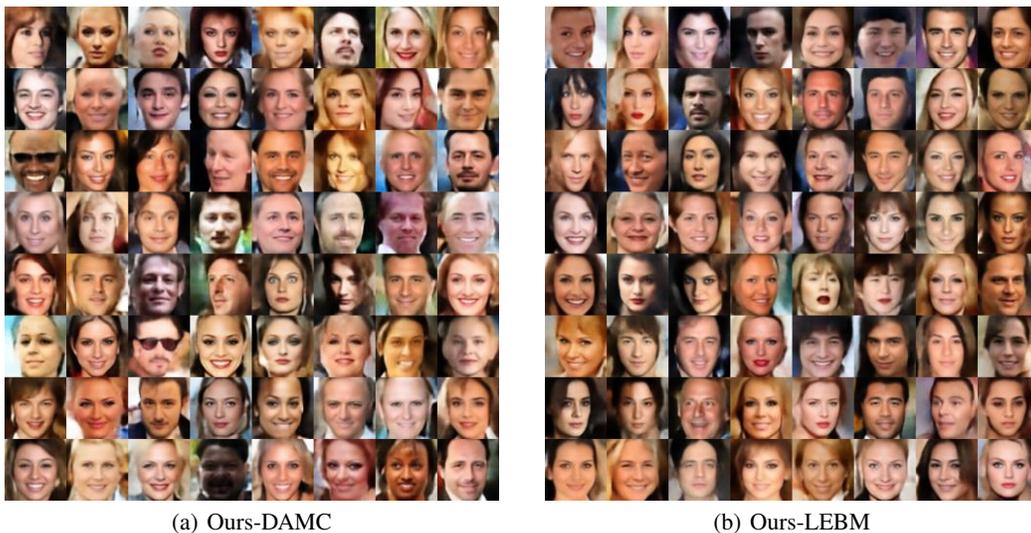


Figure S4: Samples generated from the DAMC sampler and LEBM trained on the CelebA dataset.

E.3 Reconstruction

We provide qualitative examples about the reconstruction results from our models trained on SVHN (Fig. S7), CelebA (Fig. S8), CIFAR-10 (Fig. S9) and CelebA-HQ (Fig. S10). Observed images are sampled from the testing set unseen during training.

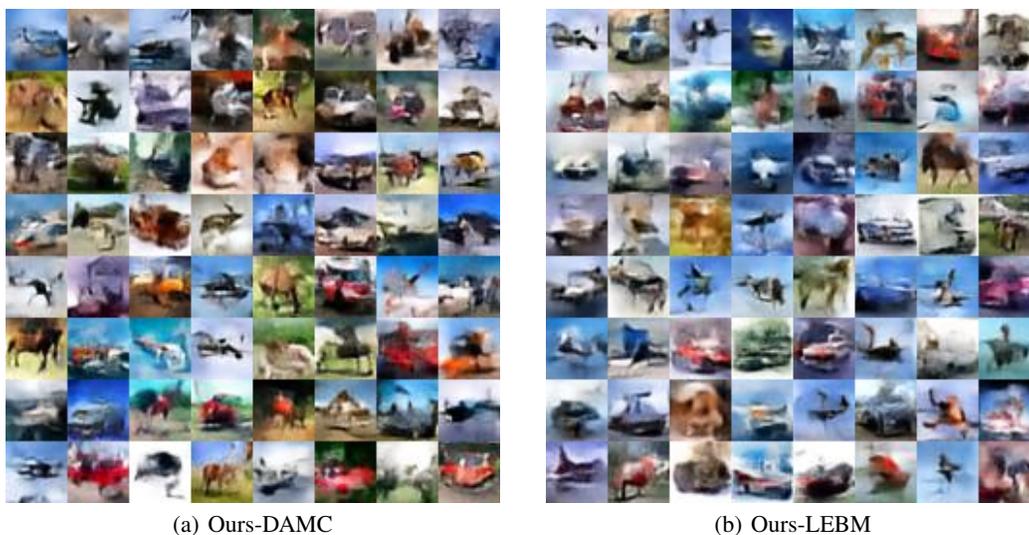


Figure S5: **Samples generated from the DAMC sampler and LEBM trained on the CIFAR-10 dataset.**

