Title

Energy-Based Model and its Applications

Author

Zhu, Yaxuan

Publication Date

2024

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Energy-Based Model and its Applications

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Statistics

by

Yaxuan Zhu

2024

ABSTRACT OF THE DISSERTATION

Energy-Based Model and its Applications

by

Yaxuan Zhu

Doctor of Philosophy in Statistics

University of California, Los Angeles, 2024

Professor Yingnian Wu, Chair

The Energy-Based Model (EBM) represents a class of probabilistic generative models that offers a robust and versatile framework for modeling arbitrary data distributions. In recent years, EBMs have increasingly captured the interest of both the academic and industrial sectors. Despite their potential, the training and practical application of EBMs pose significant challenges. This thesis conducts a systematic study of EBMs, beginning with a case study on 3D modeling. This example not only showcases the strengths of EBMs but also highlights the difficulties encountered during their training process. The discussion then progresses to the intricacies of EBM training, particularly emphasizing the challenge of the time-consuming sampling process, which may not always yield beneficial samples for updating EBMs. To address this issue, we introduce a strategy to amortize the sampling process using specially designed initializer models. We developed algorithms to facilitate cooperative training between the EBM and the initializer models. The resulting algorithms, CoopFlow and CDRL, demonstrate competitive performance across a variety of tasks, showcasing their efficacy in overcoming traditional EBM training hurdles.

The dissertation of Yaxuan Zhu is approved.

Qing Zhou

Jingyi Li

Mark S. Handcock

Yingnian Wu, Committee Chair

University of California, Los Angeles

2024

To my parents, Yuliang and Jianhong,

whose love and support have illuminated my path.

To my dear friends,

whose unwavering presence and encouragement sustained me through moments of stress.

And to myself,

for the resilience shown in the face of numerous challenges, leading me to this moment of achievement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I express my profound gratitude to my advisor, Dr. Yingnian Wu, for his unwavering support and invaluable guidance throughout my Ph.D. journey. My heartfelt thanks also go to my esteemed committee members: Dr. Mark S. Handcock, Dr. Jingyi Li, and Dr. Qing Zhou, whose expertise and advice were crucial to my research.

In addition to my committee and advisor, I owe a deep sense of gratitude to my mentors and significant collaborators, Dr. Jianwen Xie and Dr. Ruiqi Gao. They introduced me to the world of generative modeling and provided hands-on research guidance. I am also thankful for the insightful discussions and assistance from all my talented peers and lab alumni at UCLA, as well as the many excellent individuals I've had the pleasure of working with during my academic and internship experiences at Baidu Seattle Research, Alibaba DAMO Group and Google.

The Ph.D. journey is challenging and stressful, but I was fortunate to have friends who supported me during the toughest times. Special thanks to Dr. Haoxin Zheng, my roommate, for being there to listen and encourage me when I was down, and to Dr. Weinan Song, for his substantial help throughout my time in Los Angeles. I am grateful to Wenyan Zhao, Xixie Zhou, Xitong Zhou, Xuan Hu, Yuqi Zhang, Yudi Zhuo, Mingfei Dong, Minqi Liu, Yiming Zhou, Yifeng Lan, Yifei Xu, Lin Du, Junzi Tan, Qianhui Gao, Hanzhi Xia, Linfang Wang, Yu Zhao, Haoyun Jiang, Xiaofeng Gao, Qian Long, Ziheng Zhou, Zixiang Chen, Mei Wu, Siting Liu, Qi Wu, Dehong Xu, Minglu Zhao, Zhujun Fang, Jiangtao Chen, Zhanhao Peng, Peijun Zhu, Yiling Chen, Ruiqi Gao, Yutan Gu, Shuwen Qiu, Peiyu Yu, Yasi Zhang, Lingyu Zhan, Tianxiang Li, Zhuqi Li, Kun Jin, Zhuqi Li and many others for their camaraderie in board games, badminton, and hiking. These activities, although mostly for fun, help me a lot in fighting against the stress in my routine life.

Finally, I must express my deepest appreciation to my parents, Yuliang Zhu and Jianhong Ma, for their selfless material and emotional support, which was instrumental in my achievements. I am also grateful to my grandparents, whose love and care have left an indelible mark on my heart.

VITA

2013–2017 B.S. in Electrical Engineering, Peking University

2017–2019 M.S. in Computer Science, UCLA

2019–2024 Ph.D. Candidate in Statistics, UCLA

2020 Research Intern, Alibaba, DAMO Academy

2021–2022 Research Intern, Baidu Cognitive Computing Laboratory, Seattle Research

2023 Research Intern, Google

PUBLICATIONS

Yaxuan Zhu, Jianwen Xie, Ying Nian Wu and Ruiqi Gao. Learning Energy-Based Models by Cooperative Diffusion Recovery Likelihood. In the 12th International Conference on Learning Representations (ICLR), 2024. (Spotlights Presentation)

Peiyu Yu, **Yaxuan Zhu**, Sirui Xie, Xiaojian Ma, Ruiqi Gao, Song-Chun Zhu and Ying Nian Wu. Learning Energy-Based Prior Model with Diffusion-Amortized MCMC. In the 37th Conference on Neural Information Processing Systems (NeurIPS), 2023.

Yaxuan Zhu, Jianwen Xie, and Ping Li. Generative neural radiance field with energy-based latent space for 3d-aware disentangled image representation. In the 25th International Conference on Artificial Intelligence and Statistics (AISTATS), 2023.

Jianwen Xie, **Yaxuan Zhu**, Yifei Xu, Dingcheng Li, and Ping Li. A Tale of Two Latent Flows:

Learning Latent Space Normalizing Flow with Short-Run Langevin Flow for Approximate Inference. In the 37th AAAI Conference on Artificial Intelligence (AAAI), 2023.

Jianwen Xie, **Yaxuan Zhu**, Jun Li, Ping Li. A Tale of Two Flows: Cooperative Learning of Langevin Flow and Normalizing Flow Toward Energy-Based Model. In the 10th International Conference on Learning Representations (ICLR), 2022.

Yaxuan Zhu, Ruiqi Gao, Siyuan Huang, Song-chun Zhu and Yingnian Wu. Learning Neural Representation of Camera Pose with Matrix Representation of Pose Shift via View Synthesis. In the Conference on Computer Vision and Pattern Recognition (CVPR), 2021. (Oral Presentation)

CHAPTER 1

Introduction

1.1 Background

The exploration of Energy-Based Models (EBMs) harks back to seminal contributions by (ZWM98a; LCH06; Hin12). EBMs represent a class of probabilistic generative models. An EBM define an unnormalized probability density function for its data, articulated as the exponential of an energy function. This function allocates a scalar energy value to each input configuration, where higher energy values signify configurations of greater likelihood.

In recent developments, EBMs have demonstrated their versatility and utility across a wide spectrum of application areas, including realistic image synthesis (XLZ16a; XLG18; NHZ19; DM19; AZG21; HNM22; XKK21; LJP23; GKH21a; CH23), graph generation (LYO21), compositional generation (DLM20; DDS23), video generation (XZW21), 3D generation (XXZ21; XZG18), physics simulation (CC24), simulation-based inference (GAD22), stochastic optimization (KCZ22), out-of-distribution detection (GWJ20; LWO20), and latent space modeling (PHN20; ZXL23; ZXB23; YZX23).

Despite the success and broad applicability of EBMs, the challenges associated with their training and sampling processes, primarily due to the intractability of the partition function, remain significant. Efforts to scale up EBM training have been made, but a perceivable gap persists between EBMs and other generative models, such as Generative Adversarial Networks (GANs) (CWD18) and diffusion models (SWM15; HJA20; SE19). This gap poses a barrier to the wider application of EBMs.

1.2 Research Objective

This thesis aims to advance the development of EBM. We seek to address a crucial question: How can we construct a robust EBM capable of modeling complex distributions and generating high-quality samples? Answering this requires a thorough analysis of the existing EBM training algorithms, pinpointing the primary obstacles, and devising an algorithm to overcome them. Beyond the foundational training mechanisms, this work explores the application of EBMs to real-world challenges. With a specific focus on the task of 3D space modeling, we examine the integration of an EBM prior within the latent space of a specialized 3D-aware generator. The findings underscore the efficacy and potential of EBMs.

1.3 Dissertation Outline

The rest of this dissertation is arranged as follows:

- **Chapter2** introduces the basic formulation of EBMs. This chapter discusses the central algorithm for training EBMs, focusing on maximum likelihood estimation (MLE), and examines the sampling algorithms crucial for EBMs' practical application. It also briefly introduces the development of EBMs, from classic models with predefined filters to recent models using deep convolutional neural networks. Additionally, the chapter covers conditional EBMs and EBMs applied to the latent space of generator models. Through this discussion, the chapter aims to provide readers with a solid understanding of the fundamental concepts and methodologies behind EBMs, setting the groundwork for further exploration in the following chapters.

- **Chapter3** pivots to the application of EBM in 3D shape modeling, proposing the innovative NeRF-LEBM. This model, a likelihood-based, top-down, 3D-aware 2D image generative framework, seamlessly integrates 3D representation through Neural Radiance Fields (NeRF) and the 2D imaging process via differentiable volume rendering. It conceptualizes image

generation as a rendering process from a 3D object to a 2D image, conditioned on latent variables that encapsulate object characteristics within informative, trainable energy-based prior models. To train the NeRF-LEBM, we introduce two likelihood-based frameworks: (i) maximum likelihood estimation with inference via Markov chain Monte Carlo and (ii) variational inference employing the reparameterization trick. Our investigations span scenarios with both known and unknown camera positions, and empirical results on benchmark datasets underscore NeRF-LEBM's prowess in inferring 3D structures from 2D images, generating novel views and objects, and learning from partial images and those with varying camera perspectives. This exploration not only showcases the practical applications of EBM but also highlights the model's challenges and advantages in training.

- Following the insights from **Chapter3**, **Chapter4** explores the development of enhanced training algorithms for EBM. It scrutinizes the cooperative learning of two generative models - a normalizing flow model and a Langevin flow model, which are iteratively updated with jointly synthesized samples. The chapter outlines a generative framework that employs a normalizing flow to initialize MCMC chains for an energy-based model, followed by a short-run Langevin flow adjustment. Then we treat the synthesized examples as fair samples from the energy-based model and update the model parameters with the maximum likelihood learning gradient, while the normalizing flow directly learns from the synthesized examples by maximizing the tractable likelihood. Under the short-run non-mixing MCMC scenario, the estimation of the energy-based model is shown to follow the perturbation of maximum likelihood, and the short-run Langevin flow and the normalizing flow form a two-flow generator that we call CoopFlow. The effectiveness of CoopFlow, demonstrated through realistic image synthesis, reconstruction, and interpolation, underscores the strategic advantage of selecting an appropriate initializer model to enhance EBM's generative capabilities.

- Building on the achievements of **Chapter4**, **Chapter5** endeavors to further narrow the performance divide between EBM and other generative models. Motivated by recent advancements in learning EBMs through maximizing diffusion recovery likelihood (DRL) (GSP21a), we

introduce the Cooperative Diffusion Recovery Likelihood (CDRL) framework. This novel approach facilitates the learning and sampling from a sequence of EBMs across progressively noisier dataset versions, each paired with a designated initializer model. Within a cooperative training paradigm, samples from the initializer model serve as preliminary points, refined by a few MCMC steps from the EBM, which is then optimized via recovery likelihood maximization. Meanwhile, the initializer model enhances based on the refinement difference. Incorporating practical EBM training enhancements, CDRL markedly improves generation quality over existing methods on CIFAR-10 and ImageNet datasets. Additionally, we validate CDRL's various scenarios and tasks, such as classifier-free guided generation, compositional generation, image inpainting, and out-of-distribution detection.

CHAPTER 2

Preliminaries of Energy-Based Model

2.1 Basic Formulation

Consider sample $\mathbf{x}$ drawn from an underlying data distribution denoted as $p_{\text{data}}(\mathbf{x})$. An energy-based model (EBM) models the density of $\mathbf{x}$ through the following equation:

$$p_\theta(\mathbf{x}) = \frac{1}{Z_\theta} \exp(f_\theta(\mathbf{x})), \tag{2.1}$$

where f_θ represents the unnormalized log density, or the negative energy, parameterized by a neural network yielding a scalar output. The term $Z_\theta = \int \exp[f_\theta(\mathbf{x})] d\mathbf{x}$ refers to the normalizing constant or partition function. It is noteworthy that Z_θ is generally analytically intractable.

2.1.1 Maximum Likelihood Learning

Given a set of unlabeled training examples $\{\mathbf{x}_i, i = 1, ..., n\}$ from an unknown data distribution $p_{\text{data}}(\mathbf{x})$, the EBM described in Equation (2.1) can be trained utilizing the examples $\{\mathbf{x}_i\}$ via Markov chain Monte Carlo (MCMC)-based maximum likelihood estimation. In this process, MCMC samples are generated from the model $p_\theta(\mathbf{x})$ to approximate the gradient of the log-likelihood function, facilitating the update of model parameters θ. Specifically, the log-likelihood function is defined as $L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i)$. For sufficiently large n, maximizing $L(\theta)$ effectively minimizes the Kullback-Leibler (KL) divergence $\mathbb{D}_{\text{KL}}(p_{\text{data}} || p_\theta)$. The gradient of the learning

process is given by

$$L'(\theta) = \mathbb{E}_{p_{\text{data}}}[\nabla_\theta f_\theta(x)] - \mathbb{E}_{p_\theta}[\nabla_\theta f_\theta(x)] \approx \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta f_\theta(x_i) - \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta f_\theta(\tilde{x}_i), \tag{2.2}$$

where the expectations are approximated by averaging over both the observed examples $\{\mathbf{x}_i\}$ and the synthesized examples $\{\tilde{\mathbf{x}}_i\}$, the latter being generated from the current model $p_\theta(\mathbf{x})$. Thus, gradient-based optimization for an EBM typically involves an inner loop of MCMC sampling, which can be computationally intensive for high-dimensional datasets.

