Learning Latent Energy-Based Models via Interacting Particle Langevin Dynamics

Joanna Marks* **Tim Y. J. Wang*** **O. Deniz Akyildiz**

Department of Mathematics, Imperial College London

Abstract

We develop interacting particle algorithms for learning latent variable models with energy-based priors. To do so, we leverage recent developments in particle-based methods for solving maximum marginal likelihood estimation (MMLE) problems. Specifically, we provide a continuous-time framework for learning latent energy-based models, by defining stochastic differential equations (SDEs) that provably solve the MMLE problem. We obtain a practical algorithm as a discretisation of these SDEs and provide theoretical guarantees for the convergence of the proposed algorithm. Finally, we empirically validate the effectiveness of our method on synthetic and image datasets and demonstrate that using a particle based approach offers significant improvement in computational efficiency.

1 INTRODUCTION

Energy-Based Models (EBMs) offer a flexible framework for statistical and generative modelling by learning models of the form $p_\theta \propto e^{-U_\theta}$ from data. The energy function U_θ can be flexibly parameterised, enabling EBMs to approximate any probability density (Atchadé et al., 2023). Once the parameters θ are learned for a given dataset, these models support sampling and inference, allowing their use in a wide range of applications, including generation across modalities (Deng et al., 2021; Gladstone et al., 2026), robust classification (Grathwohl et al., 2020), anomaly detection (Zhai et al., 2016), likelihood-based evaluation (Du and Mordatch, 2019), simulation-based inference (Glaser et al., 2022), and compositional generation (Du et al., 2020, 2023).

Maximum Likelihood Estimation (MLE) procedures to learn EBMs remain highly challenging to implement since common methods require samples from the model p_θ to perform parameter updates, typically drawn using Markov chain Monte Carlo (MCMC) methods (Hinton, 2002; Song and Kingma, 2021). In high dimensional settings, these often suffer from slow mixing times. Moreover, real-world data often concentrates on a low-dimensional manifold (Bengio et al., 2013; Whiteley et al., 2025), meaning that the learning task might be ill-defined (Loaiza-Ganem et al., 2024).

A natural way to address both challenges within the EBM framework is to learn a low-dimensional latent representation of the data and model the encoded data as an EBM, giving rise to Latent Energy-Based Model (LEBM) (Pang et al., 2020). This model comprises of a likelihood (decoder) and a prior in the (lower-dimensional) latent space, which is parameterised as an EBM. The latent formulation, however, introduces new challenges in the MLE-training of such models. First, the model parameters must be learned by maximising the *marginal likelihood* of the observed data, which involves integrating out the latent variables, a typical setting known as Maximum Marginal Likelihood Estimation (MMLE) (Dempster et al., 1977). Second, the normalising constant of the prior distribution is generally intractable, resulting in a "doubly-intractable" setting.

A notable method to tackle this problem is built by Pang et al. (2020), who propose an MCMC-based approach to tackle both the intractable posterior and the intractable normalising constant of the prior. This procedure can be seen as an Expectation-Maximization (EM)-like procedure, in a similar spirit to De Bortoli et al. (2021). Adapted to the EBM setting, these methods involve (double) MCMC loops to sample from the posterior and/or the prior at each iteration of the training, which introduce significant computational overhead and complicate the non-asymptotic analysis.

In this work, we introduce a novel, diffusion-based approach to solve the MMLE problem for the EBM prior latent variable model. By leveraging the rich connection between Langevin dynamics and optimisation (Raginsky et al., 2017; Zhang et al., 2023), we develop a training algorithm based on an interacting particle system that simultaneously estim-

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:4299–4322.

ates the parameters and the posterior distribution through solving a system of SDEs.

Contributions. Our contributions can be summarised as follows:

- In Section 3, we introduce a continuous-time, SDE-based framework for training latent EBMs. This interacting particle system is inspired by the Interacting Particle Langevin Algorithm (IPLA) (Akyildiz et al., 2025) and adapted to the LEBM setting. By construction, the θ-marginal of the stationary measure of the proposed system concentrates on the maximisers of the marginal likelihood with increasing N under very mild assumptions. We then develop an algorithm by discretising SDE systems that target the MMLE solution, termed Energy-Based Interacting Particle Langevin Algorithm (EBIPLA), given in Algorithm 1.
- Section 4 provides a non-asymptotic analysis for our method under log-concavity and smoothness assumptions. We show that, under these assumptions and for a suitable discretisation step size, the parameter estimates $(\theta_k)_{k\geq 0}$ generated by EBIPLA approach the empirical maximiser $\theta_\star$ as the iterations k and number of particles N increase. These results constitute the first convergence bounds for training LEBMs.
- Furthermore, by introducing a novel rescaling, we derive convergence bounds that explicitly depend on the number of data points M and show that the distance of the parameter estimates to the empirical maximiser $\theta_\star$ decreases as M increases, which to the best of our knowledge has not been addressed in the available IPLA literature. This provides a theoretical justification for why particle-based methods remain effective even when using a relatively small number of particles on large datasets.
- In Section 6, we empirically validate the effectiveness of our method on synthetic datasets and image generation tasks, showing that it achieves competitive performance among relevant baselines. We also demonstrate that using a particle based approach offers significant improvement in computational efficiency.

Notation. We use $\mathcal{O}(\cdot)$ to denote upper bounds up to constant factors, and $\widetilde{\mathcal{O}}(\cdot)$ to denote the same while suppressing polylogarithmic terms. For a random variable X, the notation $\mathcal{L}(X)$ denotes the law (distribution) of X while $\sigma(X)$ denotes the σ-algebra generated by X. For any two probability distributions μ and ν, we define the Wasserstein distance:

$$W_2(\mu,\nu) := \left(\inf_{\gamma\in\Gamma(\mu,\nu)} \int \|x-y\|^2 \,\mathrm{d}\gamma(x,y)\right)^{1/2},$$

where $\Gamma(\mu,\nu)$ is the set of all couplings of μ and ν. The notation δ_x denotes the Dirac measure at x. Where appropriate, we use the notation $x^{1:M} = \{x^m\}_{m=1}^M$ and $x^{1:M,1:N} = \{x^{m,n}\}_{m=1,n=1}^{M,N}$ to denote collections of variables.

2 BACKGROUND

We start by defining a real-valued latent energy-based probabilistic model (Pang et al., 2020; Yu et al., 2023):

$$p_\theta(x,y) = p_\alpha(x)p_\beta(y \mid x), \tag{1}$$

where $\theta = (\alpha,\beta) \in \mathbb{R}^{d_\alpha+d_\beta} = \mathbb{R}^{d_\theta}$ and $p_\beta(y|x) = \mathcal{N}\big(y; g_\beta(x), \sigma^2 I\big)$ is an isotropic Gaussian distribution. Here, $g_\beta : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ denotes a generator function parametrised by β that maps from the latent space to the ambient space. The prior $p_\alpha(x)$ is modelled as an EBM:

$$p_\alpha(x) = \frac{1}{Z(\alpha)} e^{-U_\alpha(x)}, \tag{2}$$

where $Z(\alpha) = \int e^{-U_\alpha(x)}\mathrm{d}x$ is the α-dependent normalising constant and we assume $Z(\alpha) < \infty$ for all $\alpha \in \mathbb{R}^{d_\alpha}$ to ensure the model is well defined. When $U_\alpha(x)$ is chosen to be a neural network, the EBM can capture meaningful latent structure.

2.1 MAXIMUM MARGINAL LIKELIHOOD ESTIMATION

Given a dataset $\{y_m\}_{m=1}^M \sim p_{\text{data}}$, we are interested in learning the parameters θ of the model by maximising the marginal likelihood of the observed data. This problem is termed the MMLE problem. Specifically, we want to solve

$$\theta_\star^{\text{pop}} \in \arg\max_{\theta\in\mathbb{R}^{d_\theta}} \ell(\theta), \qquad \ell(\theta) := \mathbb{E}_{p_{\text{data}}}\big[\log p_\theta(Y)\big].$$

where $p_\theta(y) = \int p_\alpha(x)p_\beta(y|x)\mathrm{d}x$.

Since we do not have access to the true data distribution p_{data}, we can approximate the expectation over p_{data} by an empirical measure $p_{\text{data}}^M = (1/M)\sum_{m=1}^M \delta_{y_m}(\mathrm{d}y)$ which yields the empirical loss function

$$\ell_M(\theta) = \frac{1}{M}\sum_{m=1}^M \log p_\theta(y_m) \tag{3}$$

converting the population MMLE problem to the empirical MMLE problem:

$$\theta_\star \in \arg\max_{\theta\in\mathbb{R}^{d_\theta}} \frac{1}{M}\sum_{m=1}^M \log p_\theta(y_m), \tag{4}$$

where M denotes the number of data-points. Our main goal in this paper is to solve (4) efficiently and to provide theoretical guarantees for the obtained solution.

2.2 OPTIMISATION VIA LANGEVIN DYNAMICS

Following Akyildiz et al. (2025), in order to maximise $\ell(\theta)$, we will exploit stochastic dynamics whose long-run behaviour naturally concentrates around the maximisers. A classical choice is *Langevin dynamics*, given by the SDE (Raginsky et al., 2017; Zhang et al., 2023)

$$\mathrm{d}\theta_t = \nabla_\theta \ell(\theta_t)\,\mathrm{d}t + \sqrt{2/\eta}\,\mathrm{d}B_t, \tag{5}$$

where $(B_t)_{t\geq 0}$ is a standard d_θ-dimensional Brownian motion and $\eta > 0$ is an inverse temperature parameter. Under standard regularity assumptions, this SDE admits invariant density $\pi_\eta(\theta) \propto \exp(\eta\ell(\theta))$. Moreover, $\pi_\eta(\theta)$ concentrates on the set of global maximisers of ℓ as $\eta \to \infty$ (Hwang, 1980).

3 THE ALGORITHM

In practice, simulating Langevin dynamics given in (5) requires repeated evaluations of $\nabla_\theta \ell(\theta)$; we therefore turn to how this gradient can be estimated in the marginal likelihood setting. We note that, under standard regularity conditions allowing differentiation under the integral sign, by using Fisher's identity (Douc et al., 2014, Proposition D.4) for $\nabla_\theta \log p_\theta(y)$, we have

$$\nabla_\theta \ell(\theta) = \mathbb{E}_{p_{\text{data}}} \mathbb{E}_{p_\theta(\cdot|Y)} \left[\nabla \log p_\theta(\cdot, Y)\right]. \tag{6}$$

Replacing p_{data} with the empirical distribution p^M_{data}, we can estimate the gradient in (6) with

$$\nabla_\theta \ell_M(\theta) = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{p_\theta(\cdot|y_m)} \left[\nabla \log p_\theta(\cdot, y_m)\right].$$

This gradient remains intractable as $p_\theta(\cdot|y_m)$ is a posterior distribution which is typically not available in closed form. A typical workaround is to use MCMC to sample from $p_\theta(\cdot|y_m)$ (Pang et al., 2020; De Bortoli et al., 2021). We will instead follow Akyildiz et al. (2025), Kuntz et al. (2023) and use a particle approach where a set of N particles will target the posterior $p_\theta(\cdot|y_m)$ for each data point y_m. We denote these particles as $X_t^{m,n}$, where $m \in \{1,\dots,M\}$ indexes the data point and $n \in \{1,\dots,N\}$ indexes the particle.

3.1 AN INTERACTING PARTICLE ALGORITHM

For each fixed $y \in \mathbb{R}^{d_y}$, define $\phi_y : \mathbb{R}^{d_\theta} \times \mathbb{R}^{d_x} \to \mathbb{R}$ by $\phi_y(\theta, x) = -\log p_\theta(x, y)$. For each data point y_m, we evolve N particles, denoted $X_t^{m,n}$ for $n = 1,\dots,N$, according to Langevin dynamics targeting the posterior $p_\theta(\cdot|y_m)$. Simultaneously, we use the particles to estimate the gradient $\nabla_\theta \ell_M(\theta)$ and evolve θ according to Langevin dynamics targeting the maximisers of $\ell_M(\theta)$. This can be ensured by setting the noise scaling in (5) to be $\eta = MN$. With this choice, as evident in our proof the associated joint Langevin system has a θ-marginal proportional to $\exp(MN\ell_M(\theta))$ which concentrates around the maximisers of $\ell_M(\theta)$ with increasing number of particles N.

Let us denote the joint empirical measure of the particles and data points as $\mathsf{p}_t^{MN} = (1/MN)\sum_{m=1}^{M}\sum_{n=1}^{N} \delta_{(y_m, X_t^{m,n})}$. Using this, we propose to use the following SDE system:

$$\mathrm{d}\theta_t = -\mathbb{E}_{\mathsf{p}_t^{MN}}\left[\nabla_\theta \phi_Y(\theta_t, X)\right]\mathrm{d}t + \sqrt{\frac{2}{MN}}\mathrm{d}B_t, \tag{7}$$

$$\mathrm{d}X_t^{m,n} = -\nabla_x \phi_{y_m}(\theta_t, X_t^{m,n})\mathrm{d}t + \sqrt{2}\mathrm{d}B_t^{m,n}, \tag{8}$$

where $(B_t)_{t\geq 0}$ is a d_θ-dimensional Brownian motion and $(B_t^{m,n})_{t\geq 0}$ for $m \in \{1,\dots,M\}$, $n \in \{1,\dots N\}$ denote a family of d_x-dimensional independent standard Brownian motions. This is a generalisation of the interacting particle system introduced in Akyildiz et al. (2025) to the M data point setting. One can observe that Eq. (7) aligns with the global optimisation setting given in Eq. (5) with the choice of $\eta = MN$.