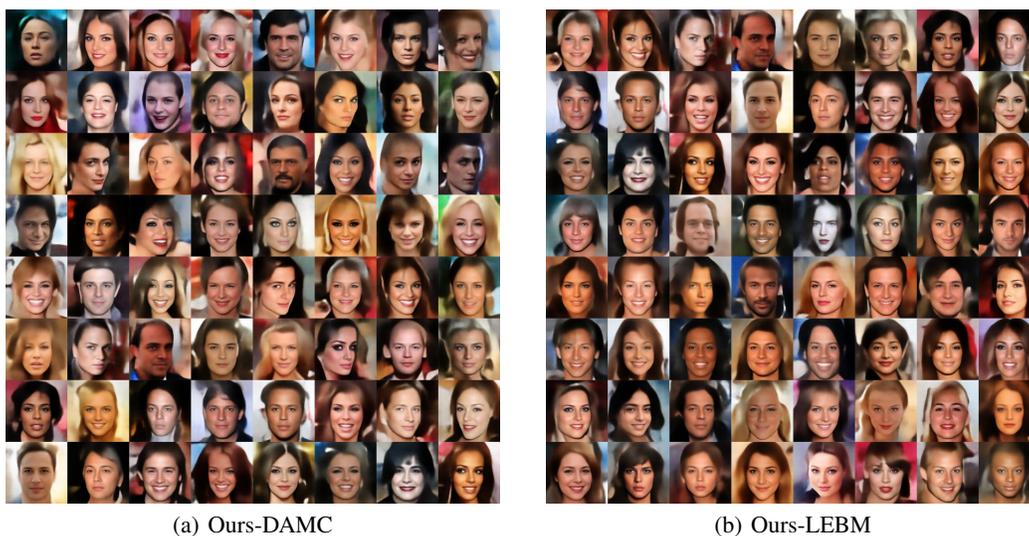


Figure S6: **Samples generated from the DAMC sampler and LEBM trained on the CelebA-HQ dataset.**

E.4 Visualization of Transitions

We provide additional visualization results of LD transitions initialized from $\mathcal{N}(0, \mathbf{I}_d)$ on SVHN (Fig. S11) and CIFAR-10 datasets (Fig. S12). For the 200-step set-up, we can see that the generation quality quickly improves by exploring the local modes with LD. For the 2500-step long-run set-up, we can see that the LD produces consistently valid results without the oversaturating issue of the long-run chain samples.



(a) Observation

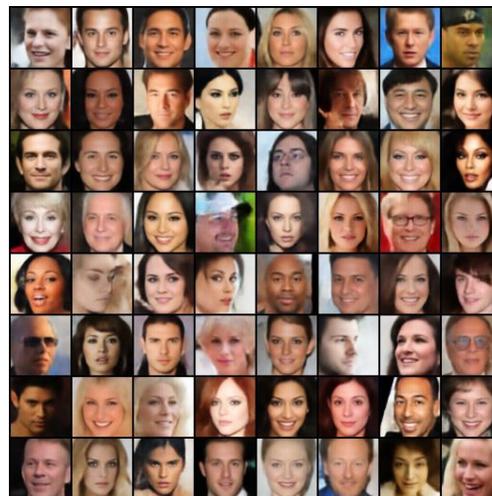


(b) Reconstruction

Figure S7: Reconstructed samples from the posterior DAMC sampler trained on the SVHN dataset.



(a) Observation



(b) Reconstruction

Figure S8: Reconstructed samples from the posterior DAMC sampler trained on the CelebA dataset.