2.1.2 Sampling for EBM

To sample from the currently estimated distribution $p_\theta(\mathbf{x})$, gradient-based Markov Chain Monte Carlo (MCMC) methods such as Langevin Dynamics are frequently employed. These methods iteratively update the current sample according to the formula:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\delta^2}{2}\nabla_{\mathbf{x}} f_\theta(\mathbf{x}_{t-1}) + \delta\epsilon_t; \quad \mathbf{x}_0 \sim p_0(\mathbf{x}), \quad \epsilon_t \sim \mathcal{N}(0, I_D), \quad t = 1, \cdots, T, \tag{2.3}$$

where $\sim p_0(\mathbf{x})$ denotes the initial distribution for the sampling process. For intricate data distributions, starting directly from Gaussian or Uniform noise often necessitates an extensive MCMC updating process to yield meaningful samples. Consequently, techniques such as contrastive divergence (Hin02a; DLT21a), persistent chains (XLZ16a), replay buffers (DM19), or short-run MCMC sampling (NHZ19) are utilized to approximate the analytically intractable learning gradient. To enhance the scalability and stability of EBM training for generating high-fidelity data, approaches such as multi-grid sampling (GLZ18), progressive training (ZXL21), and diffusion (GSP21a) have been implemented. Amortizing MCMC sampling with learned networks (KB16; XLG18; KOG19; XKK21; HNF19a; GKH21a) is also a widely used strategy. Among these, cooperative networks (CoopNets) (XLG18) conduct joint training of a top-down generator and an EBM via MCMC teaching, leveraging the generator as a rapid initializer for Langevin sampling.

Our work adopts a cooperative learning strategy. By carefully designing the EBM and the generator model, we introduce the algorithms CoopFlow and CDRL in subsequent chapters. We demonstrate that these designs significantly enhance the performance of Energy-Based Models.

2.2 Different Energy-Based Models

2.2.1 From FRAME Model to Deep Energy-Based Models

In the initial phases, Energy-Based Models (EBMs) utilized predefined filters. For instance, with image data $\mathbf{x}$, the FRAME (Filter, Random Field, And Maximum Entropy) model (ZWM98b) articulated the energy function as follows:

$$p_\theta(\mathbf{x}) = \frac{1}{Z_\theta} \exp\left(\sum_{u,s,a} \theta_{u,s,a} |\langle \mathbf{x}, H_{u,s,a} \rangle|\right) p_0(\mathbf{x}), \tag{2.4}$$

where p_0 denotes the reference distribution, such as the Normal or Uniform Distribution, and $H = H_{u,s,a}$ signifies a collection of linear heuristic filters characterized by position u, scale s, and orientation a. Common examples of such filters include isotropic Difference of Gaussian (DoG) filters and elongated, oriented Gabor wavelet filters. The notation $\langle ., . \rangle$ represents the convolution operation between the image and the filters. Model training encompasses iterative processes of filter selection and parameter adjustment.

With the advent of neural networks, more recent endeavors (LZW16; XLZ16b) have expanded upon the concept of predefined linear filters to encompass features extracted by deep convolutional neural networks. Consequently, the training paradigm has evolved from selecting specific filters to updating the features across the entire neural network architecture.

2.2.2 Conditional Energy-Based Model

Energy-Based Models (EBMs) can be extended to model conditional distributions. For example, consider modeling the distribution of $\mathbf{x}$ given a condition $\mathbf{c}$. This approach involves the representation of both the joint probability of $\mathbf{x}$ and $\mathbf{c}$, and the conditional probability of $\mathbf{x}$ given $\mathbf{c}$, as follows:

$$\begin{aligned} p_\theta(\mathbf{x}, \mathbf{c}) &= \frac{1}{Z(\theta)} \exp(f_\theta(\mathbf{x}, \mathbf{c})) \\ p_\theta(\mathbf{x}|\mathbf{c}) &= \frac{p_\theta(\mathbf{x}, \mathbf{c})}{p_\theta(\mathbf{c})} = \frac{1}{Z(\theta, \mathbf{c})} \exp(f_\theta(\mathbf{x}, \mathbf{c})), \end{aligned} \tag{2.5}$$

where $Z(\theta, \mathbf{c}) = Z(\theta) \cdot p_\theta(\mathbf{c})$. This framework closely resembles that of the standard EBM, except that both $\mathbf{x}$ and $\mathbf{c}$ are inputs to the neural network. Sampling from $p_\theta(\mathbf{x}|\mathbf{c})$ remains feasible using 2.3, given that the logarithm of the conditional probability, $\log p_\theta(\mathbf{x}|\mathbf{c}) = f_\theta(\mathbf{x}, \mathbf{c}) + \text{const}$, can be directly computed.

2.2.3 Energy-Based Model in the latent space

In certain applications, the observed sample $\mathbf{x}$ may be associated with latent factors $\mathbf{z}$. Then we can model the joint probability of $\mathbf{x}$ and $\mathbf{z}$ as follows:

$$\begin{aligned} p_\theta(\mathbf{x}, \mathbf{z}) &= p_\alpha(\mathbf{z}) p_\beta(\mathbf{x}|\mathbf{z}) \\ p_\theta(\mathbf{x}) &= \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \end{aligned} \tag{2.6}$$

From this, the gradient of the log-likelihood function is derived as:

$$\begin{aligned} \nabla_\theta \log p_\theta(\mathbf{x}) &= \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z})] \\ &= \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\nabla_\theta \log p_\alpha(\mathbf{z}) + \log p_\beta(\mathbf{x}|\mathbf{z})] \\ &= \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\nabla_\alpha \log p_\alpha(\mathbf{z})] + \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\log p_\beta(\mathbf{x}|\mathbf{z})] \end{aligned} \tag{2.7}$$

where the conditional distribution $p_\beta(\mathbf{x}|\mathbf{z})$ can be instantiated by a specifically designed generator g_β. For instance, $p_\beta(\mathbf{x}|\mathbf{z})$ might be modeled as $\mathcal{N}(g_\beta(\mathbf{z}), \sigma^2\mathbf{I})$. The optimization of the generator model is guided by the reconstruction loss $\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\log p_\beta(\mathbf{x}|\mathbf{z})]$.

Regarding the prior model, studies such as (PHN20; XH22) have investigated using an Energy-Based Model (EBM) to represent the prior distribution:

$$p_\alpha(\mathbf{x}) = \frac{1}{Z(\alpha)} \exp(f_\alpha(\mathbf{x}))p_0(\mathbf{z}) \tag{2.8}$$

Subsequently, the loss function for the EBM is articulated as:

$$\begin{aligned} \nabla_\alpha \log p_\theta(\mathbf{x}) &= \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\nabla_\alpha \log p_\alpha(\mathbf{z})] \\ &= \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\nabla_\alpha f_\alpha(\mathbf{z})] - \mathbb{E}_{p_\alpha(\mathbf{z})}[\nabla_\alpha f_\alpha(\mathbf{z})], \end{aligned} \tag{2.9}$$

Optimizing Equation 2.9 necessitates sampling from both the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ and the prior distribution $p_\alpha(\mathbf{z})$. Both sampling processes can be efficiently conducted through Langevin Dynamics.

CHAPTER 3

A Case Study First: Likelihood-Based Generative Radiance Field with Latent Space Energy-Based Model for 3D-Aware Disentangled Image Representation

3.1 Motivation and Introduction

Before talking about improving the training algorithms of EBMs, it is essential to first understand the advantages and challenges of EBMs. Let's first take a look at a case study of 3D-aware image synthesis. In 3D-aware image synthesis, traditional methods generate 3D representations of objects either in a voxel-based format (ZZZ18) or via intermediate 3D features (AMG18), then use differentiable rendering to convert the 3D object into 2D views. However, voxel-based 3D representation is discrete and memory-inefficient, limiting these methods to generating low-quality and low-resolution images. Recently, the Neural Radiance Field (NeRF)(MST22) has emerged as a new type of 3D representation, achieving impressive results in new view synthesis. NeRF represents a continuous 3D scene or object through a mapping function parameterized by a neural network, which takes a 3D location and a viewing direction as input and outputs color and density values. Visualization of the 3D object is achieved by querying the mapping function at each specific 3D location and viewing direction, followed by volume rendering(KH84) to produce image pixel intensities.

Typically, each NeRF function can only represent a single object and must be trained from multiple views of that object. By generalizing the original NeRF function to a conditional version

with latent variables that account for the object's appearance and shape, GRAF (SLN20) builds a 2D image generator based on conditional NeRF and trains the generator for 3D-aware controllable image synthesis via adversarial learning. The NeRF-VAE (KSZ21) proposes training the NeRF-based generator through variational inference, where a bottom-up inference network allows the inference of 3D structures of objects in unseen test images. Both GRAF and NeRF-VAE assume the object-specific latent variables follow simple, non-informative Gaussian distributions. As a likelihood-based model, NeRF-VAE can only handle training images with known camera poses due to the difficulty of inferring the unknown camera pose for each observed image. In contrast, GRAF, can easily learn from images with unknown camera poses because its adversarial learning scheme does not require inference. Therefore, recently, adversarial NeRF-based generative models have advanced rapidly, while progress in developing likelihood-based NeRF-based generative models has lagged. Likelihood-based generative models offer many advantages, such as a stable learning process without mode collapse, the ability to infer latent variables from training and testing examples, and the capability to learn from incomplete data via unsupervised learning. Our study aims to advance the development of likelihood-based generative radiance fields.

Specifically, by leveraging the NeRF-based image generator and latent space energy-based models (LEBMs)(PHN20), we propose NeRF-LEBM, a novel likelihood-based 3D-aware generative model for 2D images. We build energy-based models (EBMs) in the latent space of the NeRF-based generator(SLN20). The latent space EBMs are treated as informative prior distributions. We follow empirical Bayes and train the EBM priors and the NeRF-based generator simultaneously from observed data. The trainable EBM priors over latent variables (appearance and shape of the object) allow sampling novel objects from the model and rendering images from arbitrary viewpoints, improving the latent spaces' capacity and the NeRF-based generator's expressivity. Given a set of 2D training images presenting multiple objects with various appearances, shapes, and viewpoints, we first study the scenario in (KSZ21), where each image's viewpoint is known. We propose training the models using maximum likelihood estimation (MLE) with Markov chain Monte Carlo (MCMC)-based inference (BZ20), which does not require an additional assisting network. At each iteration,

the learning algorithm runs MCMC sampling of the latent variables from the EBM priors and the posteriors. The EBM priors are updated based on samples from the prior and posterior distributions, while the generator is updated based on samples from the posteriors and the observed data. For efficient training and inference, we also propose using amortized inference to train the NeRF-LEBM as an alternative. Lastly, we assume that the camera pose of each image is unknown and treat it as a latent variable following a uniform prior distribution. We use the von Mises-Fisher (vMF) (DFC18) distribution to approximate the camera pose's posterior in our amortized inference framework. Our experiments show that the proposed likelihood-based generative model can synthesize images with new objects and arbitrary viewpoints and learn meaningful disentangled representations of images in scenarios with both known and unknown camera poses. The model can even learn from incomplete 2D training images for controlled generation and 3D-aware inference. In summary, in this chapter:

1. We demonstrate the power of EBMs through the task of 3D-aware image synthesis.

2. To address this task, we propose a novel NeRF-based 2D generative model with a trainable energy-based latent space for 3D-aware image synthesis and disentangled representation.

3. The model can be trained by MLE, which requires inference at each iteration. We use two different inference methods: MCMC-based inference and amortized inference with an inference network.

4. Through our analysis and experiments, we test the efficiency, effectiveness, and performance of the proposed NeRF-LFBM model and learning algorithms, highlighting the advantages of employing EBMs. We also identify the key challenge of EBM training: the difficulty of generating good negative samples during training.

3.2 Related work

3D-aware image synthesis Prior works study controllable image generation by adopting 3D data as supervision (WG16; ZZZ18) or 3D information as input (AMG18; OMN19). Several

works (KUH18; HMR19; NMJ19; LCL19) build discriminative mapping functions from 2D images to 3D shapes, followed by differentiable rendering to project the 3D generated objects back to images for computing reconstruction errors on image domain. Unlike the aforementioned reconstruction-based frameworks, several recent works, such as GRAF (SLN20), GIRAFFE (NG21), pi-GAN (CMK21), and NeRF-VAE (KSZ21), build 2D generative models with NeRF function and differentiable rendering and assume unobserved object-specific variables to follow known Gaussian prior distributions. They are trained by adversarial learning (GPM14) or variational inference (KW14a). Our model is also a NeRF-based generative model, but assumes latent object-specific variables to follow informative prior distributions parameterized by energy-based models (XLZ16b). We propose to train NeRF-based generator and EBM priors simultaneously by likelihood-based learning with either MCMC or amortized inference.