In order to construct an implementable algorithm, we discretise the SDEs (7)–(8) using an Euler-Maruyama discretisation. Given a (possibly random) initial value θ_0 and particles $X_0^{1:M,1:N}$, a step size $h > 0$, the discretised system for $k \geq 0$ is given by

$$\theta_{k+1} = \theta_k - h\mathbb{E}_{\mathsf{p}_k^{MN}}\left[\nabla_\theta \phi_Y(\theta_k, X)\right] + \sqrt{\frac{2h}{MN}}W_k, \tag{9}$$

$$X_{k+1}^{m,n} = X_k^{m,n} - h\nabla_x \phi_{y_m}(\theta_k, X_k^{m,n}) + \sqrt{2h}W_k^{m,n}, \tag{10}$$

where $\mathsf{p}_k^{MN} = (1/MN)\sum_{m=1}^{M}\sum_{n=1}^{N}\delta_{(y_m, X_k^{m,n})}$ and $(W_k)_{k\geq 0}$ and $(W_k^{m,n})_{k\geq 0}$ for all $m \in \{1,\dots,M\}$, $n \in \{1,\dots N\}$ denote independent standard Gaussian random variables of appropriate dimension[1]. Specifically, the drift coefficient in (9) can be explicitly written as

$$\mathbb{E}_{\mathsf{p}_k^{MN}}\left[\nabla_\theta \phi_Y(\theta_k, X)\right] = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\nabla_\theta \phi_{y_m}(\theta_k, X_k^{m,n}).$$

To connect this to the estimation task for our model given in (1), let us define the function $V_\beta(x, y) = -\log p_\beta(y|x)$. Given the energy function $U_\alpha : \mathbb{R}^{d_x} \to \mathbb{R}$, the gradient with respect to $\theta = (\alpha, \beta)$ is made explicit to clarify the implementation (for the derivation, see Appendix A). We

[1] With a slight abuse of notation, we use the same notation for the continuous time processes $(\theta_t, X_t^{m,n})$ and their discretisation $(\theta_k, X_k^{m,n})$.

can thus rewrite the system in Eqs. (9)–(10) as

$$\alpha_{k+1} = \alpha_k - \frac{h}{MN}\sum_{n=1}^{N}\sum_{m=1}^{M}\nabla_\alpha U_{\alpha_k}(X_k^{m,n}) + h\mathbb{E}_{p_{\alpha_k}(x)}[\nabla_\alpha U_{\alpha_k}(x)] + \sqrt{\frac{2h}{MN}}W_k^\alpha, \quad (11)$$

$$\beta_{k+1} = \beta_k - \frac{h}{MN}\sum_{n=1}^{N}\sum_{m=1}^{M}\nabla_\beta V_{\beta_k}(X_k^{m,n}, y_m) + \sqrt{\frac{2h}{MN}}W_k^\beta, \quad (12)$$

$$X_{k+1}^{m,n} = X_k^{m,n} - h\nabla_x U_{\alpha_k}(X_k^{m,n}) - h\nabla_x V_{\beta_k}(X_k^{m,n}, y_m) + \sqrt{2h}W_k^{m,n} \quad (13)$$

for all $k \in \{0, \dots, K-1\}$, where $(W_k^\alpha)_{k\geq 0}$, $(W_k^\beta)_{k\geq 0}$, and $(W_k^{m,n})_{k\geq 0}$ denote independent standard Gaussian random variables of appropriate dimension. Note that the update for α in (11) involves an expectation with respect to the prior, i.e. $\mathbb{E}_{p_{\alpha_k}(x)}[\nabla_\alpha U_{\alpha_k}(x)]$, which originates from the normalising constant Z_α (see Appendix A for details). As this expectation is generally intractable, it must also be approximated.

To approximate the expectation in Eq. (11), we resort to a short-run MCMC-based approach, as typically done in training of EBMs (Song and Kingma, 2021) and LEBMs (Pang et al., 2020). For each data point, we generate approximate prior samples using a short-run Unadjusted Langevin Algorithm (ULA) chain targeting the prior distribution p_α. Specifically, for each data point y_m and a fixed parameter value α_k, we initialise the chain at $\hat{X}_{k,0}^m \sim p_0$, where $p_0 = \mathcal{N}(0, I_{d_x})$, and evolve it for a fixed number of steps J with a step-size $\gamma > 0$:

$$\hat{X}_{k,j+1}^m = \hat{X}_{k,j}^m - \gamma\nabla_x U_{\alpha_k}(\hat{X}_{k,j}^m) + \sqrt{2\gamma}W_{k,j}^m, \quad (14)$$

where $(W_{k,j}^m)_{j,k\geq 0}$ for $m \in \{1, \dots, M\}$ denote independent standard d_x-dimensional Gaussian random variables. Then for every $k \geq 0$ and for each data point y_m, we run an MCMC chain for J steps and we obtain the following system of equations

$$\tilde{\alpha}_{k+1} = \tilde{\alpha}_k - \frac{h}{MN}\sum_{n=1}^{N}\sum_{m=1}^{M}\nabla_\alpha U_{\tilde{\alpha}_k}(\tilde{X}_k^{m,n}) + \frac{h}{M}\sum_{m=1}^{M}\nabla_\alpha U_{\tilde{\alpha}_k}(\hat{X}_{k,J}^m) + \sqrt{\tfrac{2h}{MN}}\,\tilde{W}_k^\alpha, \quad (15)$$

$$\tilde{\beta}_{k+1} = \tilde{\beta}_k - \frac{h}{MN}\sum_{n=1}^{N}\sum_{m=1}^{M}\nabla_\beta V_{\tilde{\beta}_k}(\tilde{X}_k^{m,n}, y_m) + \sqrt{\tfrac{2h}{MN}}\,\tilde{W}_k^\beta, \quad (16)$$

$$\tilde{X}_{k+1}^{m,n} = \tilde{X}_k^{m,n} - h\nabla_x U_{\tilde{\alpha}_k}(\tilde{X}_k^{m,n}) - h\nabla_x V_{\tilde{\beta}_k}(\tilde{X}_k^{m,n}, y_m) + \sqrt{2h}\,\tilde{W}_k^{m,n}, \quad (17)$$

Algorithm 1 Energy-Based Interacting Particle Langevin Algorithm (Full version)

Require:
- $\{y_m\}_{m=1}^M, M$ # Dataset and its size
- K # Number of iterations
- N # Particles per data point
- J # Prior sampling steps
- $h > 0$ # Step-size for discretisation
- $\gamma > 0$ # Step-size for prior sampling
- $\tilde{\theta}_0 = (\tilde{\alpha}_0, \tilde{\beta}_0)$ # Initial parameters values
- $\{\tilde{X}_0^{m,n}\}_{m,n=1}^{M,N}$ # Initial particles

for $k = 0$ to $K-1$ **do**
  Sample $\hat{X}_{k,J}^m$ using J steps of (14) with $\tilde{\alpha}_k$ for each $m \in \{1, \dots, M\}$.
  Update $\tilde{\alpha}_{k+1}$ according to Eq. (15)
  Update $\tilde{\beta}_{k+1}$ according to Eq. (16)
  for $(m, n) \in \{1, \dots, M\} \times \{1, \dots, N\}$ **do**
    Update $\tilde{X}_{k+1}^{m,n}$ according to Eq. (17)
  end for
end for

where $(\tilde{W}_k^\alpha)_{k\geq 0}$, $(\tilde{W}_k^\beta)_{k\geq 0}$, and $(\tilde{W}_k^{m,n})_{k\geq 0}$ denote independent standard Gaussian random variables of appropriate dimension. The complete algorithm is summarised in Algorithm 1.

3.2 PRACTICAL IMPLEMENTATION DETAILS

In practice, we use several standard techniques to ensure stable and efficient training (see Appendix C.1 for details). To optimise the parameters (α, β), we use the Adam optimiser (Kingma and Ba, 2015), which uses adaptive step sizes and momentum and has become a standard choice for training deep models due to its robustness across different scales of parameters. To handle large datasets efficiently, we rely on mini-batching: instead of using the full dataset at each iteration, we approximate the gradients using randomly sampled subsets of data (Robbins and Monro, 1951). This reduces computational cost and gives an unbiased estimator of the corresponding finite-sum gradient. Finally, since mini-batching implies that the decoder's parameters are updated per batch rather than per full dataset pass, we introduce a suitable noise addition scheme for the posterior particles to approximately match the full-data Euler-Maruyama discretisation over one epoch. The complete algorithm and further details are provided in Appendix C.1.

4 NONASYMPTOTIC ANALYSIS

In this section, we study convergence of EBIPLA to the global maximisers of the marginal log-likelihood under strong convexity and L-smoothness assumptions. We focus on how convergence depends on both the number of particles N and the number of data points M. We first consider the case where the prior expectation $\mathbb{E}_{p_{\alpha_k}}[\nabla_\alpha U_{\alpha_k}(x)]$ appearing in Eq. (11) is available exactly, and then handle the inexact setting where it is replaced by a biased MCMC estimate.

4.1 ASSUMPTIONS

We first assume the negative joint log-likelihood is strongly convex and L-smooth for every data point.

A1. *(Strong convexity) Let $v = (\theta, x)$ and $v' = (\theta', x')$. We assume that there exists $\mu > 0$ such that*

$$\langle \nabla \phi_y(v) - \nabla \phi_y(v'), v - v' \rangle \geq \mu \|v - v'\|^2$$

for all $y \in \mathbb{R}^{d_y}$ and $v, v' \in \mathbb{R}^{d_\theta} \times \mathbb{R}^{d_x}$.

Remark 1 (Log-concavity of the marginal likelihood)**.** Note that $\phi_{y_m}(\theta, x) = -\log p_\theta(x, y_m)$ is the negative joint log-likelihood, thus **A**1 is a joint log-concavity assumption that holds for $p_\theta(x, y_m)$ for every m.Since $\phi_{y_m}(\theta, \cdot)$ is μ-strongly convex, $p_\theta(y_m) < \infty$ for all θ, and by Prékopa's theorem it can be shown that $p_\theta(y_m)$ is μ-strongly log-concave in θ. This then implies that $\ell_M(\theta)$ is also μ-strongly concave and hence has a unique maximiser. ◇

A2. *(L-smoothness) Let $v = (\theta, x)$. We assume that there exists $L > 0$ such that*

$$\|\nabla \phi_y(v) - \nabla \phi_y(v')\| \leq L \|v - v'\|$$

for all $y \in \mathbb{R}^{d_y}$ and $v, v' \in \mathbb{R}^{d_\theta} \times \mathbb{R}^{d_x}$

4.2 CONVERGENCE WITH EXACT GRADIENT

We first analyse the simplified setting in Eqs. (9)–(10). In other words, we assume that, equivalently, in Eqs. (11)–(13)), the expectation w.r.t. p_{α_k} is known. This allows us to isolate the core convergence behaviour of the particle-based updates as can be seen below.

Theorem 1. *Suppose **A**1, **A**2 hold. Let θ_k be the parameter marginal generated by iterates (9)–(10) (equivalently, (11)–(13)), then given a step size $0 < h \leq 2/(\mu + L)$, the following holds*

$$\mathbb{E}\left[\|\theta_k - \theta_\star\|^2\right]^{1/2} \leq (1 - \mu h)^k C_0 + C_1 h^{1/2} + \frac{C_2}{\sqrt{MN}},$$

where C_0 is an explicit constant which depends on the initial law of the system and

$$C_1 = 1.65 \frac{L}{\mu} \sqrt{\frac{d_\theta + M N d_x}{MN}}, \quad C_2 = \sqrt{\frac{d_\theta}{\mu}},$$

and $\theta_\star$ denotes the unique maximiser of ℓ_M.

Proof. A full proof can be found in Appendix B.1. □

Remark 2 (Convergence rate)**.** Theorem 1 implies that, for sufficiently large N and k and sufficiently small h, the iterates of Eqs. (11)–(13) can be made arbitrarily close to $\theta_\star$. More precisely for a fixed number of data points M and a target accuracy $\varepsilon > 0$ choosing $N = \mathcal{O}(\varepsilon^{-2} M^{-1} d_\theta)$ makes the last term of the bound in Theorem 1 of order $\mathcal{O}(\varepsilon)$. Next, setting $h = \mathcal{O}(\varepsilon^2 d_x^{-1})$ makes the middle term of order $\mathcal{O}(\varepsilon)$. Finally setting $k \geq \tilde{\mathcal{O}}(d_x \varepsilon^{-2})$ ensures that the first term is $\mathcal{O}(\varepsilon)$. With these choices, we obtain $\mathbb{E}[\|\theta_k - \theta_\star\|^2]^{1/2} \leq \mathcal{O}(\varepsilon)$. Note that since $N = \mathcal{O}(M^{-1}\varepsilon^{-2})$, for large datasets the bound can be made small with small N. ◇

Below, we provide a sketch of the proof. We start by noting that $\mathbb{E}[\|\theta_k - \theta_\star\|^2]^{1/2} = W_2(\mathcal{L}(\theta_k), \delta_{\theta_\star})$ since $\delta_{\theta_\star}$ is a Dirac measure (Santambrogio, 2015, Section 1.4). Then via an application of the triangle inequality, we obtain:

$$\begin{aligned} \mathbb{E}\left[\|\theta_k - \theta_\star\|^2\right]^{1/2} &= W_2\left(\mathcal{L}\left(\theta_k\right), \delta_{\theta_\star}\right) \\ &\leq \underbrace{W_2\left(\pi_\Theta, \delta_{\theta_\star}\right)}_{\text{concentration}} + \underbrace{W_2\left(\mathcal{L}\left(\theta_k\right), \pi_\Theta\right)}_{\text{convergence}}. \end{aligned}$$

where π_Θ denotes the θ marginal of the invariant measure of the analysed system of SDEs. We establish concentration using Altschuler and Chewi (2024, Lemma A.8) and convergence by adapting Dalalyan and Karagulyan (2019, Theorem 1) for our system.