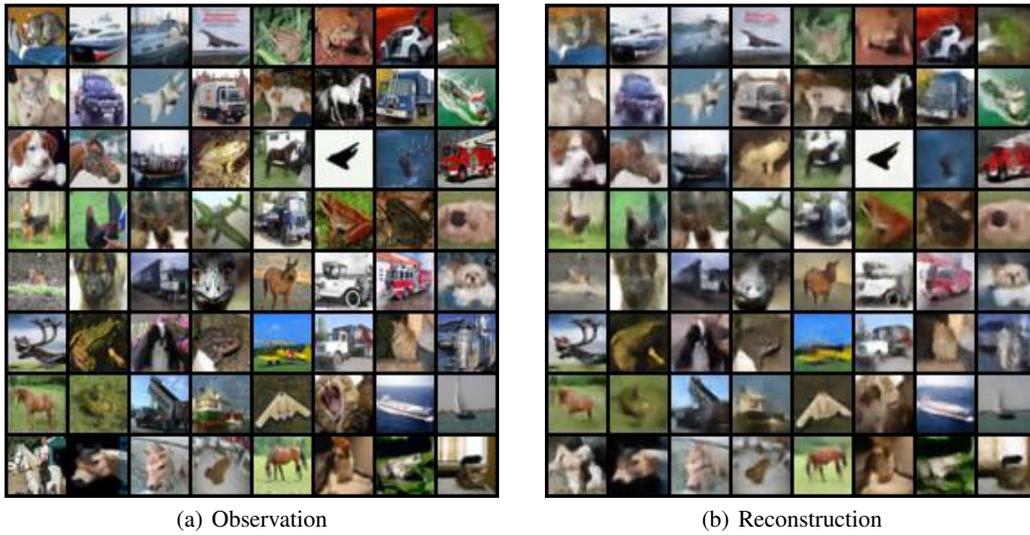


Figure S9: Reconstructed samples from the posterior DAMC sampler trained on the CIFAR-10 dataset.

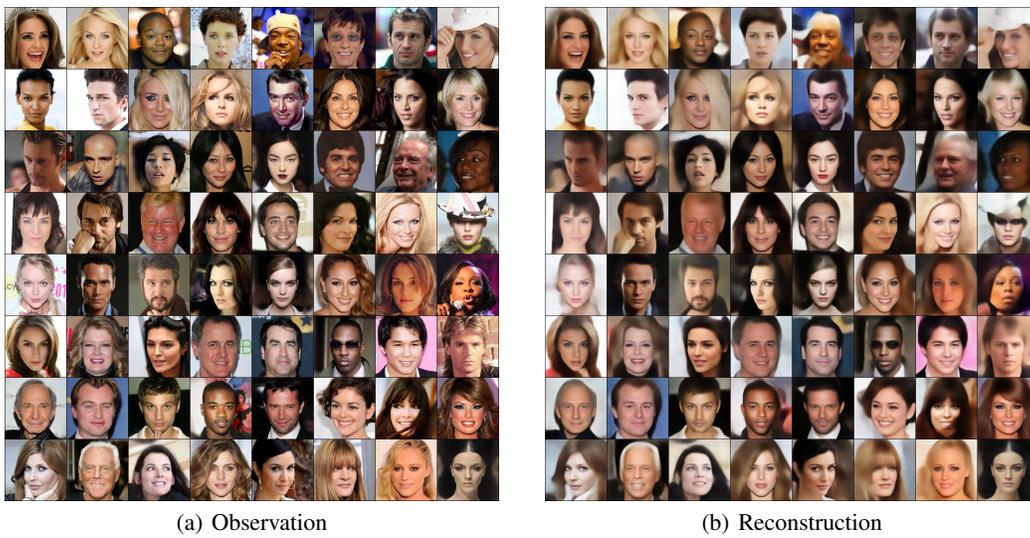


Figure S10: Reconstructed samples from the posterior DAMC sampler trained on the CelebA-HQ dataset.

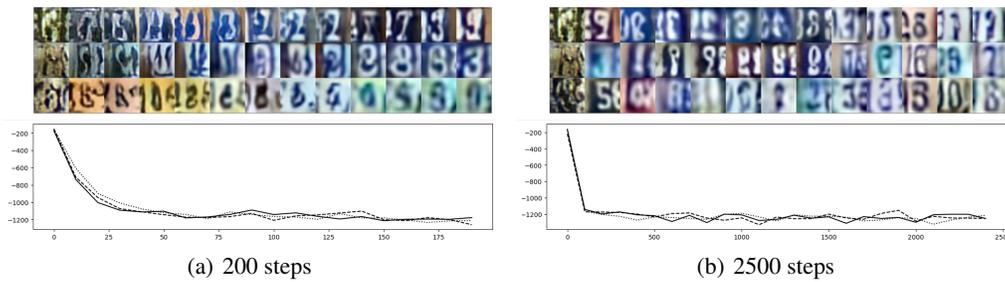


Figure S11: Transition of Markov chains initialized from $\mathcal{N}(0, \mathbf{I}_d)$ towards $p_\alpha(z)$ on SVHN. We present results by running LD for 200 and 2500 steps. In each sub-figure, the top panel displays the trajectory in the data space uniformly sampled along the chain. The bottom panel shows the energy score $f_\alpha(z)$ over the iterations.