3.3 Background

3.3.1 Neural Radiance Field

A continuous scene can be represented by a Neural Radiance Field (NeRF) (MST22), which is a mapping function f_θ whose input is a 3D location $\mathbf{x} \in \mathbb{R}^3$ and a 3D unit vector as viewing direction $\mathbf{d} \in \mathbb{R}^3$, and whose output is an RGB color value $\mathbf{c} \in \mathbb{R}^3$ and a volume density σ. Formally, $f_\theta : (\mathbf{x}, \mathbf{d}) \to (\mathbf{c}, \sigma)$, where f_θ is a neural network with parameters θ. Given a fixed camera pose, to render a 2D image from the NeRF representation f_θ, we can follow the classical volume rendering method (KH84) to calculate the color of each pixel $\mathbf{v} \in \mathbb{R}^2$ in the 2D image. The color of the pixel is determined by the color and volume density values of all points along the camera ray r that goes through that pixel $\mathbf{v}$. In practice, we can follow (MST22) and sample M points $\{\mathbf{x}_i^r\}_{i=1}^M$from the near to far bounds along the camera ray r and obtain a set of corresponding colors and densities

$\{(\mathbf{c}_i^r, \sigma_i^r)\}_{i=1}^M$ by f_θ, and then we compute the color $C(r)$ for the camera ray r by

$$C(r) = \sum_{i=1}^{M} T_i^r (1 - \exp(-\sigma_i^r \delta_i^r)) \mathbf{c}_i^r, \tag{3.1}$$

where $\delta_i^r = ||\mathbf{x}_{i+1}^r - \mathbf{x}_i^r||_2$ is the distance between adjacent sample points, and $T_i^r = \exp(-\sum_{j=1}^{i-1} \sigma_j^r \delta_j^r)$ is the accumulated transmittance along the ray from the 1st point to the i-th point, i.e., the probability that the ray travels from $\mathbf{x}_1^r$ to $\mathbf{x}_i^r$ without being blocked. To render the whole image $\mathbf{I}$, we need to compute the color for the ray that corresponds to each pixel $\mathbf{v}$ in the image. Let $r(\mathbf{v})$ be the camera ray corresponding to the pixel $\mathbf{v}$, and the rendered image is given by $\mathbf{I}(\mathbf{v}) = C(r(\mathbf{v})), \mathbf{v} \in \mathcal{D}$, where $\mathcal{D}$ is the image domain.

3.3.2 Conditional Neural Radiance Field

The original NeRF function f_θ is a 3D representation of a single scene or object. To generalize the NeRF to represent different scenes or objects, (SLN20) proposes the conditional NeRF function

$$g_\theta : (\mathbf{x}, \mathbf{d}, \mathbf{z}^s, \mathbf{z}^a) \to (\mathbf{c}, \sigma), \tag{3.2}$$

which is conditioned on object-specific variables, $\mathbf{z}^a$ and $\mathbf{z}^s$, corresponding to object appearance and shape respectively. It can be further decomposed into (i) $g_{\theta_1}^1 : (\mathbf{x}, \mathbf{z}^s) \to \mathbf{h}$, (ii) $g_{\theta_2}^2 : (\mathbf{h}, \mathbf{d}, \mathbf{z}^a) \to \mathbf{c}$, and (iii) $g_{\theta_3}^3 : \mathbf{h} \to \sigma$ to show the dependency among the input variables in the design of $g(\theta)$.

3.4 Proposed framework

3.4.1 NeRF-based 2D generator with EBM priors

We are interested in learning a 3D-aware generative model of 2D images, with the purposes of controllable image synthesis and disentangled image representation. We build a top-down 2D image generator based on a conditional NeRF structure for the intrinsic 3D representation of the object in

an image. Let $\mathbf{z}^a$ and $\mathbf{z}^s$ be the latent variables that define the shape and the appearance of an object, respectively. $\mathbf{z}^a$ and $\mathbf{z}^s$ are assumed to be independent. They together specify an object. Let $\boldsymbol{\xi}$ be the camera pose. The generator G_θ consists of an object-conditioned NeRF function g_θ as shown in Eq. (3.2) and a differentiable rendering function as shown in Eq. (3.1). θ are trainable parameters of the generator. Given an object specified by $(\mathbf{z}^a, \mathbf{z}^s)$, the generator takes the camera pose $\boldsymbol{\xi}$ as input and outputs an image by using the NeRF g_θ to render an image from the pose $\boldsymbol{\xi}$ with the render operation in Eq. (3.1). Given a dataset of 2D images of different objects captured from different viewing angles (i.e., different camera pose), in which the camera pose of each image is provided. We assume each image is generated by following the generative process defined by G_θ and each of the latent variables $(\mathbf{z}^a, \mathbf{z}^s)$ is assumed to follow an informative prior distribution that is defined by a trainable energy-based model (EBM). Specifically, the proposed 3D-aware image-based generative model is given by the following deep latent variable model

$$\begin{aligned} \mathbf{I} &= G_\theta(\mathbf{z}^a, \mathbf{z}^s, \boldsymbol{\xi}) + \epsilon, \\ \epsilon &\sim \mathcal{N}(0, \sigma_\epsilon^2 I), \\ \mathbf{z}^a &\sim p_{\alpha_a}(\mathbf{z}^a), \\ \mathbf{z}^s &\sim p_{\alpha_s}(\mathbf{z}^s), \end{aligned} \tag{3.3}$$

where ϵ is the observation residual following a Gaussian distribution $\mathcal{N}(0, \sigma_\epsilon^2 I)$ with a known standard deviation σ_ϵ, and I denotes the identity matrix. Both $p_{\alpha_a}(\mathbf{z}^a)$ and $p_{\alpha_s}(\mathbf{z}^s)$ are modeled by EBMs

$$p_{\alpha_a}(\mathbf{z}^a) = \frac{1}{Z(\alpha_a)} \exp[f_{\alpha_a}(\mathbf{z}^a)] q_0(\mathbf{z}^a), \tag{3.4}$$

$$p_{\alpha_s}(\mathbf{z}^s) = \frac{1}{Z(\alpha_s)} \exp[f_{\alpha_s}(\mathbf{z}^s)] q_0(\mathbf{z}^s), \tag{3.5}$$

which are in the form of exponential tilting of a Gaussian reference distribution $q_0 \sim \mathcal{N}(0, \sigma^2 I)$. (Note that q_0 could be a uniform reference distribution.) $f_{\alpha_a}(\mathbf{z}^a)$ and $f_{\alpha_s}(\mathbf{z}^s)$ are called energy functions, both of which are parameterized by multilayer perceptrons (MLPs) with trainable

parameters α_a and α_s, respectively. The energy function takes the corresponding latent variables as input and outputs a scalar as energy. Besides, $Z(\alpha_a) = \int \exp[f_{\alpha_a}(\mathbf{z}^a)]q_0(\mathbf{z}^a)d\mathbf{z}^a$ and $Z(\alpha_s) = \int \exp[f_{\alpha_s}(\mathbf{z}^s)]q_0(\mathbf{z}^s)d\mathbf{z}^s$ are intractable normalizing constants. Although $q_0(\mathbf{z}^a)$ and $q_0(\mathbf{z}^s)$ are Gaussian distributions, $p_{\alpha_a}(\mathbf{z}^a)$ and $p_{\alpha_z}(\mathbf{x}^s)$ are non-Gaussian priors, where α_a and α_s are learned from the data together with the parameters θ of the generator G_θ.

3.4.2 Learning with MCMC-based inference

For convenience of notation, let $\beta = (\theta, \alpha_a, \alpha_s)$ and $\alpha = (\alpha_a, \alpha_s)$. Given a set of 2D images with known camera poses, i.e., $\{(\mathbf{I}_i, \boldsymbol{\xi}_i), i = 1, ..., n\}$, we can train β by maximizing the observed-data log-likelihood function defined as

$$\begin{aligned} L(\beta) &= \frac{1}{n}\sum_{i=1}^{n} \log p_\beta(\mathbf{I}_i|\boldsymbol{\xi}_i) \\ &= \frac{1}{n}\sum_{i=1}^{n} \log \left[\int p_\alpha(\mathbf{z}_i^a, \mathbf{z}_i^s) p_\theta(\mathbf{I}_i|\mathbf{z}_i^a, \mathbf{z}_i^s, \boldsymbol{\xi}_i) d\mathbf{z}_i^a d\mathbf{z}_i^s\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} \log \left[\int p_{\alpha_a}(\mathbf{z}_i^a) p_{\alpha_s}(\mathbf{z}_i^s) p_\theta(\mathbf{I}_i|\mathbf{z}_i^a, \mathbf{z}_i^s, \boldsymbol{\xi}_i) d\mathbf{z}_i^a d\mathbf{z}_i^s\right], \end{aligned}$$

where $p_\alpha(\mathbf{z}^a, \mathbf{z}^s) = p_{\alpha_a}(\mathbf{z}^a)p_{\alpha_s}(\mathbf{z}^s)$ because $\mathbf{z}^a$ and $\mathbf{z}^s$ are statistically independent, and the latent variables are integrated out in the complete-data log-likelihood. According to the Law of Large Number, maximizing the likelihood $L(\beta)$ is approximately equivalent to minimizing the Kullback-Leibler (KL) divergence between model $p_\beta(\mathbf{I}|\boldsymbol{\xi})$ and data distribution $p_{\text{data}}(\mathbf{I}|\boldsymbol{\xi})$ if the number n of training examples is very large. The gradient of $L(\beta)$ is calculated based on

$$\begin{aligned} \nabla_\beta \log p_\beta(\mathbf{I}|\boldsymbol{\xi}) &= \mathbb{E}_{p_\beta(\mathbf{z}^a, \mathbf{z}^s|\mathbf{I}, \boldsymbol{\xi})}\left[\nabla_\beta \log p_\beta(\mathbf{I}, \mathbf{z}^a, \mathbf{z}^s|\boldsymbol{\xi})\right] \\ &= \mathbb{E}_{p_\beta(\mathbf{z}^a, \mathbf{z}^s|\mathbf{I}, \boldsymbol{\xi})}[\nabla_\beta \log p_{\alpha_a}(\mathbf{z}^a) + \nabla_\beta \log p_{\alpha_s}(\mathbf{z}^s) \\ &\quad + \nabla_\beta \log p_\theta(\mathbf{I}|\mathbf{z}^a, \mathbf{z}^s, \boldsymbol{\xi})], \end{aligned} \tag{3.6}$$

which can be further decomposed into three parts, i.e., the gradients for the EBM prior of object appearance α_a

$$\begin{aligned} &\mathbb{E}_{p_\beta(\mathbf{z}^a,\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})}\left[\nabla_\beta \log p_{\alpha_a}(\mathbf{z}^a)\right] \\ =&-\mathbb{E}_{p_{\alpha_a}(\mathbf{z}^a)}\left[\nabla_{\alpha_a} f_{\alpha_a}(\mathbf{z}^a)\right]+\mathbb{E}_{p_\beta(\mathbf{z}^a|\mathbf{I},\boldsymbol{\xi})}\left[\nabla_{\alpha_a} f_{\alpha_a}(\mathbf{z}^a)\right], \end{aligned} \tag{3.7}$$

the gradients for the EBM prior of object shape α_s

$$\begin{aligned} &\mathbb{E}_{p_\beta(\mathbf{z}^a,\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})}\left[\nabla_\beta \log p_{\alpha_s}(\mathbf{z}^s)\right] \\ =&-\mathbb{E}_{p_{\alpha_s}(\mathbf{z}^s)}\left[\nabla_{\alpha_s} f_{\alpha_s}(\mathbf{z}^s)\right]+\mathbb{E}_{p_\beta(\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})}\left[\nabla_{\alpha_s} f_{\alpha_s}(\mathbf{z}^s)\right], \end{aligned} \tag{3.8}$$

as well as the gradients for the NeRF-based generator θ

$$\begin{aligned} &\mathbb{E}_{p_\beta(\mathbf{z}^a,\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})}\left[\nabla_\beta \log p_\theta(\mathbf{I}|\mathbf{z}^a,\mathbf{z}^s,\boldsymbol{\xi})\right] \\ =&\mathbb{E}_{p_\beta(\mathbf{z}^a,\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})}\left[\nabla_\theta G_\theta(\mathbf{z}^a,\mathbf{z}^s,\boldsymbol{\xi})(\mathbf{I}-G_\theta(\mathbf{z}^a,\mathbf{z}^s,\boldsymbol{\xi}))/\sigma_\epsilon^2\right]. \end{aligned} \tag{3.9}$$