Remark 3 (Non-convex setting)**.** It is important to note that the θ-marginal of the stationary measure of the proposed SDE system in (7)-(8) concentrates on the maximisers of the likelihood $\ell_M(\theta)$ with increasing N under Laplace-type regularity conditions (Hwang, 1980) on ℓ_M without the need for the strong-convexity assumption **A**1. This can be readily observed in the proof of Theorem 1 (see Appendix B.1), where the stationary measure of the joint system is shown to have the desired marginal properties. Thus, while **A**1 is used to establish the non-asymptotic convergence rates presented in this section, the algorithm is inherently designed to target the MMLE solution in more general landscapes. ◇

4.3 CONVERGENCE WITH INEXACT GRADIENT

Since the expectation with respect to the prior p_α cannot be computed exactly, the gradient update in Eq. (11) is not directly available. To address this, we approximate the expectation using a ULA chain, which yields a biased estimator of

$\mathbb{E}_{p_{\alpha_k}(x)}\left[\nabla_\alpha U_{\alpha_k}(x)\right]$ at each iteration. Following Dalalyan and Karagulyan (2019), we impose the assumptions below to control the bias introduced by this approximation and the variance of the deviation from the true quantity.

Let $g_k := (1/M)\sum_{m=1}^{M} \nabla_\alpha U_{\tilde{\alpha}_k}(\hat{X}_{k,J}^m)$ be the estimate of $\mathbb{E}_{p_{\tilde{\alpha}_k}(x)}[\nabla_\alpha U_{\tilde{\alpha}_k}(x)]$ obtained using the ULA samples as defined in Eq. (15). Let $\zeta_k := g_k - \mathbb{E}_{p_{\tilde{\alpha}_k}(x)}[\nabla_\alpha U_{\tilde{\alpha}_k}(x)]$ denote the difference between the estimator and the true expectation value for all $k \geq 0$. Finally, let $\mathcal{F}_k$ be the σ-algebra generated by the algorithm (Eqs. (15)–(17)) up to iteration k.

A3. *There exist $\delta > 0$ and $\sigma > 0$ such that*

$$\mathbb{E}\left[\left\|\mathbb{E}\left[\zeta_k|\mathcal{F}_k\right]\right\|^2\right] \leq \delta^2, \quad \mathbb{E}\left[\left\|\zeta_k - \mathbb{E}\left[\zeta_k|\mathcal{F}_k\right]\right\|^2\right] \leq \sigma^2$$

for $k \geq 0$, where $\tilde{W}_{k+1}^{\alpha}$, $\tilde{W}_{k+1}^{\beta}$, $\tilde{W}_{k+1}^{m,n}$ for all $m \in \{1,\dots,M\}$, $n \in \{1,\dots N\}$ in Eqs. (15)–(17) are independent of $(\zeta_0,\dots\zeta_k)$.

Remark 4. Intuitively, the bias ζ_k arises from a finite run of ULA. Here $\delta := \delta(J)$ decreases with the number of ULA iterations J, and both δ and σ implicitly depend on d_α as the gradient error mainly arises in the α component. While not directly applicable to our setting - which involves inexact gradients and averaging across parallel chains - the decomposition of Durmus and Moulines (2017) under geometric ergodicity offers useful intuition on how δ depends on J: ULA error splits into an irreducible $O(\gamma)$ discretisation bias and a decaying initialisation term which decays geometrically in J. This motivates choosing γ small enough to control the bias from inexactly targeting p_α. Hence, if the ULA chain is stable, geometrically ergodic and J is large enough while γ is small enough, **A** 3 can hold in practice. $\diamond$

In this setting we can get a similar bound as in Theorem 1 with additional terms arising from the bias in the expectation estimate.

Theorem 2. *Suppose **A**1, **A**2, **A**3 hold. Let $\tilde{\theta}_k = (\tilde{\alpha}_k, \tilde{\beta}_k)$ be the parameter marginal generated by iterates (15)–(17), then given a step size $0 < h \leq 2/(\mu + L)$, the following holds*

$$\mathbb{E}\left[\|\tilde{\theta}_k - \theta_\star\|^2\right]^{1/2} \leq (1-\mu h)^k \tilde{C}_0 + \tilde{C}_1 h^{1/2} + \frac{\tilde{C}_2}{\sqrt{MN}} + \tilde{C}_3,$$

where $\tilde{C}_0$ is a fully explicit constant that only depends on the initial law of the system, $\tilde{C}_1$ is an explicit constant that depends on $M, N, \sigma, \mu, d_\theta$ and d_x and stabilises as $M, N \to \infty$, $\tilde{C}_2$ is an explicit constant that depends on μ and d_θ, and $\tilde{C}_3$ is an explicit constant that depends on μ and δ.

Proof. See Appendix B.3 for the proof with full expressions of constants. $\square$

The proof follows similar steps to that of Theorem 1, adapted to our setting by applying Theorem 4 from Dalalyan and Karagulyan (2019). Crucially, the resulting bound includes an irreducible bias term that does not vanish as $M, N \to \infty$. As discussed above, this term originates directly from the irreducible bias $\delta(J)$.

5 RELATED WORKS

Training EBMs. Efficient training of EBMs remains a fundamental challenge in generative modelling. As an approximation to MLE, Hinton (2002) introduce Contrastive Divergence (CD) and (Tieleman, 2008) follow up with Persistent CD (PCD) which use short-run chains, though these introduce uncontrolled bias (Nijkamp et al., 2020a). Recent alternatives include non-equilibrium thermodynamic approaches (Carbone et al., 2024; Cuin et al., 2026) and the Energy Discrepancy loss (Schröder et al., 2023, 2024), which bypasses MCMC by optimising a different objective. EBIPLA differentiates itself by providing a diffusion-based alternative in the latent setting of Pang et al. (2020). Based on a novel SDE framework, our Euler-Maruyama discretisation replaces sequential posterior MCMC with simultaneous particle updates. While this resembles a one-step form of PCD, it arises from a distinct theoretical foundation. To our knowledge, the only other work providing a diffusion limit for EBM training is Oliva et al. (2025), though they focus exclusively on the non-latent setting.

EBMs in the Latent Space. Recently, several works studying latent EBMs have appeared. Even though they focus on the study of latent EBMs they are quite different in spirit to our method. Yu et al. (2023) modify the LEBM setting and train an additional diffusion model to facilitate training. Although they obtain superior results, their method is substantially more expensive than the classic Pang et al. (2020). Xiao et al. (2021) train an EBM in the latent space of a pre-trained VAE to motivate the use of an EBM as a prior, as they show it enhances performance of the VAE. Xiao and Han (2022) suggest using a multi-stage noise contrastive estimation instead of improving the classical likelihood training as we do in our work. Finally, Yuan et al. (2025) propose a variational learning scheme for the generator posterior, which is different from our particle based approach, in order to extend the training method to a multimodal setting.

Interacting Particle Algorithms for Learning. Kuntz et al. (2023); Caprio et al. (2025) introduce Particle Gradient Descent (PGD), a diffusion-based particle algorithm for MMLE in latent variable models that jointly optimises model parameters and samples from the posterior distribution of the latent variables. Building on this framework, Akyildiz et al. (2025) incorporate additive noise in the parameter updates and establish associated non-asymptotic convergence guarantees which we build upon and extend to

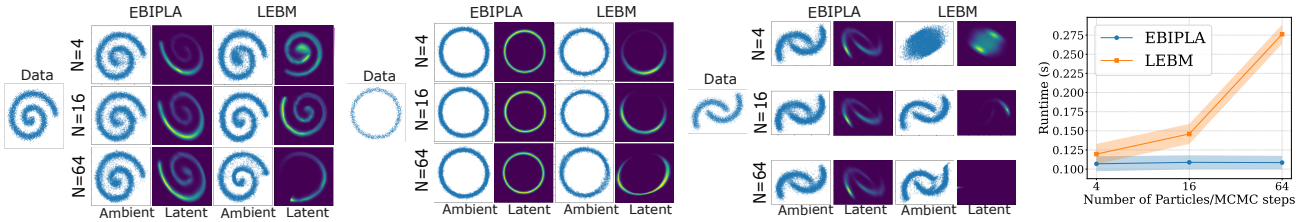


Figure 1: **Left:** Samples generated with EBIPLA and LEBM trained on three synthetic datasets for increasing numbers of posterior particles/MCMC steps together with the corresponding exponentiated energy landscapes. Sample quality and energy accuracy improve with larger particle counts/iterations. The two methods achieve similar performance in terms of sample quality, however EBIPLA learns a more realistic energy landscape **Right:** Comparison of runtime for increasing number of posterior particles (EBIPLA) or MCMC steps (LEBM). EBIPLA consistently achieves comparative qualitative performance while being substantially faster: its runtime scales sub-linearly with the number of particles, remaining below LEBM even at the largest particle count.

multiple data-points and inexact gradient setting. Since then, particle-based learning methods have been taking off in various directions. Momentum-enriched variants of IPLA (Lim et al., 2024; Oliva and Akyildiz, 2024) and extensions to non-smooth settings following Encinar et al. (2025) offer natural extensions to EBIPLA. From a geometric viewpoint, MMLE can be seen as a discretisation of a gradient flow in Wasserstein-2 space (Kuntz et al., 2023) , which suggests potential generalisations based on alternative geometries, as explored by Sharrock et al. (2024) for Stein variational gradient descent, which could be another potential extension of our method. We also mention recent sequential Monte Carlo (SMC)-based approaches for learning (Carbone et al., 2024; Crucinio, 2025; Cuin et al., 2025, 2026), which can be used to develop generalisations of our work where particles have weights.

6 EXPERIMENTS

To evaluate the generative performance of EBIPLA we train it on three synthetic datasets (Section 6.1) and on three image datasets (Section 6.2). On the synthetic task we benchmark against the closest baseline, LEBM (Pang et al., 2020). Compared to the benchmark across both settings, EBIPLA attains slightly better generative performance while delivering substantially lower runtime at matched compute budgets. For the image datasets, we additionally compare to other Latent Variable Models (LVMs).

6.1 SYNTHETIC DATASETS

We evaluate EBIPLA on three synthetic datasets: rotated Swiss roll, rotated Half-moons and rescaled Circle. More precisely, we generate a Swiss roll/Half-moons/Circle in a 2-dimensional latent space $\mathbb{R}^2$ and then map it to the ambient space through an orthogonal transformation $T : \mathbb{R}^2 \to \mathbb{R}^2$ (for Swiss roll and Half-moons) or linear transformation $T : \mathbb{R}^2 \to \mathbb{R}^2$ for the Circle (for details see Appendix D.1).

To contextualise our results, we consider the closest baseline, LEBM, for comparison. To ensure a fair comparison, we match the computational budgets of both models so that they require the same number of gradient evaluations per training iteration. Specifically, the computational complexity for both EBIPLA and LEBM is $\mathcal{O}(B(N + J))$, where B is the batch size and J denotes the number of prior iterations. To choose an appropriate value of J we run an empirical study of the effect of J on the performance of the algorithm (see Appendix E). We select $J = 500$ as a practical trade-off between sample/energy quality and computational cost. By setting $N \in \{4, 16, 64\}$ to represent the number of posterior particles for EBIPLA and the number of posterior MCMC steps for LEBM, we ensure that the same number of gradient calls for the two methods are needed. We train both models for 200 epochs and report per training-step runtime for the Swiss roll dataset averaged over 20 runs (Figure 1).

Sample Quality. In Figure 1, we can see that the sample quality and the accuracy of the learned energy landscape improves with increasing number of posterior particles/iterations. EBIPLA performs comparably in terms of sample quality to LEBM. While EBIPLA learns the energy landscape more accurately, LEBM results in distorted energy landscapes.

Computational Efficiency. Figure 1 shows that even though EBIPLA achieves comparable qualitative performance, empirically EBIPLA is significantly faster: its runtime grows sub-linearly with particle count, whereas LEBM's runtime increases faster than linear with iteration count. This discrepancy stems from the underlying implementation: while LEBM requires sequential posterior updates (typically implemented via a `for`-loop), EBIPLA updates all particles simultaneously. In terms of arithmetic gradient evaluations, both methods scale as $\mathcal{O}(B(N + J))$ under the matched-budget setup; however, because the N posterior-particle updates in EBIPLA are vectorised, the observed wall-clock time scales as $\mathcal{O}(BJ)$ for EBIPLA.