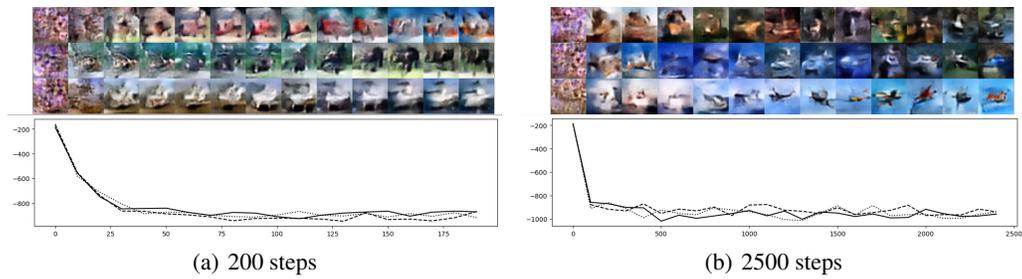


Figure S12: **Transition of Markov chains initialized from $\mathcal{N}(0, \mathbf{I}_d)$ towards $p_\alpha(z)$ on CIFAR-10.** We present results by running LD for 200 and 2500 steps. In each sub-figure, the top panel displays the trajectory in the data space uniformly sampled along the chain. The bottom panel shows the energy score $f_\alpha(z)$ over the iterations.

F Additional Quantitative Results

F.1 Learning DAMC in the Latent Space of Other DLVMs

We have compared our method with directly training diffusion models in the latent space. In the NALR-LEBM column in Table 4, we compare our method with learning a diffusion model in the pre-trained energy-based latent space on CIFAR-10 dataset. We further add experiments of learning this model in the pre-trained VAE [1] and ABP [51] model latent spaces summarized in Table S5. We can see that our method greatly outperforms these baselines using the same network architectures.

Table S5: **Further baseline results for learning DAMC on CIFAR-10 dataset.** These models are implemented using the same encoder and decoder network architectures for fair comparison

	VAE	ABP	NALR	Ours
FID	102.54	69.93	64.38	57.52
MSE	0.036	0.017	0.016	0.015

F.2 Further Ablation Studies

We conduct more ablation studies to inspect the impact of some hyper-parameters. For prior LD sampling, we follow [22] and set the sampling step as 60 throughout the experiments for fair comparison. We further add experiments on CIFAR-10 for training our model with different posterior LD sampling steps T and with different capacities of the amortizer q_ϕ , summarized in Table S6. We observe that larger posterior sampling steps and larger model capacities for training brings marginal improvement compared with our default set-up. Less sampling steps and lower model capacities, however, may have negative impact on the performances.

(a) Posterior sampling steps T				(b) Model capacity of q_ϕ					
	$T=10$	$T=30$	$T=50$		$f=1/4$	$f=1/2$	$f=1$	$f=2$	$f=4$
FID	74.20	57.72	57.03	FID	116.28	80.28	57.52	57.82	57.56
MSE	0.016	0.015	0.015	MSE	0.017	0.016	0.015	0.015	0.015

Table S6: **Ablation Studies for the choice of Langevin steps and model capacity.** We highlight the results of our set-up reported in the main text. T is the posterior sampling step. f stands for the factor for the model capacity, *e.g.*, $f=2$ means 2x the size of the original model.

G Further Discussion

G.1 Limitations

We mentioned in the main text that one potential disadvantage of our method is its parameter inefficiency for introducing an extra DDPM. Although fortunately, our models are in the latent space so the network is lightweight. To be specific, on SVHN, CelebA, CIFAR-10 and CelebA-HQ datasets the number of parameters in the diffusion network is around 10% of those in the generator.

Another issue is the time efficiency. We mentioned in the main text that the time efficiency for sampling is competitive. With the batch size of 64, on these datasets the DAMC prior sampling takes 0.3s, while 100 steps of short-run LD with LEBM takes 0.2s. The DAMC posterior sampling takes 1.0s, while LEBM takes 8.0s. However, during training we need to run 30 steps of posterior LD sampling and 60 steps of prior LD sampling in each training iteration. We observe that the proposed learning method takes 15.2 minutes per training epoch, while the short-run LD-based learning method takes 14.8 minutes per epoch. These methods are slower than the VAE-base method, which takes 5.5 minutes for an training epoch. We can see that the time efficiency for training is generally bottlenecked by the LD sampling process, and could be improved in future works.

G.2 Broader Impacts

Generative models could be misused for disinformation or faking profiles. Our work focuses on the learning algorithm of energy-based prior model. Though we consider our work to be foundational and not tied to particular applications or deployments, it is possible that more powerful energy-based generative models augmented with this method may be used maliciously. Work on the reliable detection of synthetic content could be important to address such harms from generative models.