Since the expectations in Eq. (3.7), Eq. (3.8), and Eq. (3.9) are analytically intractable, Langevin dynamics (Nea11), which is a gradient-based MCMC sampling method, is employed to draw samples from the prior distributions (i.e., $p_{\alpha_a}(\mathbf{z}^a)$ and $p_{\alpha_s}(\mathbf{z}^s)$) and the posterior distribution (i.e., $p_\beta(\mathbf{z}^a,\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})$), and then Monte Carlo averages are computed to estimate the expectation terms. As shown in Eq. (3.7) and Eq. (3.8), the update of the EBM prior model α_a (or α_s) is based on the difference between $\mathbf{z}^a$ (or $\mathbf{z}^s$) sampled from the prior distribution $p_{\alpha_s}(\mathbf{z}^a)$ (or $p_{\alpha_s}(\mathbf{z}^s)$) and $\mathbf{z}^a$ (or $\mathbf{z}^s$) inferred from the posterior distribution $p_\beta(\mathbf{z}^a|\mathbf{I},\boldsymbol{\xi})$ (or $p_\beta(\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})$). According to Eq.(3.9), the update of the generator θ relies on $\mathbf{z}^a$ and $\mathbf{z}^s$ inferred from the posterior distribution $p_\beta(\mathbf{z}^a,\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})$. To sample from the prior distributions $p_{\alpha_a}(\mathbf{z}^a)$ and $p_{\alpha_s}(\mathbf{z}^s)$ by Langevin dynamics, we update $\mathbf{z}^a$

and $\mathbf{z}^s$ by

$$\mathbf{z}^a_{t+1} = \mathbf{z}^a_t + \delta \nabla_{\mathbf{z}^a} \log p_{\alpha_a}(\mathbf{z}^a_t) + \sqrt{2\delta}\mathbf{e}^a_t, \tag{3.10}$$

$$\mathbf{z}^s_{t+1} = \mathbf{z}^s_t + \delta \nabla_{\mathbf{z}^s} \log p_{\alpha_s}(\mathbf{z}^s_t) + \sqrt{2\delta}\mathbf{e}^s_t, \tag{3.11}$$

where t indexes the time step, δ is the Langevin step size, and $\mathbf{e}^a_t$ and $\mathbf{e}^s_t$ are independent Gaussian noises that help the MCMC chains to escape from local modes during sampling. The gradients in Eq. (3.10) and Eq. (3.11) are given by

$$\nabla_{\mathbf{z}^a} \log p_{\alpha_a}(\mathbf{z}^a) = \nabla_{\mathbf{z}^a} f_{\alpha_a}(\mathbf{z}^a) - \mathbf{z}^a/\sigma^2, \tag{3.12}$$

$$\nabla_{\mathbf{z}^s} \log p_{\alpha_s}(\mathbf{z}^s) = \nabla_{\mathbf{z}^s} f_{\alpha_s}(\mathbf{z}^s) - \mathbf{z}^s/\sigma^2, \tag{3.13}$$

where $\nabla_{\mathbf{z}^a} f_{\alpha_a}(\mathbf{z}^a)$ and $\nabla_{\mathbf{z}^s} f_{\alpha_s}(\mathbf{z}^s)$ are efficiently computed by back-propagation.

For each observed $(\mathbf{I}, \boldsymbol{\xi})$, we can sample from the posterior $p_\beta(\mathbf{z}^a, \mathbf{z}^s|\mathbf{I}, \boldsymbol{\xi})$ by alternately running Langevin dynamics: we fix $\mathbf{z}^s$ and sample $\mathbf{z}^a$ from $p_\beta(\mathbf{z}^a|\mathbf{z}^s, \mathbf{I}, \boldsymbol{\xi}) \propto p_\beta(\mathbf{I}, \mathbf{z}^a|\mathbf{z}^s, \boldsymbol{\xi})$, and then fix $\mathbf{z}^a$ and sample $\mathbf{z}^s$ from $p_\beta(\mathbf{z}^s|\mathbf{z}^a, \mathbf{I}, \boldsymbol{\xi}) \propto p_\beta(\mathbf{I}, \mathbf{z}^s|\mathbf{z}^a, \boldsymbol{\xi})$. The Langevin sampling step follows

$$\mathbf{z}^a_{t+1} = \mathbf{z}^a_t + \delta \nabla_{\mathbf{z}^a} \log p_\beta(\mathbf{I}, \mathbf{z}^a_t|\mathbf{z}^s_t, \boldsymbol{\xi}) + \sqrt{2\delta}\mathbf{e}^a_t, \tag{3.14}$$

$$\mathbf{z}^s_{t+1} = \mathbf{z}^s_t + \delta \nabla_{\mathbf{z}^s} \log p_\beta(\mathbf{I}, \mathbf{z}^s_t|\mathbf{z}^a_t, \boldsymbol{\xi}) + \sqrt{2\delta}\mathbf{e}^s_t. \tag{3.15}$$

The key steps in Eq. (3.14) and Eq. (3.15) are to compute the gradients of

$$\begin{aligned}
&\log p_\beta(\mathbf{I}, \mathbf{z}^a|\mathbf{z}^s, \boldsymbol{\xi}) = \log[p_{\alpha_a}(\mathbf{z}^a) p_\theta(\mathbf{I}|\mathbf{z}^a, \mathbf{z}^s, \boldsymbol{\xi})] = C_a \\
&\quad - ||\mathbf{I} - G_\theta(\mathbf{z}^a, \mathbf{z}^s, \boldsymbol{\xi})||^2/2\sigma^2_\epsilon + f_{\alpha_a}(\mathbf{z}^a) - ||\mathbf{z}^a||^2/2\sigma^2, \\
&\log p_\beta(\mathbf{I}, \mathbf{z}^s|\mathbf{z}^a, \boldsymbol{\xi}) = \log[p_{\alpha_s}(\mathbf{z}^s) p_\theta(\mathbf{I}|\mathbf{z}^s, \mathbf{z}^a, \boldsymbol{\xi})] = C_s \\
&\quad - ||\mathbf{I} - G_\theta(\mathbf{z}^a, \mathbf{z}^s, \boldsymbol{\xi})||^2/2\sigma^2_\epsilon + f_{\alpha_s}(\mathbf{z}^s) - ||\mathbf{z}^s||^2/2\sigma^2,
\end{aligned}$$

where C_a and C_s are constants independent of $\mathbf{z}^a$, $\mathbf{z}^s$ and θ. After sufficient alternating Langevin

Algorithm 1 Learning NeRF-LEBM with MCMC inference

Input: (1) Images and viewpoints $\{(\mathbf{I}_i, \boldsymbol{\xi}_i)\}_{i=1}^n$; (2) Numbers of Langevin steps for priors and posterior $\{K^-, K^+\}$; (3) Langevin step sizes for priors and posterior$\{\delta^-, \delta^+\}$; (4) Learning rates for priors and generator $\{\eta_\alpha, \eta_\theta\}$.

Output: (1) θ for generator; (2) (α_a, α_s) for EBM priors; (3) Latent variables $\{(\mathbf{z}_i^a, \mathbf{z}_i^s)\}_{i=1}^n$.

1: Randomly initialize θ, α_a, α_s, and $\{(\mathbf{z}_i^a, \mathbf{z}_i^s)\}_{i=1}^n$.
2: **repeat**
3: For each $(\mathbf{I}_i, \boldsymbol{\xi}_i)$, sample the prior of object appearance $\mathbf{z}_i^{a-} \sim p_{\alpha_a}(\mathbf{z}^a)$ and the prior of object shape $\mathbf{z}_i^{s-} \sim p_{\alpha_s}(\mathbf{z}^s)$ using K^- steps of Langevin dynamics with a step size δ^-, which follows Eq. (3.10) and Eq. (3.11), respectively.
4: For each $(\mathbf{I}_i, \boldsymbol{\xi}_i)$, run K^+ Langevin steps with a step size δ^+, to alternatively sample $\mathbf{z}_i^a$ from $p_\beta(\mathbf{z}_i^a|\mathbf{z}_i^s, \mathbf{I}_i, \boldsymbol{\xi}_i)$, while fixing $\mathbf{z}_i^s$; and sample $\mathbf{z}_i^s$ from $p_\beta(\mathbf{z}_i^s|\mathbf{z}_i^a, \mathbf{I}_i, \boldsymbol{\xi}_i)$, while fixing $\mathbf{z}_i^a$.
5: $\alpha_a \leftarrow \alpha_a + \eta_\alpha \nabla_{\alpha_a} L$.
6: $\alpha_s \leftarrow \alpha_s + \eta_\alpha \nabla_{\alpha_s} L$.
7: $\theta \leftarrow \theta + \eta_\theta \nabla_\theta L$.
8: **until** converge

steps, the updated $\mathbf{z}^a$ and $\mathbf{z}^s$ follow the joint posterior $p_\beta(\mathbf{z}^a, \mathbf{z}^s|\mathbf{I}, \boldsymbol{\xi})$, and $\mathbf{z}^a$ and $\mathbf{z}^s$ follow $p_\beta(\mathbf{z}^a|\mathbf{I}, \boldsymbol{\xi})$ and $p_\beta(\mathbf{z}^s|\mathbf{I}, \boldsymbol{\xi})$, respectively.

Let $\mathbf{z}_i^{a-}$ and $\mathbf{z}_i^{s-}$ be the samples drawn from the EBM priors by Langevin dynamics in Eqs. (3.10) and (3.11). Let $\mathbf{z}_i^{a+}$ and $\mathbf{z}_i^{s+}$ be the inferred latent variables of the observation $(\mathbf{I}_i, \boldsymbol{\xi}_i)$ by Langevin dynamics in Eqs. (3.14) and (3.15). The gradients of the log-likelihood L over α_a, α_s, and θ are estimated by

$$\nabla_{\alpha_a} L = -\frac{1}{n}\sum_{i=1}^n \left[\nabla_{\alpha_a} f_{\alpha_a}(\mathbf{z}_i^{a-})\right] + \frac{1}{n}\sum_{i=1}^n \left[\nabla_{\alpha_a} f_{\alpha_a}(\mathbf{z}_i^{a+})\right],$$

$$\nabla_{\alpha_s} L = -\frac{1}{n}\sum_{i=1}^n \left[\nabla_{\alpha_s} f_{\alpha_s}(\mathbf{z}_i^{s-})\right] + \frac{1}{n}\sum_{i=1}^n \left[\nabla_{\alpha_s} f_{\alpha_s}(\mathbf{z}_i^{s+})\right],$$

$$\nabla_\theta L = \frac{1}{n}\sum_{i=1}^n \left[\nabla_\theta G_\theta(\mathbf{z}_i^{a+}, \mathbf{z}_i^{s+}, \boldsymbol{\xi}_i)\frac{\mathbf{I}_i - G_\theta(\mathbf{z}_i^{a+}, \mathbf{z}_i^{s+}, \boldsymbol{\xi}_i)}{\sigma_\epsilon^2}\right].$$

The learning algorithm of the NeRF-LEBM with MCMC inference can be summarized in Algorithm 1.

Algorithm 2 Variational Learning for NeRF-LEBM

Input: (1) Images and viewpoints $\{(\mathbf{I}_i, \boldsymbol{\xi}_i)\}_{i=1}^n$; (2) Number of Langevin steps K^- for priors; (3) Langevin step size for priors δ^-; (4) Learning rates $\{\eta_\alpha, \eta_\omega\}$.

Output: (1) θ for generator; (2) (α_a, α_s) for EBM priors; (3) ϕ for inference net.

1: Randomly initialize θ, ϕ, α_a, and α_s.
2: **repeat**
3: For each $(\mathbf{I}_i, \boldsymbol{\xi}_i)$, sample the priors $\mathbf{z}_i^{a-} \sim p_{\alpha_a}(\mathbf{z}^a)$ and $\mathbf{z}_i^{s-} \sim p_{\alpha_s}(\mathbf{z}^s)$ using K^- Langevin steps with a step size δ^-, which follow Eq. (3.10) and Eq. (3.11) respectively.
4: For each $(\mathbf{I}_i, \boldsymbol{\xi}_i)$, sample $\mathbf{z}^a \sim q_{\phi_a}(\mathbf{z}^a|\mathbf{I}_i, \boldsymbol{\xi}_i)$ and $\mathbf{z}_i^s \sim q_{\phi_s}(\mathbf{z}^s|\mathbf{I}_i, \boldsymbol{\xi}_i)$ using the inference network.
5: $\alpha_a \leftarrow \alpha_a + \eta_\alpha \nabla_{\alpha_a}$ELBO ($\nabla_{\alpha_a}$ELBO is in Eq. (3.18)).
6: $\alpha_s \leftarrow \alpha_s + \eta_\alpha \nabla_{\alpha_s}$ELBO ($\nabla_{\alpha_s}$ELBO is in Eq. (3.19)).
7: $\omega \leftarrow \omega + \eta_\omega \nabla_\omega$ELBO ($\nabla_\omega$ELBO is in Eq. (3.20), where $\omega = (\phi_a, \theta)$).
8: **until** converge