Model	SVHN		CIFAR-10		CelebA64	
	MSE↓	FID↓	MSE↓	FID↓	MSE↓	FID↓
VAE (Kingma and Welling, 2014)	0.019	46.78	0.057	106.37	0.021	65.75
2s-VAE (Dai and Wipf, 2018)	0.019	42.81	0.056	72.90	0.021	44.40
RAE (Ghosh et al., 2019)	0.014	40.02	0.027	74.16	0.018	40.95
SRI ($L = 5$) (Nijkamp et al., 2020b)	0.011	35.32	–	–	0.015	47.95
LEBM (Pang et al., 2020)	0.008	29.44	<u>0.020</u>	**70.15**	0.013	37.87
SM-LEBM (Schröder et al., 2023)	0.010	34.44	0.026	77.82	0.014	41.21
ED-LEBM (Schröder et al., 2023)	<u>0.006</u>	<u>28.10</u>	0.023	<u>73.58</u>	**0.009**	<u>36.73</u>
EBIPLA (ours)	$\mathbf{0.004}_{\pm 4e-5}$	$\mathbf{27.54}_{\pm 0.42}$	$\mathbf{0.017}_{\pm 5e-4}$	$75.13_{\pm 0.74}$	$\underline{0.013}_{\pm 9e-5}$	$\mathbf{35.72}_{\pm 0.46}$

Table 1: Comparison of MSE(↓) and FID(↓) on the SVHN, CIFAR-10, and CelebA64 datasets. We use bold font to indicate the best performance and underline to indicate the second best performance. We report the mean and standard error of the metrics averaged over 5 runs.

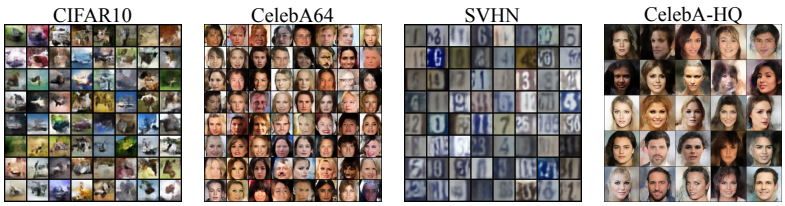


Figure 2: Samples generated with EBIPLA trained on CIFAR-10, CelebA64, SVHN, and CelebA-HQ.

Model	FID↓
VAE	180.49
ABP	160.21
LEBM	133.07
DAMC (Yu et al., 2023)	85.88
EBIPLA (ours)	92.10

Table 2(a) Comparison of FID on the CelebA-HQ dataset. We do not report standard deviation due to computational constraints.

No. of Particles	FID↓
$N = 1$	$31.00_{\pm 0.81}$
$N = 4$	$27.83_{\pm 0.38}$
$N = 16$	$27.09_{\pm 0.31}$
$N = 32$	$26.74_{\pm 0.22}$

Table 2(b) Effect of the number of posterior particles (N) on the generation performance of EBIPLA trained on SVHN averaged over 5 runs.

6.2 IMAGE MODELLING

To validate the effectiveness of our method on high dimensional data, we train EBIPLA on three standard image datasets: CIFAR10 (Krizhevsky and Hinton, 2009), SVHN (Netzer et al., 2011), and CelebA64 (Liu et al., 2015). Following Pang et al. (2020), we use the same architecture for both the energy-based prior and the generator network. We train all models for 200 epochs with a batch size of 128; the number of particles is set to $N = 10$ to balance performance and efficiency following Kuntz et al. (2023). We additionally test our method on high resolution CelebA-HQ 256×256 datasets (Liu et al., 2015; Karras et al., 2018) using $N = 5$ particles to demonstrate its scalability. The complete implementation details and training hyperparameters are provided in Appendix D.2.

Generation and Reconstruction. We evaluate the generative performance of our model quantitatively with the Fréchet Inception Distance (FID) (Heusel et al., 2017) using $50,000$ samples generated by the energy-based prior and decoded back to the pixel space. Additionally, we measure the reconstruction quality in terms of mean-squared error (MSE) on the validation set. We obtain the reconstructions as the maximum a posteriori estimates of $p_\theta(x|y)$ (cf. Section 3) following the implementation in Kuntz et al. (2023); we provide further details of the evaluation procedure in Appendix D.2. In Table 1, we show that our EBIPLA achieves comparable performance among relevant baselines. We focus on comparisons with latent EBM baselines, which allow us to isolate the impact of the introduced training method on the generative and reconstructive performance (for comparison with broader model selection see Section E). On CIFAR-10, our model is least competitive but remains comparable; we attribute this primarily to the simple Euler-Maruyama discretisation used in our implementation, rather than a fundamental limitation of the approach, noting that stronger baselines such as LEBM use substantially longer posterior chains (cf. Table 4). We highlight that EBIPLA has shorter runtimes compared to LEBM (Pang et al., 2020) across settings due to its efficient formulation (cf. Table 4). For the high resolution experiments, we additionally benchmark against Yu et al. (2023), which is a more complex model that uses an additional latent diffusion model to aid training and sampling of LEBM; despite its simplicity, EBIPLA approaches the performance of this stronger baseline while substantially outperforming LEBM.

Furthermore, we show on the SVHN dataset that the performance of EBIPLA improves as we increase the number of posterior particles used in Table 2(b). These empirical trends are consistent with the qualitative prediction of our theory: increasing the number of posterior particles reduces

the particle approximation error. Although the image experiments do not necessarily satisfy the strong log-concavity and smoothness assumptions (**A**1-3) used to obtain the non-asymptotic bounds in Section 4, this behaviour is still expected since the underlying particle system is designed to concentrate around maximisers of the marginal likelihood beyond the analysed setting.

Latent Representations. To qualitatively assess the learned representations, we visualise interpolations in the latent space. We linearly interpolate between the latent vectors x_0 and x_1 as $\texttt{lerp}(x_0, x_1, t) = tx_0 + (1-t)x_1$; we then decode those interpolants back to the pixel space using the trained generator. For training images, we select index m and extract one of the posterior particles randomly from the N particles $X_K^{m,1:N}$ obtained from training (cf. Algorithm 1); for validation images, we use the MAP estimate as in our reconstruction experiments. Figure 4 shows the resulting interpolation trajectories, which suggests that EBIPLA has learned a smooth and semantically meaningful latent space.

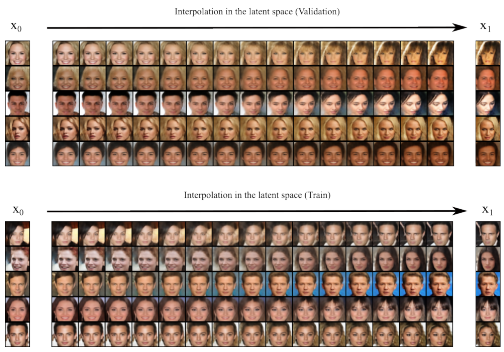


Figure 4: Latent space interpolation of EBIPLA trained on CelebA64 on the training (bottom) and validation set (top).

7 CONCLUSION

EBMs have recently demonstrated strong performance across diverse generative modelling tasks, including image and language modelling (Thornton et al., 2025; Wang and Du, 2025; Gladstone et al., 2026). They offer distinctive capabilities that are difficult to achieve with competing approaches – most notably, direct anomaly detection by thresholding the energy, and compositional generation (Thornton et al., 2025; Du et al., 2020) by combining multiple independently trained EBMs at inference time. Despite these appealing properties, they remain hard to train due to their double intractable nature.

Motivated by this and the recent advances in particle-based training methods, we propose an interacting particle method for learning latent energy-based models (EBIPLA), which provides a more computationally efficient method to train LEBMs. By discretising a system of SDEs that is designed to admit an invariant measure concentrating around the maximiser of the marginal log-likelihood, we obtain a practical algorithm that is both scalable and theoretically grounded (Section 3). In Section 4, we provide to the best of our knowledge the first convergence bounds for training latent energy-based models under the strong log-concavity and smoothness assumptions. In particular, we show that the algorithm converges to the maximiser of the empirical marginal log-likelihood with increasing number of both data points and particles; this ensures that even with a small number of particles reliable performance can be achieved on large datasets. (Theorem 1, Theorem 2). Although the convergence bounds are obtained under the strong log-concavity we expect those can be further relaxed (Zhang et al., 2023) and highlight that the particle system is designed to concentrate around the empirical maximisers of the marginal likelihood even in the non log-concave settings.

Lastly, while for fairness of comparison we use a simple Langevin chain to approximate the prior expectation, the framework is agnostic to the choice of sampler: advanced samplers such as Hamiltonian Monte Carlo (HMC) (Duane et al., 1987) could be employed to improve the generative performance. In addition, extensions to underdamped settings offer a promising direction to obtain accelerated versions of EBIPLA with theoretical guarantees (Oliva and Akyildiz, 2024).

Acknowledgements

J. M. is supported by EPSRC through the Modern Statistics and Statistical Machine Learning (StatML) CDT programme, grant no. EP/Y034813/1. T. W. is supported by the Roth Scholarship from the Department of Mathematics, Imperial College London. We acknowledge computational resources and support provided by the Department of Mathematics and the Imperial College Research Computing Service, DOI: 10.14469/hpc/2232.

Bibliography

Akyildiz, O. D., Crucinio, F. R., Girolami, M., Johnston, T., and Sabanis, S. (2025). Interacting particle Langevin algorithm for maximum marginal likelihood estimation. *ESAIM: Probability and Statistics*, 29:243–280.

Altschuler, J. M. and Chewi, S. (2024). Faster high-accuracy log-concave sampling via algorithmic warm starts. *Journal of the ACM*, 71(3):1–55.

An, D., Xie, J., and Li, P. (2021). Learning deep latent variable models by short-run MCMC inference with optimal transport correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15415–15424.

Arbel, M., Zhou, L., and Gretton, A. (2021). Generalized energy based models. In *ICLR 2021-9th International Conference on Learning Representations*. ICLR.

Atchadé, Y., Jiang, K., and Sun, Y. (2023). On generative energy-based models. Manuscript.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Caprio, R., Kuntz, J., Power, S., and Johansen, A. M. (2025). Error bounds for particle gradient descent, and extensions of the log-Sobolev and Talagrand inequalities. *Journal of Machine Learning Research*, 26(103):1–38.

Carbone, D., Hua, M., Coste, S., and Vanden-Eijnden, E. (2024). Efficient training of energy-based models using Jarzynski equality. *Journal of Statistical Mechanics: Theory and Experiment*, 2024(10):104019.

Chen, R. T., Behrmann, J., Duvenaud, D. K., and Jacobsen, J.-H. (2019). Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32.

Crucinio, F. R. (2025). A mirror descent approach to maximum likelihood estimation in latent variable models. *Journal of Computational and Graphical Statistics*, (just-accepted):1–19.

Cuin, J., Carbone, D., and Akyildiz, O. D. (2025). Learning latent variable models via Jarzynski-adjusted langevin algorithm. *Advances in Neural Information Processing Systems*, 38.

Cuin, J., Carbone, D., Tang, Y., and Akyildiz, O. D. (2026). Efficient stochastic optimisation via sequential Monte Carlo. In *International Conference on Machine Learning*.

Dai, B. and Wipf, D. (2018). Diagnosing and enhancing VAE models. In *International Conference on Learning Representations*.

Dalalyan, A. S. and Karagulyan, A. (2019). User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 129(12):5278–5311.

De Bortoli, V., Durmus, A., Pereyra, M., and Vidal, A. F. (2021). Efficient stochastic optimisation by unadjusted Langevin Monte Carlo. *Statistics and Computing*, 31(3):29.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Deng, Y., Bakhtin, A., Ott, M., Szlam, A., and Ranzato, M. (2021). Residual energy-based models for text generation. *Journal of Machine Learning Research*, 22:1–41.

Douc, R., Moulines, E., and Stoffer, D. (2014). *Nonlinear time series: Theory, methods and applications with R examples*. CRC press.

Du, Y., Durkan, C., Strudel, R., Tenenbaum, J. B., Dieleman, S., Fergus, R., Sohl-Dickstein, J., Doucet, A., and Grathwohl, W. S. (2023). Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and MCMC. In *International Conference on Machine Learning*, pages 8489–8510. PMLR.

Du, Y., Li, S., and Mordatch, I. (2020). Compositional visual generation with energy based models. In *Advances in Neural Information Processing Systems*.

Du, Y., Li, S., Tenenbaum, J., and Mordatch, I. (2021). Improved contrastive divergence training of energy-based models. In *Proceedings of the 38th International Conference on Machine Learning*, pages 2837–2848. PMLR.

Du, Y. and Mordatch, I. (2019). Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222.

Durmus, A. and Moulines, É. (2017). Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. *The Annals of Applied Probability*, 27(3):1551.

Encinar, P. C., Crucinio, F. R., and Akyildiz, O. D. (2025). Proximal interacting particle Langevin algorithms. In *The 41st Conference on Uncertainty in Artificial Intelligence*.

Gao, R., Nijkamp, E., Kingma, D. P., Xu, Z., Dai, A. M., and Wu, Y. N. (2020). Flow contrastive estimation of energy-based models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7518–7528.

Gao, R., Song, Y., Poole, B., Wu, Y. N., and Kingma, D. P. (2021). Learning energy-based models by diffusion recovery likelihood. In *International Conference on Learning Representations*.

Geng, C., Han, T., Jiang, P.-T., Zhang, H., Chen, J., Hauberg, S., and Li, B. (2024). Improving adversarial energy-based model via diffusion process. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR.

Ghosh, P., Sajjadi, M. S., Vergari, A., Black, M., and Scholkopf, B. (2019). From variational to deterministic autoencoders. In *International Conference on Learning Representations*.

Gladstone, A., Nanduru, G., Islam, M. M., Han, P., Ha, H., Chadha, A., Du, Y., Ji, H., Li, J., and Iqbal, T. (2026). Energy-based transformers are scalable learners and thinkers. *International Conference on Learning Representations*.

Glaser, P., Arbel, M., Doucet, A., and Gretton, A. (2022). Maximum likelihood learning of unnormalized models for simulation-based inference. *arXiv preprint arXiv:2210.14756*.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky, K. (2020). Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*.

Grathwohl, W. S., Kelly, J. J., Hashemi, M., Norouzi, M., Swersky, K., and Duvenaud, D. (2021). No mcmc for me: Amortized sampling for fast and stable training of energy-based models. In *International Conference on Learning Representations*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of Wasserstein GANs. *Advances in Neural Information Processing Systems*, 30.

Han, T., Lu, Y., Zhu, S.-C., and Wu, Y. N. (2017). Alternating back-propagation for generator network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Han, T., Nijkamp, E., Fang, X., Hill, M., Zhu, S.-C., and Wu, Y. N. (2019). Divergence triangle for joint training of generator model, energy-based model, and inferential model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8670–8679.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local nash equilibrium. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Hwang, C.-R. (1980). Laplace's method revisited: weak convergence of probability measures. *The Annals of Probability*, pages 1177–1182.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. (2020). Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems*, 31.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. *2nd International Conference on Learning Representations, ICLR*.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario.

Kuntz, J., Lim, J. N., and Johansen, A. M. (2023). Particle algorithms for maximum likelihood training of latent variable models. In *International Conference on Artificial Intelligence and Statistics*, pages 5134–5180. PMLR.

Lim, J. N., Kuntz, J., Power, S., and Johansen, A. M. (2024). Momentum particle maximum likelihood. In *Proceedings of the Forty-First International Conference on Machine Learning*.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Loaiza-Ganem, G., Ross, B. L., Hosseinzadeh, R., Caterini, A. L., and Cresswell, J. C. (2024). Deep generative models through the lens of the manifold hypothesis: A survey and new connections. *Transactions on Machine Learning Research*.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

Nijkamp, E., Gao, R., Sountsov, P., Vasudevan, S., Pang, B., Zhu, S.-C., and Wu, Y. (2022). Mcmc should mix: learning energy-based model with neural transport latent space mcmc. In *International Conference on Learning Representations (ICLR 2022)*.

Nijkamp, E., Hill, M., Han, T., Zhu, S.-C., and Wu, Y. N. (2020a). On the anatomy of MCMC-based maximum likelihood learning of energy-based models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5272–5280. AAAI Press.

Nijkamp, E., Hill, M., Zhu, S.-C., and Wu, Y. N. (2019). Learning non-convergent non-persistent short-run MCMC toward energy-based model. *Advances in Neural Information Processing Systems*, 32.

Nijkamp, E., Pang, B., Han, T., Zhou, L., Zhu, S.-C., and Wu, Y. N. (2020b). Learning multi-layer latent variable model via variational optimization of short run MCMC for approximate inference. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 361–378. Springer.

Oliva, P. F. V. and Akyildiz, O. D. (2024). Kinetic interacting particle Langevin Monte Carlo. *arXiv preprint arXiv:2407.05790*.

Oliva, P. F. V., Akyildiz, O. D., and Duncan, A. (2025). Uniform-in-time convergence bounds for persistent contrastive divergence algorithms. *arXiv preprint arXiv:2510.01944*.

Ostrovski, G., Dabney, W., and Munos, R. (2018). Autoregressive quantile networks for generative modeling. In *International Conference on Machine Learning*, pages 3936–3945. PMLR.

Pang, B., Han, T., Nijkamp, E., Zhu, S.-C., and Wu, Y. N. (2020). Learning latent space energy-based prior model. *Advances in Neural Information Processing Systems*, 33:21994–22008.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434.

Raginsky, M., Rakhlin, A., and Telgarsky, M. (2017). Nonconvex learning via stochastic gradient Langevin dynamics: a nonasymptotic analysis. In *Conference on Learning Theory*, pages 1674–1703. PMLR.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017). PixelCNN++: Improving the pixelCNN with discretized logistic mixture likelihood and other modifications. *International Conference on Learning Representations*.

Santambrogio, F. (2015). *Optimal transport for applied mathematicians*. Springer.

Schröder, T., Ou, Z., Li, Y., and Duncan, A. B. (2024). Energy-based modelling for discrete and mixed data via heat equations on structured spaces. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Schröder, T., Ou, Z., Lim, J. N., Li, Y., Vollmer, S. J., and Duncan, A. (2023). Energy discrepancies: A score-independent loss for energy-based models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Sharrock, L., Dodd, D., and Nemeth, C. (2024). Tuning-free maximum likelihood training of latent variable models via coin betting. In *International Conference on Artificial Intelligence and Statistics*, pages 1810–1818. PMLR.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Song, Y. and Ermon, S. (2020). Improved techniques for training score-based generative models. *Advances in Neural Information Processing Systems*, 33:12438–12448.

Song, Y. and Kingma, D. P. (2021). How to train your energy-based models. *arXiv preprint arXiv:2101.03288*.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

Thornton, J., Béthune, L., ZHANG, R., Bradley, A., Nakkiran, P., and Zhai, S. (2025). Controlled generation with distilled diffusion energy models and sequential monte carlo. In *The 28th International Conference on Artificial Intelligence and Statistics*, volume 89, page 103.

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1064–1071.

Wang, R. and Du, Y. (2025). Equilibrium matching: Generative modeling with implicit energy-based models. *arXiv preprint arXiv:2510.02300*.

Wang, T. Y. J., Kuntz, J., and Akyildiz, O. D. (2026). Training latent diffusion models with interacting particle algorithms. In *The 29th International Conference on Artificial Intelligence and Statistics*.

Whiteley, N., Gray, A., and Rubin-Delanchy, P. (2025). Statistical exploration of the manifold hypothesis. *Journal of the Royal Statistical Society: Series B*.

Xiao, Z. and Han, T. (2022). Adaptive multi-stage density ratio estimation for learning latent space energy-based model. In *Advances in Neural Information Processing Systems*.

Xiao, Z., Kreis, K., Kautz, J., and Vahdat, A. (2021). VAEBM: A symbiosis between variational autoencoders and energy-based models. In *International Conference on Learning Representations*.

Xie, J., Lu, Y., Gao, R., Zhu, S.-C., and Wu, Y. N. (2018). Cooperative training of descriptor and generator networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1):27–45.

Xie, J., Zheng, Z., and Li, P. (2021). Learning energy-based model with variational auto-encoder as amortized sampler. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10441–10451.

Xie, J., Zhu, Y., Li, J., and Li, P. (2022). A tale of two flows: Cooperative learning of Langevin flow and normalizing flow toward energy-based model. In *International Conference on Learning Representations*.

Yu, P., Zhu, Y., Xie, S., Ma, X., Gao, R., Zhu, S.-C., and Wu, Y. N. (2023). Learning energy-based prior model with diffusion-amortized MCMC. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yuan, S., Cui, J., Li, H., and Han, T. (2025). Learning multimodal latent generative models with energy-based prior. In Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., and Varol, G., editors, *Computer Vision – ECCV 2024*, pages 86–100, Cham. Springer Nature Switzerland.

Zhai, S., Cheng, Y., Lu, W., and Zhang, Z. (2016). Deep structured energy based models for anomaly detection. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1100–1109. PMLR.

Zhang, Y., Akyildiz, Ö. D., Damoulas, T., and Sabanis, S. (2023). Nonasymptotic estimates for stochastic gradient Langevin dynamics under local conditions in nonconvex optimization. *Applied Mathematics & Optimization*, 87(2):25.

Zhao, Y., Xie, J., and Li, P. (2021). Learning energy-based generative models via coarse-to-fine expanding and sampling. In *International Conference on Learning Representations*.

Zhu, Y., Xie, J., Wu, Y. N., and Gao, R. (2024). Learning energy-based models by cooperative diffusion recovery likelihood. In *International Conference on Learning Representations*.

Learning Latent Energy-Based Models via Interacting Particle Langevin Dynamics
Supplementary Material

Joanna Marks* **Tim Y. J. Wang*** **O. Deniz Akyildiz**

Department of Mathematics, Imperial College London

A PRELIMINARY RESULTS

Recall that for a data point y, we define $p_\theta(x, y) = p_\alpha(x)p_\beta(y \mid x)$, and we use the notation $\phi_y(\theta, x)$:

$$\phi_y(\theta, x) = -\log p_\theta(x, y). \tag{18}$$

We can write the gradient:

$$\nabla_\theta \log p_\theta(x, y) = \nabla_\theta \log p_\alpha(x) + \nabla_\theta \log p_\beta(y \mid x). \tag{19}$$

A.1 DERIVATION OF $\nabla_\alpha \phi_y(\theta, x)$

Since $p_\beta(y_m|x)$ is independent of α the gradient reads

$$\nabla_\alpha \phi_y(\theta, x) = -\nabla_\alpha \log p_\theta(x, y) = -\nabla_\alpha \log p_\alpha(x). \tag{20}$$

By definition $p_\alpha(x) = e^{-U_\alpha(x)}/Z_\alpha$, so

$$\nabla_\alpha \phi_y(\theta, x) = -\nabla_\alpha \log p_\alpha(x) = \nabla_\alpha U_\alpha(x) + \nabla_\alpha \log Z_\alpha \tag{21}$$

where $Z_\alpha = \int e^{-U_\alpha(x)} \mathrm{d}x$ is the normalising constant which is unknown. However, provided differentiation under the integral sign is justified the gradient of $\log Z_\alpha$ can be rewritten as

$$\nabla_\alpha \log Z_\alpha = \frac{\nabla_\alpha Z_\alpha}{Z_\alpha} = \frac{\int \nabla_\alpha e^{-U_\alpha(x)} \mathrm{d}x}{Z_\alpha} \tag{22}$$

$$= -\int \nabla_\alpha U_\alpha(x) p_\alpha(x) \mathrm{d}x = -\mathbb{E}_{p_\alpha(x)}[\nabla_\alpha U_\alpha(x)] \tag{23}$$

hence

$$\nabla_\alpha \phi_y(\theta, x) = \nabla_\alpha U_\alpha(x) - \mathbb{E}_{p_\alpha(x)}[\nabla_\alpha U_\alpha(x)] \tag{24}$$

and for a set of particles $\{X_t^{m,n}\}_{m=1,n=1}^{m=M,n=N}$ and a parameter $\theta_k = (\alpha_k, \beta_k)$

$$\frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \nabla_\alpha \phi_{y_m}(\theta_k, X_k^{m,n}) = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \nabla_\alpha U_{\alpha_k}(X_k^{m,n}) - \mathbb{E}_{p_{\alpha_k}(x)}[\nabla_\alpha U_{\alpha_k}(x)] \tag{25}$$

A.2 DERIVATION OF $\nabla_\beta \phi_y(\theta, x)$

Similarly, given a data point y since $p_\alpha(x)$ is independent of β the gradient of $\phi_y(\theta, x)$ with respect to β reads

$$\nabla_\beta \phi_y(\theta, x) = -\nabla_\beta \log p_\theta(x, y) = -\nabla_\beta \log p_\beta(y|x) \tag{26}$$

Let $V_\beta(x, y) = -\log p_\beta(y \mid x)$. Then for a set of particles $\{X_t^{m,n}\}_{m=1,n=1}^{m=M,n=N}$ and a parameter $\theta_k = (\alpha_k, \beta_k)$

$$\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N} \nabla_\beta \phi_{y_m}(\theta_k, X_k^{m,n}) = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N} \nabla_\beta V_\beta(X_k^{m,n}, y_m)) \tag{27}$$

which, if we assume a standard isotropic Gaussian decoder, that is, $p(y_m \mid x) = \mathcal{N}(y_m; g_\beta(x), \sigma^2 I)$, then $V_\beta(x, y_m) = (1/2\sigma^2)\|y_m - g_\beta(x)\|^2 + \text{const}$, where g_β denotes the decoder (generator).

A.3 DERIVATION OF $\nabla_x \phi_y(\theta, x)$

Recall $V_\beta(x, y) = -\log p_\beta(y \mid x)$. Then given a data point y, the gradient of $\phi_y(\theta, x)$ with respect to x reads

$$\begin{aligned}\nabla_x \phi_y(\theta, x) &= -\nabla_x \log p_\theta(x, y) = -\nabla_x \log p_\alpha(x) - \nabla_x \log p_\beta(y|x) &(28)\\ &= \nabla_x U_\alpha(x) + \nabla_x V_\beta(x, y) &(29)\end{aligned}$$

since the normalising constant Z_α is independent of x.