3.4.3 Learning with amortized inference

Even though both prior and posterior sampling require Langevin dynamics. Prior sampling is more affordable than posterior sampling because the network structure of f_{α_a} or f_{α_s} is much smaller than that of the NeRF-based generator G_θ and the posterior sampling need to perform back-propagation on G_θ, which is time-consuming. In this section, we propose to train the NeRF-LEBM by adopting amortized inference, in which the posterior distributions, $p_\beta(\mathbf{z}^a|\mathbf{I}, \boldsymbol{\xi})$ and $p_\beta(\mathbf{z}^s|\mathbf{I}, \boldsymbol{\xi})$, are approximated by separate bottom-up inference networks with reparameterization trick, $q_{\phi_a}(\mathbf{z}^a|\mathbf{I}, \boldsymbol{\xi}) = \mathcal{N}(\mathbf{z}^a|u_{\phi_a}(\mathbf{I}|\boldsymbol{\xi}), \sigma_{\phi_a}(\mathbf{I}|\boldsymbol{\xi}))$ and $q_{\phi_s}(\mathbf{z}^s|\mathbf{I}, \boldsymbol{\xi}) = \mathcal{N}(\mathbf{z}^s|u_{\phi_s}(\mathbf{I}|\boldsymbol{\xi}), \sigma_{\phi_s}(\mathbf{I}|\boldsymbol{\xi}))$, respectively. We denote $\phi = (\phi_a, \phi_s)$ for notation simplicity. The log-likelihood $\log p_\beta(\mathbf{I}|\boldsymbol{\xi})$ is lower

bounded by the evidence lower bound (ELBO), which is given by

$$
\begin{aligned}
&\text{ELBO}(\mathbf{I}|\boldsymbol{\xi};\beta,\phi) \\
=&\log p_\beta(\mathbf{I}|\boldsymbol{\xi}) - \mathbb{D}_{\text{KL}}(q_{\phi_a}(\mathbf{z}^a|\mathbf{I},\boldsymbol{\xi})||p_\beta(\mathbf{z}^a|\mathbf{I},\boldsymbol{\xi})) \\
&- \mathbb{D}_{\text{KL}}(q_{\phi_s}(\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})||p_\beta(\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})) \qquad (3.16) \\
=&- \mathbb{D}_{\text{KL}}(q_{\phi_s}(\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})||p_{\alpha^s}(\mathbf{z}^s)) \\
&- \mathbb{D}_{\text{KL}}(q_{\phi_a}(\mathbf{z}^a|\mathbf{I},\boldsymbol{\xi})||p_{\alpha^a}(\mathbf{z}^a)) \\
&+ \mathbb{E}_{q_{\phi_a}(\mathbf{z}^a|\mathbf{I},\boldsymbol{\xi})q_{\phi_s}(\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})}[\log p_\theta(\mathbf{I}|\mathbf{z}^a,\mathbf{z}^s,\boldsymbol{\xi})], \qquad (3.17)
\end{aligned}
$$

where $\mathbb{D}_{\text{KL}}$ denotes the Kullback-Leibler divergence. We assume $p_{\alpha^s}(\mathbf{z}^s) = p_{\alpha^s}(\mathbf{z}^s|\boldsymbol{\xi})$ and $p_{\alpha^a}(\mathbf{z}^a) = p_{\alpha^a}(\mathbf{z}^a|\boldsymbol{\xi})$. For the EBM prior models, the learning gradients to update α_a and α_s are given by

$$
\begin{aligned}
&\nabla_{\alpha_a}\text{ELBO}(\mathbf{I}|\boldsymbol{\xi};\beta,\phi) \qquad (3.18) \\
=&- \mathbb{E}_{p_{\alpha_a}(\mathbf{z}^a)}\left[\nabla_{\alpha_a} f_{\alpha_a}(\mathbf{z}^a)\right] + \mathbb{E}_{q_{\phi_a}(\mathbf{z}^a|\mathbf{I},\boldsymbol{\xi})}\left[\nabla_{\alpha_a} f_{\alpha_a}(\mathbf{z}^a)\right], \\
&\nabla_{\alpha_s}\text{ELBO}(\mathbf{I}|\boldsymbol{\xi};\beta,\phi) \qquad (3.19) \\
=&- \mathbb{E}_{p_{\alpha_s}(\mathbf{z}^s)}\left[\nabla_{\alpha_s} f_{\alpha_s}(\mathbf{z}^s)\right] + \mathbb{E}_{q_{\phi_s}(\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})}\left[\nabla_{\alpha_s} f_{\alpha_s}(\mathbf{z}^s)\right].
\end{aligned}
$$

Let $\omega = (\phi, \theta)$ be the parameters of the inference networks and generator. The learning gradients of these models are

$$
\begin{aligned}
&\nabla_\omega\text{ELBO}(\mathbf{I}|\boldsymbol{\xi};\beta,\phi) \qquad (3.20) \\
=&\nabla_\omega \mathbb{E}_{q_{\phi_a}(\mathbf{z}^a|\mathbf{I},\boldsymbol{\xi})q_{\phi_s}(\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})}[\log p_\theta(\mathbf{I}|\mathbf{z}^a,\mathbf{z}^s,\boldsymbol{\xi})] \\
-&\nabla_\omega \mathbb{D}_{\text{KL}}(q_{\phi_a}(\mathbf{z}^a|\mathbf{I},\boldsymbol{\xi})||p_0(\mathbf{z}^a)) + \nabla_\omega \mathbb{E}_{q_{\phi_a}(\mathbf{z}^a|\mathbf{I},\boldsymbol{\xi})}[f_{\alpha_a}(\mathbf{z}^a)] \\
-&\nabla_\omega \mathbb{D}_{\text{KL}}(q_{\phi_s}(\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})||p_0(\mathbf{z}^s)) + \nabla_\omega \mathbb{E}_{q_{\phi_s}(\mathbf{z}^s|\mathbf{I},\boldsymbol{\xi})}[f_{\alpha_s}(\mathbf{z}^s)]
\end{aligned}
$$

The first term on the right hand size of Eq.(3.20) is the reconstruction by the bottom-up inference encoders and the top-down generator. The second and the fourth terms are KL divergences between

the inference model and the Gaussian distribution. These three terms form the learning objective of the original VAE. The variational learning of the NeRF-LEBM is given in Algorithm 2.

3.4.4 Learning without ground truth camera pose

Many real world datasets do not contain camera pose information, therefore fitting the models from those datasets by using Algorithms 1 or 2 is not appropriate. In this section, we study learning the NeRF-LFBM model from images without knowing the ground truth camera poses, and generalizing Algorithm 2 to this scenario. We treat the unknown camera pose as latent variables and seek to infer it together with the shape and appearance variables in the amortized learning framework. In our experiments, we assume the camera is located on a sphere and the object is put in the center of the sphere. Therefore, the camera pose $\boldsymbol{\xi}$ can be interpreted as the altitude angle ξ_1 and azimuth angle ξ_2. However, different from the shape and the appearance, the camera pose is directional and can be better explained through a spherical representation (Mar75). For implementation, instead of directly representing each individual angle, we represent its Sine and Cosine values that directly construct the corresponding rotation matrix that is useful for subsequent computation. Thus, each rotation angle ξ_i is a two-dimensional unit norm vector located on a unit sphere.

Following the hyperspherical VAE in (DFC18), we use the von Mises-Fisher (vMF) distribution to model the posterior distribution of ξ. vMF can be seen as the Gaussian distribution on a hypershere. To model a hypershpere of dimension m, it is parameterized by a mean direction $\boldsymbol{\mu} \in \mathbb{R}^m$ and a concentration parameter $\kappa \in \mathbb{R}_{\geq 0}$. The probability density of the vMF is defined as $p_{\text{vMF}}(\xi|\boldsymbol{\mu}, \kappa) = \mathcal{C}_m(\kappa)\exp(\kappa\boldsymbol{\mu}^T\boldsymbol{\xi})$, where $\mathcal{C}_m(\kappa) = \frac{\kappa^{m/2-1}}{(2\pi)^{m/2}\mathcal{I}_{m/2-1}(\kappa)}$, with $\mathcal{I}_l$ denoting modified Bessel function of the first kind at order l. For each angel ξ in $\boldsymbol{\xi}$, we design an inference model as $q_{\phi_\xi}(\xi|\mathbf{I}) = p_{\text{vMF}}(\xi|\boldsymbol{\mu}_{\phi_\xi}(\mathbf{I}), \kappa_{\phi_\xi}(\mathbf{I}))$, where $\boldsymbol{\mu}_{\phi_\xi}(\mathbf{I})$ and $\kappa_{\phi_\xi}(\mathbf{I})$ are bottom-up networks with parameters ϕ_ξ that maps $\mathbf{I}$ to $\boldsymbol{\mu}$ and κ. We assume the prior of ξ to be a uniform distribution on the unit sphere (denoted as $U(S^{m-1})$), which is the special case of vMF with $\kappa = 0$. The key to use the amortized inference

is to compute the KL divergence between the posterior and the prior, which can follow

$$\begin{aligned}&\mathbb{D}_{\text{KL}}(p_{\text{vMF}}(\boldsymbol{\mu},\kappa)||U(S^{m-1}))\\=&\kappa\frac{\mathcal{I}_{m/2}(\kappa)}{\mathcal{I}_{m/2-1}(\kappa)}+\log\mathcal{C}_m(\kappa)-\log\left(\frac{2(\pi^{m/2})}{\Gamma(m/2)}\right)^{-1}.\end{aligned}\tag{3.21}$$

Besides, to compute the ELBO, we need to draw samples from the inference model $q_{\phi_\xi}(\xi|\mathbf{I})$, which amounts to sampling from the vMF distribution. We follow the sampling procedure in (DFC18) for this purpose in our implementation.

3.5 Experiments

3.5.1 Datasets

To evaluate the proposed NeRF-LEBM framework and the learning algorithms, we conduct experiments on three datasets. The Carla dataset is rendered by (SLN20) using the Carla Driving Simulator (DRC17). It contains 10k cars of different shapes, colors and textures. Each car has one 2D image rendered from one random camera pose. Another dataset is the ShapeNet (CFG15) Car dataset, which contains 2.1k different cars for training and 700 cars for testing. We use the images rendered by (SZW19) and follow its split to separate the training and testing sets. Each car in the training set has 250 views and we only use 50 views of them for training. Each car in the testing set has 251 views. each image is associated with its camera pose information.

3.5.2 Training details

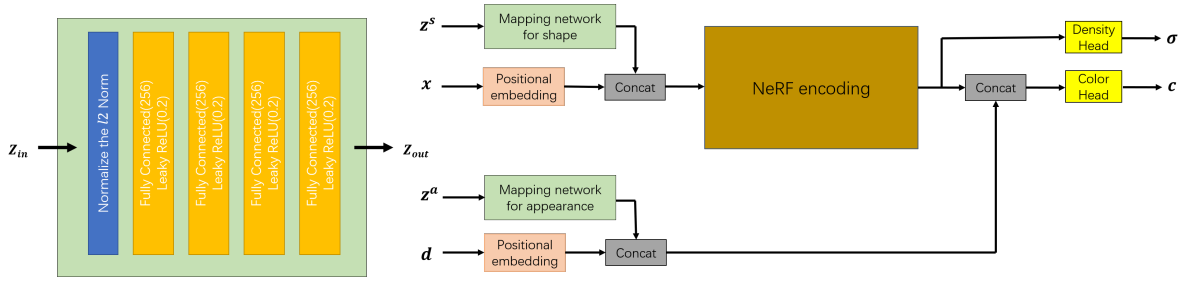
In this section, we describe the network structure designs and hyper-parameter settings in our experiments.

3.5.2.1 Network structure for conditional NeRF-based generator

For the experiments where camera poses of objects are given, the structure of the NeRF-based generator is shown in Figure 3.1. For this structure, we mainly follow the design in (SLN20). However, for each of the latent vectors $\mathbf{z}^s$ and $\mathbf{z}^a$, we add a mapping network to transform it before concatenating it with the positional embedding. This mapping network is composed of a normalizing layer and 4 linear layers, each of which is followed by a Leaky ReLU activation function. The concatenated features then enter the NeRF encoding module, which is an 8-layer MLP. The number of dimensions of each hidden layer or the output layer is 256. The output of the NeRF encoding module is then used to predict the color $\mathbf{c}$ and density σ information. For the experiments about learning without camera pose information on the CelebA dataset, inspired by (CMK21), we design a network structure with the FiLM-conditioned layers (PSV18; DPS18). To be more specific, we input the transformed $\mathbf{z}^s$ to the first 4 layers of the generator and input the transformed $\mathbf{z}^a$ to the color head using the FiLM layer. The detailed architecture is shown in Figure 3.2.

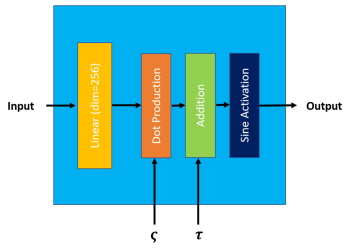
3.5.2.2 Network Structure for latent space EBM

We use two EBM structures. One (small version) contains 2 layers of linear transformations, each of which followed by a swish activation, while the other (large version) contains 4 layers of linear transformations with swish activation layers. The network structure is shown in Table 3.1. The choice of the EBM prior model for each experiment is shown in Table 3.2.

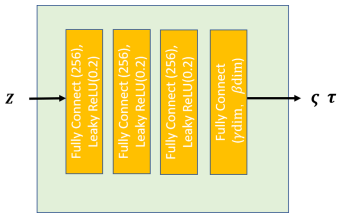


(a) Mapping network (b) Conditional NeRF-based generator.

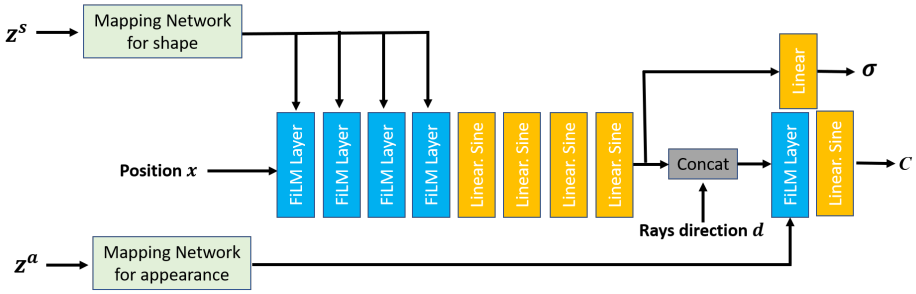
Figure 3.1: Model structure for the NeRF-LEBM generator with known camera pose.



(a) The FiLM layer.



(b) Mapping network.



(c) Conditional NeRF-based generator structure.

Figure 3.2: Model structure for the NeRF-LEBM generator with unknown camera pose.