B PROOFS

B.1 PROOF OF THEOREM 1

Proof. Recall the discretised system of SDEs defined in (9)–(10)

$$\theta_{k+1} = \theta_k - \frac{h}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N} \nabla_1 \phi_{y_m}(\theta_k, X_k^{m,n}) + \sqrt{\frac{2h}{MN}} W_k, \tag{30}$$

$$X_{k+1}^{m,n} = X_k^{m,n} - h\nabla_2 \phi_{y_m}(\theta_k, X_k^{m,n}) + \sqrt{2h} W_k^{m,n}, \tag{31}$$

where $\phi_{y_m}(\theta, x) = -\log p_\theta(x, y_m)$ and which is equivalent to the SDE system in (11)–(13). In the above display, we use ∇_1 and ∇_2 to denote the derivatives w.r.t. first and second arguments of ϕ_{y_m}, respectively, as this will be important in the proof.

Consider the function $\Phi : \mathbb{R}^{d_\theta} \times (\mathbb{R}^{d_x})^{M\times N} \to \mathbb{R}$, defined as

$$\Phi(\theta, z^{1:M,1:N}) = \sum_{m=1}^{M}\sum_{n=1}^{N} \phi_{y_m}(\theta, \sqrt{MN} z^{m,n}).$$

Next, let us define the following probability distribution:

$$\pi(\theta, z^{1:M,1:N}) \propto \exp(-\Phi(\theta, z^{1:M,1:N})).$$

Our strategy in this proof is to notice that the iterates (30)–(31) can be seen as a discretisation of an SDE, targeting π. In order to see this, consider the Langevin SDE targeting π:

$$\mathrm{d}\theta_t = -\sum_{m=1}^{M}\sum_{n=1}^{N} \nabla_1 \phi_{y_m}(\theta_t, \sqrt{MN} Z_t^{m,n})\mathrm{d}t + \sqrt{2}\mathrm{d}B_t, \tag{32}$$

$$\mathrm{d}Z_t^{m,n} = -\sqrt{MN}\nabla_2 \phi_{y_m}(\theta_t, \sqrt{MN} Z_t^{m,n})\mathrm{d}t + \sqrt{2}\mathrm{d}B_t^{m,n}. \tag{33}$$

Euler-Maruyama discretisation of this SDE with step-size $\bar{h} = h/MN$ gives us

$$\theta_{k+1} = \theta_k - \frac{h}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\nabla_1\phi_{y_m}(\theta_k, \sqrt{MN}Z_k^{m,n}) + \sqrt{\frac{2h}{MN}}W_k, \tag{34}$$

$$Z_{k+1}^{m,n} = Z_k^{m,n} - \frac{h}{\sqrt{MN}}\nabla_2\phi_{y_m}(\theta_k, \sqrt{MN}Z_k^{m,n}) + \sqrt{\frac{2h}{MN}}W_k^{m,n}. \tag{35}$$

Now, if we define $X_k^{m,n} = \sqrt{MN}Z_k^{m,n}$, we recover exactly (9)–(10) (i.e. (30)–(31) above). In other words, up to a simple rescaling, the systems are identical. Most importantly, θ-marginals of both scaled algorithms are identical, thus the analysis of the iterates defined in (34) will give us the convergence of the parameters.

To assess first whether this target distribution is desirable, we consider the θ-marginal of π, denoted by π_Θ. That is, π_Θ is obtained by integrating out all latent variables $z^{1:M,1:N}$ from the joint distribution π:

$$\begin{aligned}
\pi_\Theta(\theta) &= \int \pi(\theta, z^{1:M,1:N})\mathrm{d}z^{1:M,1:N} = \int \exp(-\Phi(\theta, z^{1:M,1:N}))\mathrm{d}z^{1:M,1:N} \\
&= \int \prod_{m=1}^{M}\prod_{n=1}^{N}\exp(-\phi_{y_m}(\theta, \sqrt{MN}z^{m,n}))\mathrm{d}z^{1:M,1:N} = \prod_{m=1}^{M}\prod_{n=1}^{N}\int \exp(-\phi_{y_m}(\theta, \sqrt{MN}z^{m,n}))\mathrm{d}z^{m,n}, \\
&\propto \prod_{m=1}^{M}\left(\int \exp(-\phi_{y_m}(\theta, x))\mathrm{d}x\right)^N \propto \prod_{m=1}^{M} p_\theta(y_m)^N, \\
&\propto \exp\left(N\sum_{m=1}^{M}\log p_\theta(y_m)\right).
\end{aligned}$$

Thus, for a fixed dataset $\{y_m\}_{m=1}^M$ as $N \to \infty$, the distribution $\pi_\Theta(\theta)$ concentrates around the maximisers of the target loss ℓ_M and we will aim to quantify this concentration rate. To proceed, we first establish the convexity properties of the function Φ. Under **A**1 and **A**2, ϕ_{y_m} is μ-strongly convex and L-smooth in both arguments for all $m \in \{1, \ldots, M\}$. By arguments analogous to those in Oliva and Akyildiz (2024, Lemmas A.4–A.5), it follows that Φ is $MN\mu$-strongly convex and MNL-smooth in both arguments. Since $-\log\pi_\Theta(\theta) = -N\sum_{m=1}^{M}\log p_\theta(y_m) + \text{const}$, and each $-\log p_\theta(y_m)$ is μ-strongly convex in θ by Remark 1 π_Θ is μMN-strongly log-concave. Hence, to obtain the concentration rate we can apply Altschuler and Chewi (2024, Lemma A.8), which gives

$$W_2(\delta_{\theta_\star}, \pi_\Theta) \leq \sqrt{\frac{d_\theta}{\mu MN}}.$$

Next, we look at the rate of convergence. Since Φ is $MN\mu$-strongly convex and MNL-smooth in both arguments, an exponential convergence rate will hold by using well-known convergence results of Langevin diffusions. Let $\nu_k = \mathcal{L}(\theta_k, Z_k^{1:M,1:N})$ be the law of the system in (34)–(35). Given that, the target π is $\bar{\mu} = MN\mu$-strongly log-concave and $\bar{L} = MNL$ log-smooth, the discretisation of this system with $\bar{h} = h/MN$ satisfies (Dalalyan and Karagulyan, 2019, Theorem 1)

$$W_2(\nu_k, \pi) \leq (1 - \bar{h}\bar{\mu})^k W_2(\nu_0, \pi) + 1.65\left(\frac{\bar{L}}{\bar{\mu}}\right)\sqrt{d_\theta + MNd_x}\bar{h}^{1/2}, \tag{36}$$

where $0 < \bar{h} \leq 2/(\bar{\mu} + \bar{L})$. Plugging $\bar{h} = h/MN$, $\bar{\mu} = MN\mu$, and $\bar{L} = MNL$ to the above bound, we get

$$W_2(\nu_k, \pi) \leq (1 - h\mu)^k W_2(\nu_0, \pi) + 1.65\left(\frac{L}{\mu}\right)\sqrt{\frac{d_\theta + MNd_x}{MN}}h^{1/2}, \tag{37}$$

where the step-size restriction can also be simplified to $0 < h \leq 2/(\mu + L)$. Now, let us denote $\nu_k^\theta = \mathcal{L}(\theta_k)$ and $\pi_\Theta(\theta) = \int \pi(\theta, z^{1:M,1:N})\mathrm{d}z^{1:M,1:N}$, then by the definition of Wasserstein distance, we have

$$W_2(\nu_k^\theta, \pi_\Theta) \leq W_2(\nu_k, \pi),$$

concluding the convergence part.

Merging these results, we arrive at

$$\mathbb{E}\left[\|\theta_k - \theta_\star\|^2\right]^{1/2} \leq (1-\mu h)^k W_2(\nu_0, \pi) + 1.65\left(\frac{L}{\mu}\right)\sqrt{\frac{d_\theta + MNd_x}{MN}}h^{1/2} + \sqrt{\frac{d_\theta}{\mu MN}}. \tag{38}$$

□

B.2 PRELIMINARY RESULT FOR THEOREM 2

In this section, we rework Theorem 4 in Dalalyan and Karagulyan (2019) in the same assumption setting as in **A**3. Let $f : \mathbb{R}^p \to \mathbb{R}$ be a μ-strongly convex function with L-Lipschitz gradients. Let $\boldsymbol{\vartheta}_{k,h}$ denote the iterates of the noisy LMC (nLMC) algorithm as defined in Dalalyan and Karagulyan (2019)

$$\boldsymbol{\vartheta}_{k+1,h} = \boldsymbol{\vartheta}_{k,h} - h\nabla f\left(\boldsymbol{\vartheta}_{k,h}\right) + h\boldsymbol{\zeta}_k + \sqrt{2h}\boldsymbol{\xi}_{k+1}; \quad k = 0, 1, 2, \ldots \tag{39}$$

where $h > 0$ and $\boldsymbol{\xi}_{k+1}$ are a collection of standard Gaussian random variables of appropriate dimension. Suppose for some $\delta > 0$ and $\sigma > 0$ and for every $k \in \mathbb{N}$, **A**3 holds for $(\boldsymbol{\zeta}_k)_{k\geq 0}$. We then have the following proposition.

Proposition 1 (Theorem 4 in Dalalyan and Karagulyan (2019) adapted)**.** *Let $\boldsymbol{\vartheta}_{K,h}$ be the K'th the iterate of the nLMC algorithm and v_K be its distribution. If the function $f : \mathbb{R}^p \to \mathbb{R}$ is L-smooth and μ-strongly convex and $h \leq 2/(\mu + L)$ then*

$$W_2\left(v_K, \pi\right) \leq (1-\mu h)^K W_2\left(v_0, \pi\right) + 1.65(L/\mu)(hp)^{1/2} + \frac{\delta}{\mu} + \frac{\sigma^2 h^{1/2}}{1.64Lp^{1/2} + \sigma\sqrt{\mu}}$$

Proof. Under **A**3, modified Proposition 2 in Dalalyan and Karagulyan (2019) yields

$$W_2\left(\nu_{k+1}, \pi\right)^2 \leq \left\{(1-\mu h)W_2\left(\nu_k, \pi\right) + \alpha L\left(h^3 p\right)^{1/2} + h\delta\right\}^2 + \sigma^2 h^2 \tag{40}$$

Applying Dalalyan and Karagulyan (2019, Lemma 1) with $A = \mu h, B = \sigma h$ and $C = \alpha L\left(h^3 p\right)^{1/2} + h\delta$, implies that $W_2\left(\nu_k, \pi\right)$ is less than or equal to

$$(1-\mu h)^k W_2\left(\nu_0, \pi\right) + \frac{\alpha L(hp)^{1/2} + \delta}{\mu} + \frac{\sigma^2 h}{\alpha L(hp)^{1/2} + \delta + (\mu h)^{1/2}\sigma} \tag{41}$$

where $\alpha = 7\sqrt{2}/6 \in (1.64, 1.65)$.

□

B.3 PROOF OF THEOREM 2

Proof. Let us define $\tilde{\theta}_k = (\tilde{\alpha}_k, \tilde{\beta}_k)$. We can compactly write (15)–(17) as:

$$\tilde{\theta}_{k+1} = \tilde{\theta}_k - \frac{h}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\nabla_1\phi_{y_m}(\tilde{\theta}_k, \tilde{X}_k^{m,n}) + h\boldsymbol{\zeta}_k + \sqrt{\tfrac{2h}{MN}}W_k, \tag{42}$$

$$\tilde{X}_{k+1}^{m,n} = \tilde{X}_k^{m,n} - h\nabla_2\phi_{y_m}(\tilde{\theta}_k, \tilde{X}_k^{m,n}) + \sqrt{2h}W_k^{m,n}, \tag{43}$$

where $\mathbb{R}^{d_\theta} \ni \boldsymbol{\zeta}_k = (\zeta_k, 0_{d_\beta})^\intercal$ and where $\zeta_k := g_k - \mathbb{E}_{p_{\tilde{\alpha}_k}(x)}[\nabla_\alpha U_{\tilde{\alpha}_k}(x)]$ and $g_k := M^{-1}\sum_{m=1}^{M}\nabla_\alpha U_{\tilde{\alpha}_k}(\hat{X}_{k,J}^m)$. Following the same steps as in the proof of Theorem 1 we rescale the system with step-size $\tilde{h} = \frac{h}{MN}$. Note that this results in a $\tilde{h}MN\zeta_k$ term in Equation (42) and that by **A**3 we have

$$\mathbb{E}\left[\left\|\mathbb{E}\left[MN\zeta_k|\mathcal{F}_k\right]\right\|^2\right] \leq (MN\delta)^2,$$
$$\mathbb{E}\left[\left\|MN\zeta_k - \mathbb{E}\left[MN\zeta_k|\mathcal{F}_k\right]\right\|^2\right] \leq (MN\sigma)^2$$

Using $\tilde{h} = \frac{h}{MN}$, $\tilde{\mu} = MN\mu$, $\tilde{L} = MNL$, $\tilde{\delta} = MN\delta$ and $\tilde{\sigma} = MN\sigma$ to Proposition 1 we obtain the following bound:

$$\mathbb{E}\left[\|\tilde{\theta}_k - \theta_\star\|^2\right]^{1/2} \leq (1-\mu h)^k W_2(\nu_0, \pi) + 1.65\left(\frac{L}{\mu}\right)\sqrt{\frac{d_\theta + MNd_x}{MN}}h^{1/2} + \frac{\sigma^2\sqrt{h}}{1.64L\sqrt{\frac{d_\theta+MNd_x}{MN}} + \sigma\sqrt{\mu}} + \frac{\delta}{\mu} + \sqrt{\frac{d_\theta}{\mu MN}}. \tag{44}$$

We can then identify

$$\begin{aligned}
\tilde{C}_0 &= W_2(\nu_0, \pi), \\
\tilde{C}_1 &= 1.65\left(\frac{L}{\mu}\right)\sqrt{\frac{d_\theta + MNd_x}{MN}} + \frac{\sigma^2}{1.64L\sqrt{\frac{d_\theta+MNd_x}{MN}} + \sigma\sqrt{\mu}}, \\
\tilde{C}_2 &= \sqrt{\frac{d_\theta}{\mu}}, \\
\tilde{C}_3 &= \frac{\delta}{\mu}
\end{aligned}$$

□

C IMPLEMENTATION DETAILS

C.1 A PRACTICAL ALGORITHM

We now provide a few modifications to enable efficient simulation of the gradient flow. We include the pseudocode in Algorithm 2.

Subsampling The updates in Eqs. (15)-(17) are prohibitively expensive due to the need of averaging over the entire training set of M data points. Instead, we adopt the mini-batching strategy (Kuntz et al., 2023; Wang et al., 2026) and compute a subsampled version of the losses. For a minibatch of indices $\mathcal{B} \subseteq [M]$, we compute the energy loss and generator loss as:

$$\hat{\mathcal{L}}_d(X^{1:M,1:N}, \mathcal{B}) = \frac{1}{N|\mathcal{B}|}\sum_{(m,n)\in\mathcal{B}\times[N]} V(X^{m,n}, y^m), \tag{45}$$

$$\hat{\mathcal{L}}_e(\hat{X}_J^{1:M}, X^{1:M,1:N}, \mathcal{B}) = \frac{1}{N|\mathcal{B}|}\sum_{(m,n)\in\mathcal{B}\times[N]} U(X^{m,n}) - \frac{1}{|\mathcal{B}|}\sum_{m\in\mathcal{B}} U(\hat{X}_J^m), \tag{46}$$

where $X^{1:M,1:N}$ are the posterior particles, $\hat{X}_J^{1:M}$ are particles sampled from the energy prior using J Langevin steps. This reduces the computational cost for each update to the posterior particles from $\mathcal{O}(MN)$ to $\mathcal{O}(|\mathcal{B}|N)$.

Noise Addition Due to the use of mini-batching, we also adapt the Langevin dynamics to account for the noise added to the posterior particles $X^{1:M,1:N}$. For $M = BL$, where $B = |\mathcal{B}|$, so that one epoch contains L mini-batch updates and each cloud $X^{m,1:N}$ receives one posterior drift update per epoch. We add the Brownian motion scaled by $\sqrt{1/L}$ to all particles every time the generator is updated, which matches with the amount of noise added in (17). When M is not divisible by B, we set $L = M/B$ as an effective noise-scaling factor and sample mini-batches according to the implementation described below. Since the noise addition does not involve gradient computations, it is relatively cheap compared to the posterior updates.

Adaptive Optimisers To accelerate convergence, we use the Adam optimiser Kingma and Ba (2015) for the network parameters (α, β), following the practice in Pang et al. (2020); Schröder et al. (2023). We detail the hyperparameters in Appendix D.1 and Appendix D.2.

Algorithm 2 A practical version of EBIPLA

Require: Number of training iterations K, number of particles N, number of prior iterations J, initial parameters (α_0, β_0), observed data $\{y^m\}_{m=1}^M$, batch size $|\mathcal{B}|$, scaling $L = M/|\mathcal{B}|$, initial particles $X_0^{1:M,1:N}$, stepsizes $h > 0$ and $\gamma > 0$

for $k = 0$ to $K - 1$ **do**
  Sample batch of indices $\mathcal{B} \subseteq [M]$
  $X_{k+1}^{1:M,1:N} \leftarrow X_k^{1:M,1:N}$
  $X_{k+1}^{\mathcal{B},1:N} \leftarrow X_k^{\mathcal{B},1:N} - h\left[\nabla_x U_{\alpha_k}(X_k^{\mathcal{B},1:N}) + \nabla_x V_{\beta_k}(X_k^{\mathcal{B},1:N}, y^{\mathcal{B}})\right]$ # Update posterior
  $X_{k+1}^{1:M,1:N} \leftarrow X_{k+1}^{1:M,1:N} + \sqrt{2h/L}\,\tilde{W}_k^{1:M,1:N}$
  Sample $\hat{X}_{k,0}^{\mathcal{B}} \sim \mathcal{N}(0, I)$
  for $j = 0$ to $J - 1$ **do** # Generate new auxiliary prior samples
    $\hat{X}_{k,j+1}^{\mathcal{B}} \leftarrow \hat{X}_{k,j}^{\mathcal{B}} - \gamma\,\nabla_x U_{\alpha_k}(\hat{X}_{k,j}^{\mathcal{B}}) + \sqrt{2\gamma}\,W_{k,j}^{\mathcal{B}}$
  end for
  Compute energy loss $\hat{\mathcal{L}}_e = \hat{\mathcal{L}}_e(\hat{X}_{k,J}^{\mathcal{B}}, X_k^{1:M,1:N}, \mathcal{B})$ in (46)
  Compute generator loss $\hat{\mathcal{L}}_d = \hat{\mathcal{L}}_d(X_k^{1:M,1:N}, \mathcal{B})$ in (45)
  $\alpha_{k+1} \leftarrow \mathrm{OptimizerStep}(\alpha_k, \hat{\mathcal{L}}_e)$
  $\beta_{k+1} \leftarrow \mathrm{OptimizerStep}(\beta_k, \hat{\mathcal{L}}_d)$
end for

C.2 EBIPLA WITH WARMUP FOR IMAGE EXPERIMENTS

To quickly traverse through the transient phase for the posterior particles in image experiments (Kuntz et al., 2023), we adopt a warm-up scheme similar to the predictor-corrector SDE solver in Song et al. (2021), correcting the solution to the posterior SDE in (8) using Langevin dynamics. We apply the warm-up during the first S_{warm} training iterations and run the Langevin dynamics for 10 steps at each warm-up iteration. After the warm-up, we initialise $X_{\text{post}}^{m,1:N}$ by creating N identical copies of each $X_{\text{post}}^{m,1}$ obtained from the persistent chains. The Gaussian noises in the inner warm-up Langevin steps are sampled independently across inner iterations. We provide the pseudocode in Algorithm 3.

C.3 COMPUTATIONAL RESOURCES

All experiments were conducted on internal HPCs with NVIDIA L40S (48G) GPUs, internal servers with NVIDIA RTX 3090 (24G) GPUs, or rented cloud GPUs.

D EXPERIMENTAL DETAILS

D.1 SYNTHETIC EXPERIMENTS

For implementation of EBIPLA on the synthetic dataset we use the practical algorithm as detailed in Section C.1. For fairness of comparison we implement LEBM (Pang et al., 2020) without the prior tilting introduced by the authors. We observed that for our example dataset the tilting which is a form of prior regularisation hindered the performance of both EBIPLA and LEBM and hence decided to drop it for both methods.

Architecture We set the standard deviation of the Gaussian decoder for both models to $\sigma = 0.05$. For the prior energy $U_\alpha(x)$ we use a multi-layer perceptron (MLP) with three hidden layers, each with 128 hidden units and SiLU activation for EBIPLA and ReLU activation for LEBM. We observed that ReLU worked better for the baseline method. The generator $g_\beta(z) : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$ is a single linear layer mapping.

Training We train both models for 200 epochs with the batch size of 1000 data points. For the parameter updates (α, β) we use the Adam optimiser with learning rate 1×10^{-2} for both. The exponential decay rates for the first and second moment estimates were configured as $\beta_1 = 0.5$ and $\beta_2 = 0.999$, respectively. We vary the prior ULA step size γ and the posterior discretisation step size h in EBIPLA, as well as the prior γ and posterior h ULA step sizes in LEBM, according to the number of posterior particles or MCMC steps (see Table 2, Table 3).

Algorithm 3 A practical version of EBIPLA with warmup

Require: Number of training iterations K, number of prior iterations J, warmup iterations S_{warm}, initial parameters (α_0, β_0), observed data $\{y^m\}_{m=1}^M$, batch size $|\mathcal{B}|$, scaling $L = M/|\mathcal{B}|$, initial particles $X_0^{1:M,1:N'}$ with N', final number of particles N with $N \equiv 0 \pmod{N'}$, stepsizes $h > 0$, $h_{\text{warm}} > 0$, and $\gamma > 0$.

for $k = 0$ to $K - 1$ **do**
  Sample batch of indices $\mathcal{B} \subseteq [M]$ `# Update posterior samples`
  $X_{k+1}^{1:M,1:N'} \leftarrow X_k^{1:M,1:N'}$
  if $k < S_{\text{warm}}$ **then**
    $X^\dagger \leftarrow X_k^{\mathcal{B},1:N'}$
    for $i = 1, \ldots, 10$ **do**
      $X \leftarrow X^\dagger - h_{\text{warm}} \left[\nabla_x U_{\alpha_k}(X^\dagger) + \nabla_x V_{\beta_k}(X^\dagger, y^{\mathcal{B}})\right] + \sqrt{2h_{\text{warm}}}\epsilon_{k,i}^{\mathcal{B}}$
    end for
    $X_{k+1}^{\mathcal{B},1:N'} \leftarrow X^\dagger$
  else if $k = S_{\text{warm}}$ **then**
    Replicate each particle cloud in $X_{k+1}^{1:M,1:N'}$ to obtain $X_{k+1}^{1:M,1:N}$
    Set $N' \leftarrow N$
  else
    $X_{k+1}^{\mathcal{B},1:N'} \leftarrow X_k^{\mathcal{B},1:N'} - h\left[\nabla_x U_{\alpha_k}(X_k^{\mathcal{B},1:N'}) + \nabla_x V_{\beta_k}(X_k^{\mathcal{B},1:N'}, y^{\mathcal{B}})\right]$
    $X_{k+1}^{1:M,1:N'} \leftarrow X_{k+1}^{1:M,1:N'} + \sqrt{2h/L}\,\tilde{W}_k^{1:M,1:N'}$
  end if
  Sample $\hat{X}_{k,0}^{\mathcal{B}} \sim \mathcal{N}(0, I)$
  for $j = 0$ to $J - 1$ **do** `# Generate new auxiliary prior samples`
    $\hat{X}_{k,j+1}^{\mathcal{B}} \leftarrow \hat{X}_{k,j}^{\mathcal{B}} - \gamma\,\nabla_x U_{\alpha_k}(\hat{X}_{k,j}^{\mathcal{B}}) + \sqrt{2\gamma}\,W_{k,j}^{\mathcal{B}}$
  end for
  Compute energy loss $\hat{\mathcal{L}}_e = \hat{\mathcal{L}}_e(\hat{X}_{k,J}^{\mathcal{B}}, X_{k+1}^{1:M,1:N'}, \mathcal{B})$ in (46)
  Compute generator loss $\hat{\mathcal{L}}_d = \hat{\mathcal{L}}_d(X_{k+1}^{1:M,1:N'}, \mathcal{B})$ in (45)
  $\alpha_{k+1} \leftarrow \text{OptimizerStep}(\alpha_k, \hat{\mathcal{L}}_e)$
  $\beta_{k+1} \leftarrow \text{OptimizerStep}(\beta_k, \hat{\mathcal{L}}_d)$
end for

D.1.1 Dataset creation details

Swiss roll To create the rotated Swiss roll we sample $t \in [t_{\min}, t_{\max}]$ uniformly in arc length to ensure even distribution of the data points. Latent points are $x_{\text{lat}}(t) = \frac{1}{8}[t\cos t,\ t\sin t] + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 0.01I)$. We map to the ambient space via a fixed linear decoder $T : \mathbb{R}^2 \to \mathbb{R}^2$ with orthonormal rows, which results in a rotated Swiss roll (see Figure 5). The dataset size was set to 10,000 data points.

Half-moons To create the rotated Half-moons dataset, we use the `make_moons` function from `sklearn` to sample $N = 10000$ points from the standard two-moons distribution in latent space $\mathbb{R}^2$ with zero intrinsic noise. We then add isotropic Gaussian perturbations $x_{\text{lat}} = x_{\text{moon}} + \varepsilon, \varepsilon \sim \mathcal{N}(0,\ 0.01I)$ which corresponds to noise standard deviation 0.1 per coordinate. To obtain observations in ambient space, we apply a fixed linear decoder $T_\theta : \mathbb{R}^2 \to \mathbb{R}^2$, given by a rotation matrix $T_\theta = \left[\begin{smallmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{smallmatrix}\right]$ with $\theta = \pi/3$ in our experiments. This yields a rotated version of the noisy half-moons geometry in the ambient space (see Figure 5).