3.5.2.3 Network structure for inference model

In the experiments using amortized inference, we need an inference model to infer the latent vectors (i.e., appearance, shape, and camera pose). Inspired by (BGK18), we use the ResNet34 (HZR16) structure as the feature extractor and build separate inference heads on the top of it for different latent vectors. Each inference head is composed of a couple of MLP layers. We let the inference models for $\mathbf{z}^a$, $\mathbf{z}^s$ and camera pose to share the same feature extractor and only differ in their inference heads. When we update the parameters of the inference model, we use a learning rate 3×10^{-5} for the feature extractor and the pose head while using a smaller learning rate 5×10^{-6} for the inference heads of $\mathbf{z}^a$ and $\mathbf{z}^s$.

3.5.3 Training hyperparameters

During our training, we use the Adam(KB15a) optimizer and we set $\beta_{\text{Adam}} = (0.9, 0.999)$. We set the standard deviation of the residual in Eq. (3.3) $\sigma_\epsilon = 0.05$. We disable the noise term in

the Langevin inference for a better performance in the MCMC inference case. For the persistent chain MCMC inference setting, we use the Adam instead of a noise-disable Langevin dynamics to infer the latent vectors. As to the MCMC sampling for the EBM priors, we assign a weight as a hyperparameter to the noise term in the Langevin dynamics for adjusting its magnitude in our practise. Please check Table 3.3 for hyperparameter settings, including learning rate, batch size, etc, used in other experiments.

Table 3.1: Model structures of the latent EBMs with hidden dimension C.

(a) EBM (small)
Linear C, swish
Linear C, swish
Linear 1

(b) EBM (large)
Linear C, swish
Linear C, swish
Linear C, swish
Linear C, swish
Linear 1

Table 3.2: Choice of the EBM prior in each experiment.

Experiment	EBM prior for shape			EBM prior for appearance		
	type	vector dimension	hidden dimension	type	vector dimension	hidden dimension
Carla / MCMC inference / with poses	large	128	256	large	128	256
Carla / amortized inference / with poses	small	128	256	small	128	128
ShapeNet Car / MCMC inference / with poses	large	128	256	large	128	256
Carla / amortized inference / without poses	small	128	128	small	128	64

Table 3.3: Hyperparameter settings in different experiments.

Experiments	batch size	Number of views	η_α	η_θ	η_ϕ	q_0	δ^+	K^+	δ^-	K^-	MCMC noise weight (EBM)
Carla / MCMC inference / with poses	8	1	2e-5	1e-4	-	Uniform	0.1	60	0.5	60	0.02
Carla / amortized inference / with poses	8	1	7e-6	1e-4	1e-4	Normal	-	-	0.5	60	1.0
ShapeNet Car / persistent MCMC inference / with poses	12	2	2e-5	1e-4	-	Normal	1e-4	1	0.5	40	0.0
Carla / amortized inference / without poses	16	1	7e-6	3e-5	3e-5	Normal	-	-	0.5	60	0.0

(a) NeRF-LEBM with MCMC inference (b) NeRF-LEBM with amortized inference (c) NeRF-Gaussian with MCMC inference

Figure 3.3: Images generated by the NeRF-LEBM models trained on the Carla dataset, where the camera poses are given. (a) MCMC inference (b) amortized inference. (c) Baseline NeRF-Gaussian model with MCMC inference.

3.5.4 Random image synthesis

We first evaluate the capability of image generation of the NeRF-LEBM on the Carla dataset, where the camera pose information is available. We try to answer whether the latent space EBMs can capture the underlying factors of objects in images and whether it is better than the Gaussian prior. We train our models on images of resolution 64×64 through both MCMC-based inference in Algorithm 1 and amortized inference in Algorithm 2. Once a model is trained, we can generate new images by first randomly sampling $(\mathbf{z}^a, \mathbf{z}^s)$ from the learned EBM priors and a camera pose $\boldsymbol{\xi}$ from a uniform distribution, and then using the NeRF-based generator to map the sampled latent variables to the image space. The synthesized images by NeRF-LEBM using MCMC inference and amortized inference are displayed in Figures 3.3a and 3.3b, respectively. We can see the learned models can generate meaningful and highly diversified cars with different shapes, appearances and camera poses. To quantitatively evaluate the generative performance, we compare our NeRF-LEBMs with some baselines in terms of Fréchet inception distance (FID) (HRU17a) in Table 3.4. The baselines include the NeRF-VAE (KSZ21), which is a NeRF-based generator using Gaussian prior and trained with variational learning, and the NeRF-Gaussian-MCMC, which is a NeRF-based generator using MCMC inference and Gaussian prior. To make a fair comparison, we implement

the NeRF-VAE using the same NeRF-based generator and inference network as those in our NeRF-LEBM using amortized inference, except that the NeRF-VAE and the NeRF-Gaussian-MCMC only adopt Gaussian priors for latent variables. We compute FID using 10k samples. Table 3.4 shows that NeRF-LEBMs perform very well in the sense that the learned models can generate realistic images. Especially, the NeRF-EBM trained with amortized inference obtains the best performance. The comparison between the NeRF-VAE and our NeRF-LEBM using amortized inference demonstrates the effectiveness of the EBM priors. The efficacy of the EBM priors is also validated by the comparison between the NeRF-Gaussian-MCMC and our NeRF-LEBM using MCMC inference. We show the generated examples from the NeRF-Gaussian-MCMC in Figure 3.3c. Comparing the results in Figure 3.3c with those in Figure 3.3, we can see that examples obtained from the simple Gaussian prior have less diversity than those from the EBM prior. This is consistent with the numerical evaluation shown Table 3.4 and demonstrates the advantage of using latent EBMs as informative prior distributions for modeling latent variables.

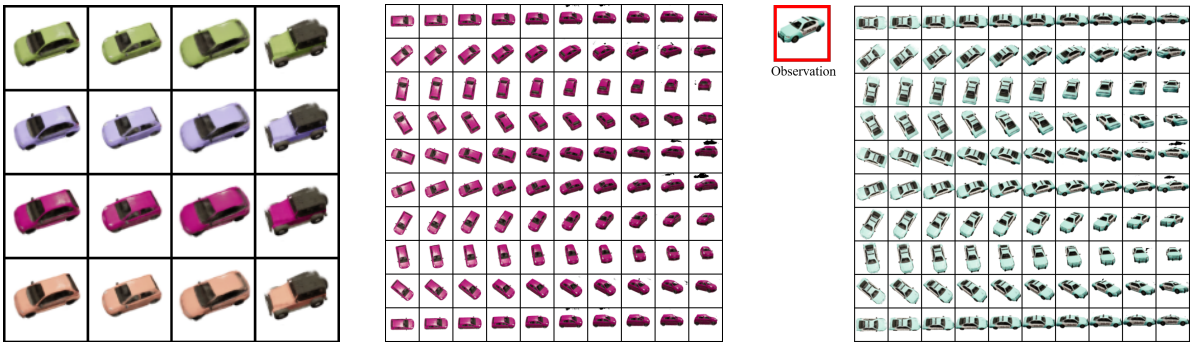
3.5.5 Disentangled representation

We investigate the ability of disentanglement of the NeRF-LEBM. We test the model using amortized inference trained in Section 3.5.4. We generate images by varying one of the three latent vectors, i.e., $\mathbf{z}^a$, $\mathbf{z}^s$ and $\boldsymbol{\xi}$, while fixing the other two, and observe how the manipulated vector influences the generated images. The generated images are shown in Figure 3.4. In Figure 3.4a, the objects

Table 3.4: Comparing the NeRF-LEBMs with likelihood-based baselines on the 64×64 Carla dataset for random image synthesis. The image qualities are evaluated via FID.

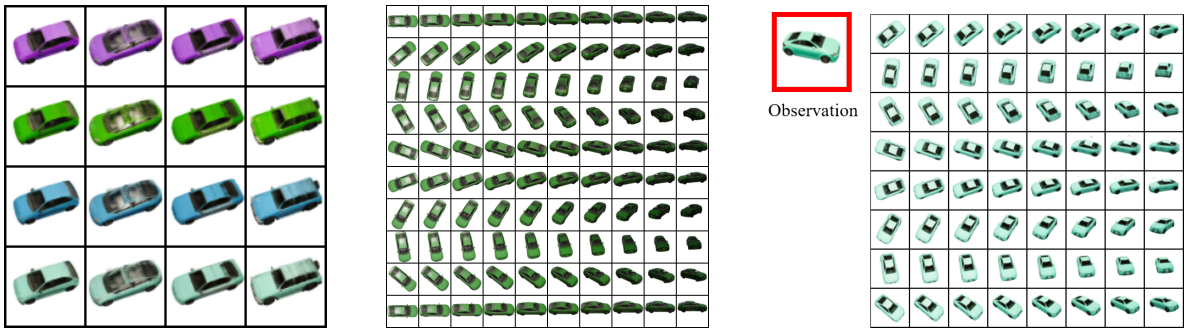
Likelihood-based Model	FID $\downarrow$
NeRF-Gaussian-MCMC	54.13
NeRF-VAE (KSZ21)	38.15
NeRF-LEBM (MCMC inference)	37.19
NeRF-LEBM (Amortized inference)	**20.84**

in each row share the same appearance vector and camera pose but have different shape vectors, while the objects in each column share the same shape vector and camera pose but have different appearance vectors. From Figure 3.4a, we can see that the shape latent vectors do not encode any appearance information, such as color. The colors in the generated images only depend on the appearance latent vectors, and are not influenced by the shape latent vectors. Figure 3.4b displays the synthesized images sharing the same appearance and shape vectors but having different camera poses. Figures 3.4a and 3.4b show that the learned model can successfully disentangle the appearance, shape and camera pose of an object because $\mathbf{z}^a$, $\mathbf{z}^s$ and $\boldsymbol{\xi}$ can, respectively, control the appearance, shape, and viewpoint of the generated images. We can also perform novel view synthesis of a seen 2D object by first inferring its appearance and shape vectors and then using different camera poses to generate different views of the object. Figure 3.4c shows one example. We show the results obtained by a model trained with MCMC inference Figure 3.5.



(a) (b) (c)

Figure 3.4: Disentangled representation. The generated images are obtained by the learned NeRF-LEBM using amortized inference on the Carla dataset. (a) shows the influences of the shape vector $\mathbf{z}^s$ and the appearance vector $\mathbf{z}^a$ in image synthesis. The objects in each row share the same appearance vector $\mathbf{z}^a$ and camera pose $\boldsymbol{\xi}$ but have different shape vectors $\mathbf{z}^s$, while the objects in each column share the same shape vector $\mathbf{z}^s$ and camera pose $\boldsymbol{\xi}$ but have different appearance vectors $\mathbf{z}^a$. (b) demonstrates the effect of the camera pose variable $\boldsymbol{\xi}$ by varying it while fixing the shape and appearance vectors for a randomly sampled object. (c) shows an example of novel view synthesis for an observed 2D image.



(a) (b) (c)

Figure 3.5: Disentangled representation. The generated images are obtained by the learned NeRF-LEBM using MCMC inference on the Carla dataset. (a) shows the influences of the shape vector $\mathbf{z}^s$ and the appearance vector $\mathbf{z}^a$ in image synthesis. The objects in each row share the same appearance vector $\mathbf{z}^a$ and camera pose $\boldsymbol{\xi}$ but have different shape vectors $\mathbf{z}^s$, while the objects in each column share the same shape vector $\mathbf{z}^s$ and camera pose $\boldsymbol{\xi}$ but have different appearance vectors $\mathbf{z}^a$. (b) demonstrates the effect of the camera pose variable $\boldsymbol{\xi}$ by varying it while fixing the shape and appearance vectors for a randomly sampled object. (c) shows an example of novel view synthesis for an observed 2D image.

3.5.6 Inferring 3D structures of unseen 2D objects

Once a NeRF-LEBM model is trained, it is capable of inferring the 3D structure of a previously unseen object from only a few observations. Following (SZW19), we first train our model on the 128×128 resolution ShapeNet Car training set and then apply the trained model to a two-shot novel view synthesis task, in which only two views of an unseen car in the holdout testing set are given to synthesize novel views of the same car. We use the NeRF-LEBM with MCMC inference in this task. To reduce the computational cost, we follow (XGZ19) to adopt a persistent MCMC chain setting for the Langevin inference. Given a unseen object, we first infer its latent appearance and shape vectors via 600 steps of MCMC guided by the posterior distribution, and then we generate novel views of the same object with randomly sampled camera poses. The qualitative results of novel view synthesis are shown in Figure 3.6. In Table 3.5, we compare our NeRF-LEBM with baselines, such as GQN (EJB18) and the NeRF-VAE, in terms of PSNR. Our model has the best performance. We also demonstrate the generative ability of our model by showing the generated images in Figure 3.9. For a quantitative comparison, the FID for our NeRF-LEBM is 94.486 while

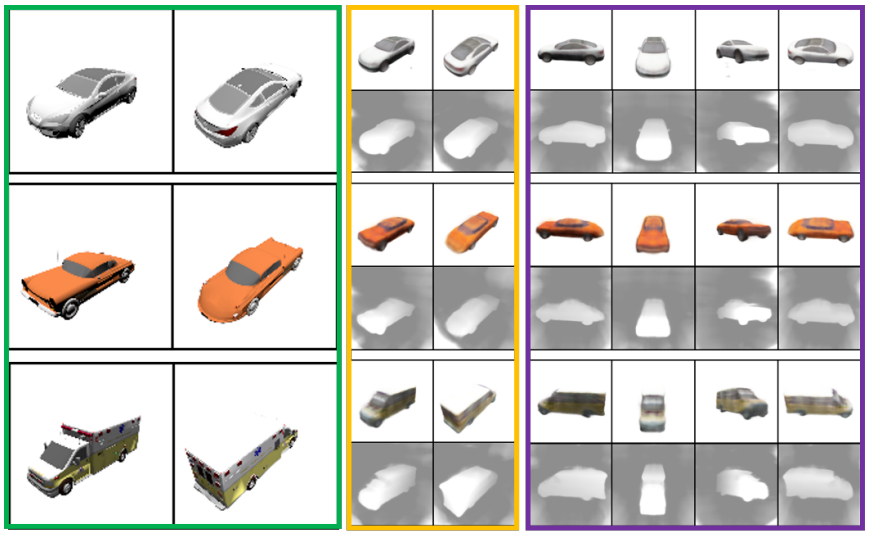


Figure 3.6: Two-shot novel view synthesis results on (128×128) the ShapeNet Car testing set. The left two columns displays two views of unseen cars in testing set. The middle two columns show the reconstruction results obtained by a model trained on training set. The right four columns shows novel view synthesis for the unseen cars. For each reconstructed or synthesized image, we show an RGB image at the first row and the inverse depth map at the second row.

the one obtained by the NeRF-VAE is 112.465.