Circle For the rescaled Circle dataset, we sample $N = 10,000$ points in latent space $\mathbb{R}^2$ by drawing angles $\phi_i \sim \text{Unif}(0, 2\pi)$ and setting $x_{\text{circle},i} = r\,[\cos\phi_i, \sin\phi_i]^T$. We then add isotropic Gaussian perturbations $x_{\text{lat}} = x_{\text{circle}} + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ with $\sigma = 0.05$ in our default setup. Observations in ambient space are obtained via a fixed linear scaling map $T(x) = 3x$. This yields a rescaled circle in the ambient space (see Figure 5).

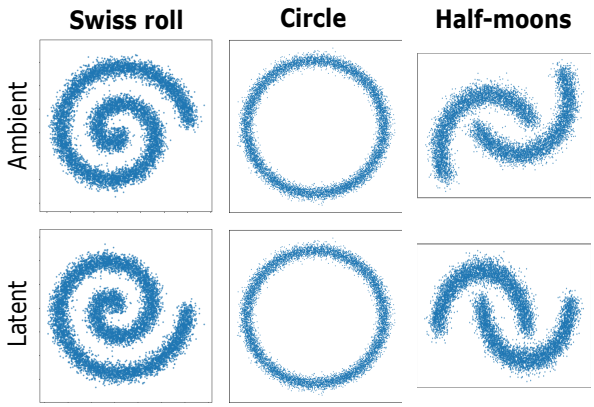


Figure 5: Visualisation of the datasets in latent and ambient space

Table 2: Step sizes used for the prior and posterior updates in EBIPLA across the Swiss roll, Half-moons, and Circle datasets. N denotes the number of posterior particles.

N	Swiss roll		Half-moons		Circle	
	γ	h	γ	h	γ	h
4	0.005	0.9	0.004	1.0	0.005	0.7
16	0.007	0.9	0.004	1.0	0.005	0.7
64	0.009	0.9	0.004	1.0	0.005	0.7

D.2 IMAGE EXPERIMENTS

Model and Architecture. We set the standard deviation of the Gaussian decoder to $\sigma = 0.3$ following Pang et al. (2020). The architecture of the energy and generator networks for SVHN and CelebA experiments are chosen to be the same as Pang et al. (2020) to ensure a fair comparison. For CIFAR10, we use the same generator network as our SVHN experiment instead of the larger version used in Pang et al. (2020) to avoid overfitting. We adopt the same energy and generator networks as Yu et al. (2023) for the CelebA-HQ experiments.

Training. We train EBIPLA on all datasets for 200 epochs with a batch size of 128. We use the Adam optimiser for both the energy and generator networks: for the energy network, we use a learning rate of 2×10^{-4} and set $(\beta_1, \beta_2) = (0.5, 0.999)$, for the generator we use a learning rate of 1×10^{-3} and $(\beta_1, \beta_2) = (0.9, 0.999)$; an exponential learning rate schedule with rate 0.999 was used for both networks. The warmup is applied during the first 10K iterations of the training with $h_{\text{warm}} = 0.005$ (cf. Appendix C.2). For particle updates, we choose $\gamma = 0.1$ and $h = 0.1$ with the number of prior steps set to 60.

Evaluation. For the computation of FID values, we sample $50,000$ from the energy-based prior. To ensure our evaluation accurately reflects the learned model, we use sampling hyperparameters identical to our training procedure: 60 Langevin dynamics steps with a step size of 0.1.

For the evaluation of reconstruction error, we use a similar setup to Kuntz et al. (2023) and compute the maximum a posteriori (MAP) estimate for the latents x^m corresponding to each y^m in the validation set:

$$x^m_{\text{map}} = \arg\max_{x \in \mathbb{R}^{d_x}} \log p(x|y^m) = \arg\min_{x \in \mathbb{R}^{d_x}} \frac{1}{2\sigma^2} \|y^m - g_\beta(x)\|_2^2 + U_\alpha(x).$$

Then, we recover the image by mapping x^m_{map} through the generator. To solve the MAP estimation problem above, we use 4 randomly initialised runs of Adam (Kingma and Ba, 2015) of length 50 iterations and set the learning rate using PyTorch's `ReduceLROnPlateau` scheduler with an initial learning rate of 1; all other Adam parameters are set to the defaults.

Runtime Comparison. We additionally provide a comparison of runtime between our method EBIPLA and LEBM (Pang et al., 2020) under the same settings in our image experiments. The runtime results were averaged over 5 runs.

Table 3: Step sizes used for the prior and posterior updates in LEBM across the Swiss roll, Half-moons, and Circle datasets. N denotes the number of MCMC steps.

N	Swiss roll		Half-moons		Circle	
	γ	h	γ	h	γ	h
4	0.003	0.005	0.004	0.01	0.005	0.003
16	0.006	0.005	0.004	0.005	0.005	0.003
64	0.005	0.005	0.004	0.005	0.01	0.003

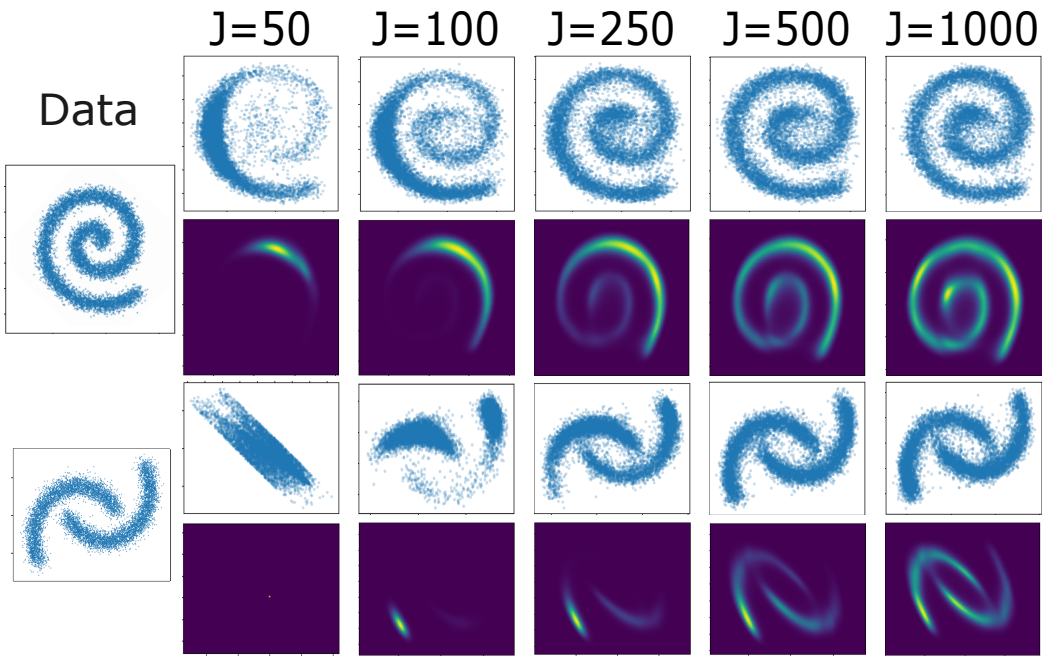
Dataset	**LEBM (Pang et al., 2020)**		**EBIPLA (ours)**	
	Walltime (s)	GFLOPS	Walltime (s)	GFLOPS
CIFAR10	$2.06_{\pm 0.07}$	1.50	$0.99_{\pm 0.07}$	0.80
CelebA64	$1.59_{\pm 0.06}$	0.80	$1.46_{\pm 0.06}$	0.79
SVHN	$0.49_{\pm 0.07}$	0.75	$0.33_{\pm 0.07}$	0.74

Table 4: Per-training-step wall-clock time (↓) and Giga Floating-Point Operations Per Second (GFLOPS) across three dataset configurations, measured on a single NVIDIA RTX3090 GPU.

E ADDITIONAL RESULTS

E.1 IMPACT OF MCMC STEPS

The figure below shows samples generated with EBIPLA for $N = 64$ trained on rotated Swiss roll and rotated Half Moons datasets for increasing numbers of prior ULA chain steps J, together with the corresponding exponentiated energy landscapes. Both sample quality and the energy landscape improve with increasing J. We select $J = 500$ as a practical trade-off between sample/energy quality and computational cost. While performance improves further at $J = 1000$, the gains are marginal relative to the additional runtime for this study.



E.2 FURTHER COMPARISON WITH GENERATIVE MODELS

The following tables contextualise our results against a broader set of mainstream modern generative models. We report FID comparisons on CIFAR-10, SVHN, and CelebA across latent-variable models, VAEs, autoregressive models, GANs, score-based models, flows, EBMs, and extensions thereof. We note that EBIPLA is not intended to compete with state-of-the-art generative models such as diffusion or flow-based models. Our comparisons focus on latent EBM baselines, which allow us to assess the impact of the introduced training method on the generative and reconstructive performance.

Table 5: Additional comparison of FID(↓) on CIFAR-10.

Family	**Model**	**FID↓**
VAE	VAE (Kingma and Welling, 2014)	78.41
	2s-VAE (Dai and Wipf, 2018)	72.90
	RAE (Ghosh et al., 2019)	74.16
Autoregressive	PixelCNN (Salimans et al., 2017)	65.93
	PixelIQN (Ostrovski et al., 2018)	49.46
GAN	WGAN-GP (Gulrajani et al., 2017)	36.40
	SN-GAN (Miyato et al., 2018)	21.70
	StyleGAN2-ADA (Karras et al., 2020)	2.92
Score-based	NCSN (Song and Ermon, 2019)	25.32
	NCSN-v2 (Song and Ermon, 2020)	31.75
	NCSN++ (Song et al., 2021)	2.20
Flow	Glow (Kingma and Dhariwal, 2018)	45.99
	Residual Flow (Chen et al., 2019)	46.37
EBM	LEBM (Pang et al., 2020)	70.15
	EBM-SR (Nijkamp et al., 2019)	44.50
	EBM-IG (Du and Mordatch, 2019)	38.20
	EBM-CD (Du et al., 2021)	25.10
	SM-LEBM (Schröder et al., 2023)	77.82
	ED-LEBM (Schröder et al., 2023)	73.58
EBM+Extensions	CoopVAEBM (Xie et al., 2021)	36.20
	CoopNets (Xie et al., 2018)	33.61
	Divergence Triangle (Han et al., 2019)	30.10
	VARA (Grathwohl et al., 2021)	27.50
	GEBM (Arbel et al., 2021)	19.31
	CF-EBM (Zhao et al., 2021)	16.71
	VAEBM (Xiao et al., 2021)	12.16
	EBM-Diffusion (Gao et al., 2021)	9.60
	CDRL (Zhu et al., 2024)	3.68
	DDAEBM (Geng et al., 2024)	4.82
	NT-EBM (Nijkamp et al., 2022)	78.12
	EBM-FCE (Gao et al., 2020)	37.30
	CoopFlow (Xie et al., 2022)	15.80
Ours	EBIPLA	$75.13_{\pm 0.74}$

Table 6: Additional comparison of FID(↓) on SVHN.

Family	**Model**	**FID↓**
Latent variable	ABP (Han et al., 2017)	49.71
	ABP-SRI (Nijkamp et al., 2020b)	35.23
	ABP-OT (An et al., 2021)	19.48
	SRI ($L = 5$) (Nijkamp et al., 2020b)	35.32
VAE	VAE (Kingma and Welling, 2014)	46.78
	2s-VAE (Dai and Wipf, 2018)	42.81
	RAE (Ghosh et al., 2019)	40.02
GAN	DCGAN (Radford et al., 2015)	21.40
Flow	Glow (Kingma and Dhariwal, 2018)	41.70
EBM	LEBM (Pang et al., 2020)	29.44
	SM-LEBM (Schröder et al., 2023)	34.44
	ED-LEBM (Schröder et al., 2023)	28.10
EBM+Extensions	NT-EBM (Nijkamp et al., 2022)	48.01
	EBM-FCE (Gao et al., 2020)	20.19
	CoopFlow (Xie et al., 2022)	15.32
Ours	EBIPLA	$27.54_{\pm 0.42}$

Table 7: Additional comparison of FID(↓) on CelebA.

Family	**Model**	**FID↓**
Latent variable	ABP (Han et al., 2017)	51.50
	ABP-SRI (Nijkamp et al., 2020b)	36.84
	SRI ($L = 5$) (Nijkamp et al., 2020b)	47.95
VAE	VAE (Kingma and Welling, 2014)	38.76
	VAE (Kingma and Welling, 2014)	65.75
	2s-VAE (Dai and Wipf, 2018)	44.40
	RAE (Ghosh et al., 2019)	40.95
GAN	DCGAN (Radford et al., 2015)	12.50
Flow	Glow (Kingma and Dhariwal, 2018)	23.32
EBM	LEBM (Pang et al., 2020)	37.87
	SM-LEBM (Schröder et al., 2023)	41.21
	ED-LEBM (Schröder et al., 2023)	36.73
EBM+Extensions	EBM-FCE (Gao et al., 2020)	12.21
	DDAEBM (Geng et al., 2024)	10.29
	GEBM (Arbel et al., 2021)	5.21
	CoopFlow (Xie et al., 2022)	4.15
Ours	EBIPLA	$35.72_{\pm 0.46}$