Table 3.5: Comparing the NeRF-LEBM with other baselines on two-shot novel view synthesis of unseen objects in terms of PSNR. Models are tested on the ShapeNet Car dataset.

Generative Models	PSNR ↑
GQN (EJB18)	18.79
NeRF-VAE (KSZ21)	18.37
NeRF-LEBM (MCMC inference)	**20.28**

3.5.7 Learning from incomplete 2D observations

To show the advantage of the MCMC-based inference, we test our model on a task where observations are incomplete or masked. To create a dataset for this task, we firstly randomly select 500 cars from the original ShapeNet Car dataset, and then for each of them, we use 50 views

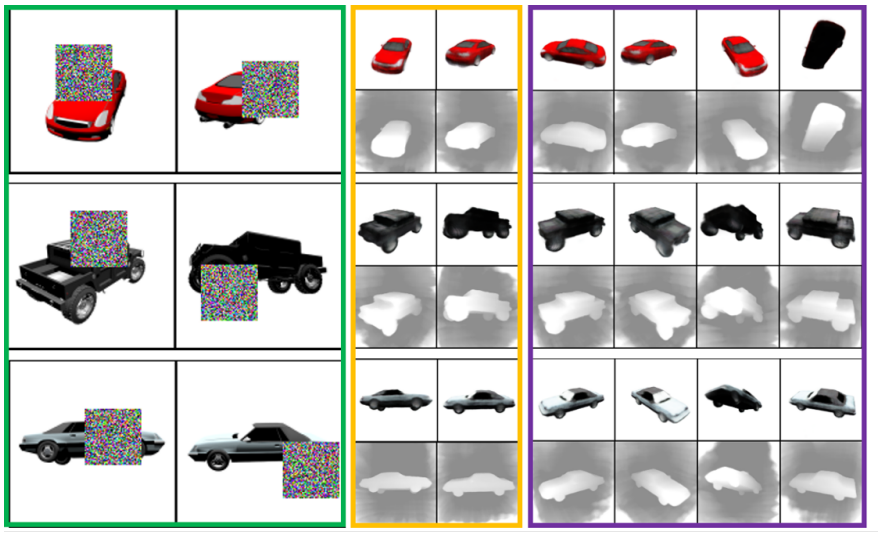
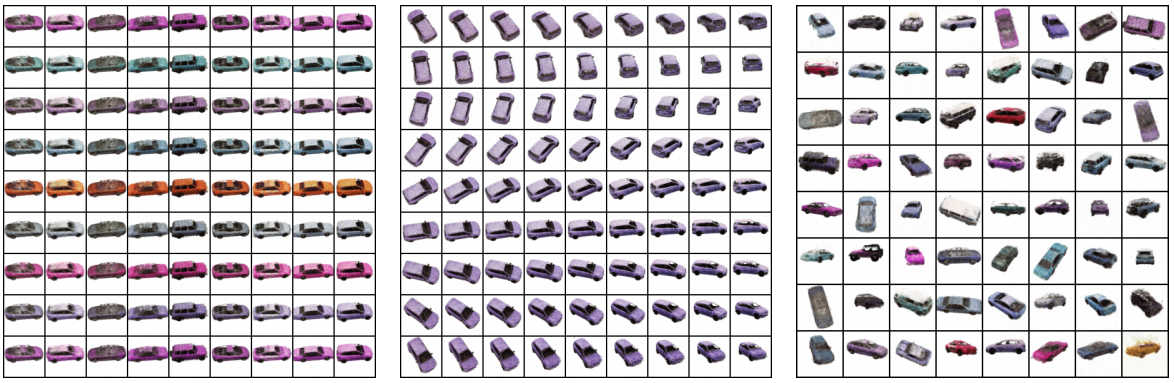


Figure 3.7: Learning from incomplete 2D observations. The left two columns show some examples of the incomplete observations in the training set. The middle two columns show the corresponding recovery results obtained by the learning algorithm. The right four columns are synthesis results for unobserved views of the same objects. For each recovery or synthesized result, we show an RGB 2D image at the first row and an inverse depth map at the second row.

for training and 200 views for testing. For each training image, we randomly mask an area with Gaussian noise (see Figure 3.7). To enable our model to learn from incomplete data, we only maximize the data likelihood computed on the unmasked areas of the training data. This only leads to a minor modification in Algorithm 1 involving the computation of $||\mathbf{I} - G_\theta(\mathbf{z}^a, \mathbf{z}^s, \boldsymbol{\xi})||^2$ in the likelihood term. For a partially observed image, we compute it by summing over only the visible pixels. The latent variables can still be inferred by explaining the visible parts of the incomplete observations, and the model can still be updated as before. In each iteration, we feed our model with two randomly selected observations of each object. Qualitative results in Figure 3.7 demonstrate that our algorithm can learn from incomplete images while recovering the missing pixels, and the learned model can still perform novel view synthesis. We quantitatively compare our NeRF-LEBM with the NeRF-VAE in Table 3.6. Our model beats the baseline using the same generator in the tasks of novel view synthesis and image generation.



(a) (b) (c)

Figure 3.8: Synthesis results on 64×64 Carla dataset without camera poses. (a) The objects in each row share the same appearance vector $\mathbf{z}^a$ but have different shape vectors $\mathbf{z}^s$ while the objects in each column share the same $\mathbf{z}^s$ but have different $\mathbf{z}^a$. They all share the same camera pose $\boldsymbol{\xi}$. (b) The Effect of changing the camera pose while fixing the shape and appearance vectors for a sampled object. (c) Generated samples by randomly sampling $\mathbf{z}^a$, $\mathbf{z}^s$ and camera pose $\boldsymbol{\xi}$.

Table 3.6: Comparison on tasks of novel view synthesis and image generation after learning the models from incomplete 128×128 2D observations on a masked ShapeNet Car dataset. PSNRs and FIDs are reported to measure the model performance on the two tasks, respectively.

Model	PSNR ↑	FID ↓
NeRF-VAE	21.26	128.27
NeRF-LEBM	**24.95**	**105.82**

3.5.8 Learning with unknown camera poses

We study the scenario of training the NeRF-LEBM from 2D images without known camera poses. We assume the camera to locate on a sphere and the object is in the center of this sphere. Thus, we need to infer the altitude and azimuth angles for each observed image. As discussed in Section 3.4.4, we introduce a camera pose inference network to approximate the posterior of the latent camera pose of each observation, and train EBM priors, NeRF-based generator and inference networks simultaneously via amortized inference. In practice, we let the inference networks for camera pose $\boldsymbol{\xi}$, appearance $\mathbf{z}^a$ and shape $\mathbf{z}^s$ to share the lower layers and only differ in their prediction heads. We carried out an experiment on the Carla dataset with a resolution of 64×64. Unlike in Section 3.5.4,

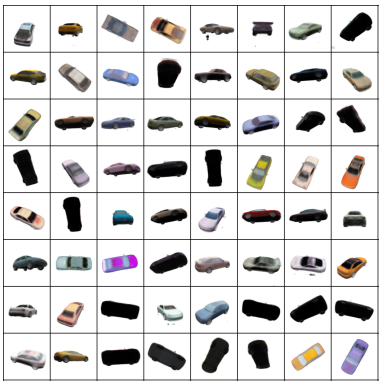
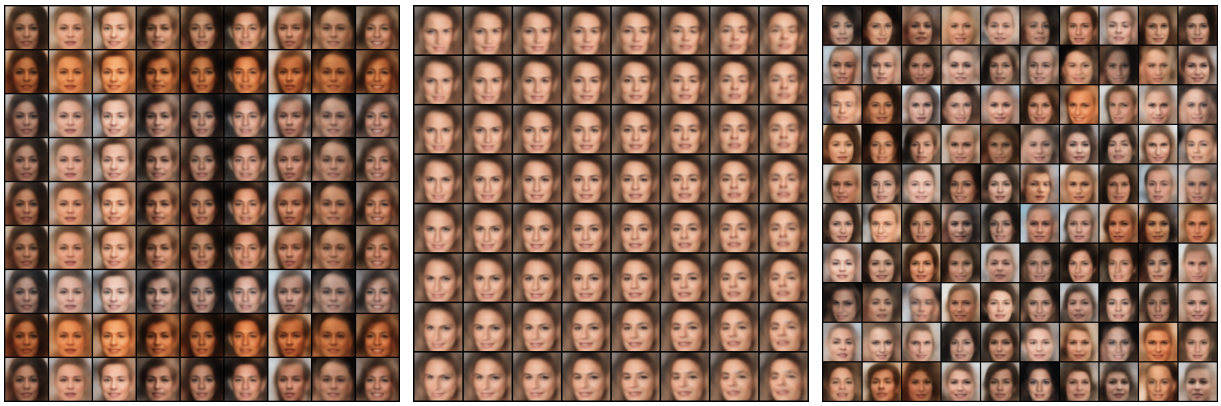


Figure 3.9: Synthesis by the NeRF-LEBM trained on the ShapeNet Car data, with persistent chain MCMC inference.

here we only use the rendered images and do not use the ground truth camera pose associated with each image. The results on are shown in Figure 3.8. From Figure 3.8a, we can see the learned model can disentangle shape and appearance factors. Figure 3.8b shows that the learned model can factor out the camera pose from the data in an unsupervised manner. Figure 3.8c shows some random synthesized examples generated from the model by randomly sampling $\mathbf{z}^a$, $\mathbf{z}^s$ and $\boldsymbol{\xi}$. Although the camera pose can take a valid value from $[0, 2\pi)$, the altitude and azimuth angles of the training examples in the Carla dataset might only lie in limited ranges. Thus, after training, we estimate a camera pose distribution using 10,000 training examples and when we generate images, we sample camera poses according to this distribution. Please refer to Figure 3.10 for more results on the CelebA dataset. These results verify that our model can learn meaningful 3D-aware 2D image generator without known camera poses.

3.6 Complexity analysis

We show a comparison of different models in terms of model size in Table 3.7a. By comparing our NeRF-LEBM using amortized inference with the NeRF-VAE which uses a Gaussian prior, we can find that introducing a latent EBM as a prior only slightly increase the number of parameters. If we rely on the MCMC of inference, then we can save a lot of parameters. In Table 3.7b, we compare



(a) (b) (c)

Figure 3.10: Synthesis results on 64×64 CelebA dataset without knowing camera poses. (a) The objects in each row share the same appearance vector $\mathbf{z}^a$ but have different shape vectors $\mathbf{z}^s$ while the objects in each column share the same $\mathbf{z}^s$ but have different $\mathbf{z}^a$. They all share the same camera pose $\boldsymbol{\xi}$. (b) The Effect of changing the camera pose while fixing the shape and appearance vectors for a sampled object. (c) Generated samples by randomly sampling $\mathbf{z}^a$, $\mathbf{z}^s$ and camera pose $\boldsymbol{\xi}$.

the computational time of randomly sampling 100 images with a resolution of 128×128. The time is computed when the algorithm is run on a single NVIDIA RTX A6000 GPU. Comparing with the Gaussian prior, the EBM prior (with a 40-step MCMC sampling) barely affects the sampling time. That is because the latent EBM is much less computationally-intensive than the NeRF-based generator and we can sample 100 random variables in a batch altogether. We also compare the computational time of the amortized inference and MCMC inference in the few-shot inference of an unseen object in Table 3.7c, from which we can see that using an inference model can do a much quicker inference but the reconstruction results may be worse. On the other hand, MCMC inference may take some time but the results can be better.

Table 3.7: Complexity analysis

(a) Model size

Model	# parameters
NeRF-VAE	24.06 M
NeRF-LEBM (amortized)	24.52 M
NeRF-LEBM (MCMC)	2.21 M

(b) Synthesis time

Model	Time
EBM prior	40.97s
Gaussian prior	40.73s

(c) Inference time

Model	Time
Amortized inference	0.1s
MCMC inference	168s

CHAPTER 4

Improving EBM Training: Cooperative Learning of Langevin Flow and Normalizing Flow Toward Energy-Based Model

4.1 Motivation and Introduction

In the previous chapter, we demonstrated the power of Energy-Based Models (EBMs) through the task of 3D-aware image synthesis. When training with maximum likelihood estimation (MLE), evaluating the gradient of the log-likelihood typically requires Markov chain Monte Carlo (MCMC) sampling (BZ20), such as Langevin dynamics (Nea11), to generate samples from the current model. This is due to the intractable integral needed to compute the normalizing constant. However, Langevin sampling on a highly multi-modal energy function, often parameterized by deep networks, is generally challenging and may require many steps for the samples to reach the target distribution. Additionally, when sampling from a density with a multi-modal landscape, Langevin dynamics, which follows the gradient information, is prone to getting trapped in local modes of the density, making it unlikely to jump out and explore other isolated modes. As a result, the estimated gradient of the likelihood may be biased due to non-mixing MCMC samples, leading to a learned EBM that cannot accurately approximate the data distribution. This naturally raises the question of whether we can improve the training of EBMs.

Recently, (NHZ19) proposed training an EBM with short-run, non-convergent Langevin dynamics. They demonstrated that even though the energy function is technically invalid, the short-run MCMC can be treated as a valid flow-like model that generates realistic examples. This provides an explanation for why an EBM with non-convergent MCMC can still synthesize realistic examples and

suggests a more practical, computationally affordable way to learn useful generative models within existing energy-based frameworks. Although EBMs have been widely applied across different domains, learning short-run MCMC in the context of EBMs remains underexplored.

In this chapter, we focus on improving the sampling quality of EBMs. We acknowledge that MCMC sampling may not mix well in practice, and therefore, we temporarily abandon the goal of training a valid EBM for the data. Instead, we treat the short-run non-convergent Langevin dynamics, which shares parameters with the energy function, as a flow-like transformation that we call the Langevin flow. This can be considered a noise-injected residual network. Implementing a short-run Langevin flow is surprisingly simple, involving the design of a bottom-up ConvNet for the energy function. However, it may still require a sufficiently large number of Langevin steps (each consisting of one gradient descent step and one diffusion step) to construct an effective Langevin flow that is expressive enough to represent the data distribution. To reduce the number of Langevin steps for computational efficiency, we propose the CoopFlow model, which trains a Langevin flow jointly with a normalizing flow model in a cooperative learning scheme (XLG20).

Normalizing flows (DKB15; DSB17; KD18) represent a class of generative models that build complex distributions by applying a series of invertible and differentiable mappings to a simple probability density, such as a Gaussian distribution. These models are increasingly favored for density estimation (KD18; HCS19; YHH19; PVC19; KBE20) and variational inference (RM15; KSJ16) due to their ability to precisely calculate exact log-likelihoods and their efficient inference and synthesis processes. Nevertheless, to maintain the capability for closed-form density evaluations, normalizing flows often employ specially designed transformation sequences that may limit their expressive capabilities.

In our CoopFlow model, the normalizing flow acts as a rapid sampler to initialize the Langevin flow, allowing the Langevin flow to be shorter. Meanwhile, the Langevin flow teaches the normalizing flow through short-run MCMC transitions towards the EBM, enabling the normalizing flow to accumulate the temporal differences in the transition and provide better initial samples. Compared to the original cooperative learning framework (XLG20), which combines the MLE

algorithm of an EBM and a generator, CoopFlow benefits from using a normalizing flow instead of a generic generator because the MLE of a normalizing flow generator is much more tractable. Other generic generators might need MCMC-based inference to evaluate the posterior distribution or another encoder network for variational inference. Additionally, the Langevin flow in CoopFlow can overcome the expressivity limitations of the normalizing flow caused by the invertibility constraint, which further motivates the study of the CoopFlow model. In summary, in this chapter:

- We propose improving the sampling quality of EBMs by pairing an EBM with a normalizing flow model as a rapid sampler. We refer to our model as CoopFlow.
- We design the algorithm that cooperatively trains the normalizing flow and EBM through MCMC teaching.
- During sampling, the normalizing flow makes the initial proposal, and the EBM acts as a short-run Langevin flow sampler.
- Through various experiments, we demonstrate that the normalizing flow, together with the Langevin flow towards the EBM, can perform as a valid generator with strong generative abilities.

4.2 Two Flow Models

4.2.1 Langevin Flow

Energy-based model. Let $x \in \mathbb{R}^D$ be the observed signal such as an image. An energy-based model defines an unnormalized probability distribution of x as follows:

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp[f_\theta(x)], \tag{4.1}$$

where $f_\theta : \mathbb{R}^D \to \mathbb{R}$ is the negative energy function and defined by a bottom-up neural network whose parameters are denoted by θ. The normalizing constant or partition function $Z(\theta) =$

$\int \exp[f_\theta(x)]dx$ is analytically intractable and difficult to compute due to high dimensionality of x.

Langevin dynamics as MCMC. Generating synthesized examples from $p_\theta(x)$ can be accomplished with a gradient-based MCMC such as Langevin dynamics, which is applied as follows

$$x_t = x_{t-1} + \frac{\delta^2}{2}\nabla_x f_\theta(x_{t-1}) + \delta\epsilon_t; \quad x_0 \sim p_0(x), \quad \epsilon_t \sim \mathcal{N}(0, I_D), \quad t = 1, \cdots, T, \tag{4.2}$$

where t indexes the Langevin time step, δ denotes the Langevin step size, ϵ_t is a Brownian motion that explores different modes, and $p_0(x)$ is a uniform distribution that initializes MCMC chains.

Langevin flow. As $T \to \infty$ and $\delta \to 0$, x_T becomes an exact sample from $p_\theta(x)$ under some regularity conditions. However, it is impractical to run infinite steps with infinitesimal step size to generate fair examples from the target distribution. Additionally, convergence of MCMC chains in many cases is hopeless because $p_\theta(x)$ can be very complex and highly multi-modal, then the gradient-based Langevin dynamics has no way to escape from local modes, so that different MCMC chains with different starting points are unable to mix. Let $\tilde{p}_\theta(x)$ be the distribution of x_T, which is the resulting distribution of x after T steps of Langevin updates starting from $x_0 \sim p_0(x)$. Due to the fixed $p_0(x)$, T and δ, the distribution $\tilde{p}_\theta(x)$ is well defined, which can be implicitly expressed by

$$\tilde{p}_\theta(x) = (\mathcal{K}_\theta p_0)(x) = \int p_0(z)\mathcal{K}_\theta(x|z)dz, \tag{4.3}$$

where $\mathcal{K}_\theta$ denotes the transition kernel of T steps of Langevin dynamics that samples p_θ. Generally, $\tilde{p}_\theta(x)$ is not necessarily equal to $p_\theta(x)$. $\tilde{p}_\theta(x)$ is dependent on T and δ, which are omitted in the notation for simplicity. The KL-divergence $\mathbb{D}_{\text{KL}}(\tilde{p}_\theta(x)||p_\theta(x)) = -\text{entropy}(\tilde{p}_\theta(x)) - \mathbb{E}_{\tilde{p}_\theta(x)}[f(x)] + \log Z$. The gradient ascent part of Langevin dynamics, $x_t = x_{t-1} + \delta^2/2\nabla_x f_\theta(x_{t-1})$, increases the negative energy $f_\theta(x)$ by moving x to the local modes of $f_\theta(x)$, and the diffusion part $\delta\epsilon_t$ increases the entropy of $\tilde{p}_\theta(x)$ by jumping out of the local modes. According to the law of thermodynamics (CT06), $\mathbb{D}_{\text{KL}}(\tilde{p}_\theta(x)||p_\theta(x))$ decreases to zero monotonically as $T \to \infty$.

4.2.2 Normalizing Flow

Let $z \in \mathbb{R}^D$ be the latent vector of the same dimensionality as x. A normalizing flow is of the form

$$x = g_\alpha(z); z \sim q_0(z), \tag{4.4}$$

where $q_0(z)$ is a known prior distribution such as Gaussian white noise distribution $\mathcal{N}(0, I_D)$, and $g_\alpha : \mathbb{R}^D \to \mathbb{R}^D$ is a mapping that consists of a sequence of L invertible transformations, i.e., $g(z) = g_L \circ \cdots \circ g_2 \circ g_1(z)$, whose inversion $z = g_\alpha^{-1}(x)$ and log-determinants of the Jacobians can be computed in closed form. α are the parameters of g_α. The mapping is used to transform a random vector z that follows a simple distribution q_0 into a flexible distribution. Under the change-of-variables law, the resulting random vector $x = g_\alpha(z)$ has a probability density $q_\alpha(x) = q_0(g_\alpha^{-1}(x))|\det(\partial g_\alpha^{-1}(x)/\partial x)|$.

Let $h_l = g_l(h_{l-1})$. The successive transformations between x and z can be expressed as a flow $z \overset{g_1}{\longleftrightarrow} h_1 \overset{g_2}{\longleftrightarrow} h_2 \cdots \overset{g_L}{\longleftrightarrow} x$, where we define $z := h_0$ and $x := h_L$ for succinctness. Then the determinant becomes $|\det(\partial g_\alpha^{-1}(x)/\partial x)| = \prod_{l=1}^{L} |\det(\partial h_{l-1}/\partial h_l)|$. The log-likelihood of a datapoint x can be easily computed by

$$\log q_\alpha(x) = \log q_0(z) + \sum_{l=1}^{L} \log \left| \det \left(\frac{\partial h_{l-1}}{\partial h_l} \right) \right|. \tag{4.5}$$

With some smart designs of the sequence of transformations $g_\alpha = \{g_l, l = 1, ..., L\}$, the log-determinant in Eq. (4.5) can be easily computed, then the normalizing flow $q_\alpha(x)$ can be trained by maximizing the exact data log-likelihood $L(\alpha) = \frac{1}{n} \sum_{i=1}^{n} \log q_\alpha(x_i)$ via gradient ascent algorithm.

4.3 CoopFlow: Cooperative Training of Two Flows

4.3.1 CoopFlow Algorithm

We have moved from trying to use a convergent Langevin dynamics to train a valid EBM. Instead, we accept the fact that the short-run non-convergent MCMC is inevitable and more affordable in practice, and we treat a non-convergent short-run Langevin flow as a generator and propose to jointly train it with a normalizing flow as a rapid initializer for more efficient generation. The resulting generator is called CoopFlow, which consists of both the Langevin flow and the normalizing flow.

Specifically, at each iteration, for $i = 1, ..., m$, we first generate $z_i \sim \mathcal{N}(0, I_D)$, and then transform z_i by a normalizing flow to obtain $\hat{x}_i = g_\alpha(z_i)$. Next, starting from each $\hat{x}_i$, we run a Langevin flow (i.e., a finite number of Langevin steps toward an EBM $p_\theta(x)$) to obtain $\tilde{x}_i$. $\tilde{x}_i$ are considered synthesized examples that are generated by the CoopFlow model. We then update α of the normalizing flow by treating $\tilde{x}_i$ as training data, and update θ of the Langevin flow according to the learning gradient of the EBM, which is computed with the synthesized examples $\{\tilde{x}_i\}$ and the observed examples $\{x_i\}$. Algorithm 3 presents a description of the proposed CoopFlow algorithm. The advantage of this training scheme is that we only need to minimally modify the existing codes for the MLE training of the EBM p_θ and the normalizing flow q_α. The probability density of the CoopFlow $\pi_{(\theta,\alpha)}(x)$ is well defined, which can be implicitly expressed by

$$\pi_{(\theta,\alpha)}(x) = (\mathcal{K}_\theta q_\alpha)(x) = \int q_\alpha(x')\mathcal{K}_\theta(x|x')dx'. \tag{4.6}$$

$\mathcal{K}_\theta$ is the transition kernel of the Langevin flow. If we increase the length T of the Langevin flow, $\pi_{(\theta,\alpha)}$ will converge to the EBM $p_\theta(x)$. The network $f_\theta(x)$ in the Langevin flow is scalar valued and is of free form, whereas the network $g_\alpha(x)$ in the normalizing flow has high-dimensional output and is of a severely constrained form. Thus the Langevin flow can potentially provide a tighter fit to $p_{\text{data}}(x)$ than the normalizing flow. The Langevin flow may also be potentially more data efficient as it tends to have a smaller network than the normalizing flow. On the flip side, sampling

from the Langevin flow requires multiple iterations, whereas the normalizing flow can synthesize examples via a direct mapping. It is thus desirable to train these two flows simultaneously, where the normalizing flow serves as an approximate sampler to amortize the iterative sampling of the Langevin flow. Meanwhile, the normalizing flow is updated by a temporal difference MCMC teaching provided by the Langevin flow, to further amortize the short-run Langevin flow.

Algorithm 3 CoopFlow Algorithm

Input: (1) Observed images $\{x_i\}_i^n$; (2) Number of Langevin steps T; (3) Langevin step size δ; (4) Learning rate η_θ for Langevin flow; (5) Learning rate η_α for normalizing flow; (6) batch size m.
Output: Parameters $\{\theta, \alpha\}$

1: Randomly initialize θ and α.
2: **repeat**
3: Sample observed examples $\{x_i\}_i^m \sim p_{\text{data}}(x)$.
4: Sample noise examples $\{z_i\}_{i=1}^m \sim q_0(z)$,
5: Starting from z_i, generate $\hat{x}_i = g_\alpha(z_i)$ via normalizing flow.
6: Starting from $\hat{x}_i$, run a T-step Langevin flow to obtain $\tilde{x}_i$ by following Eq. (4.2).
7: Given $\{\tilde{x}_i\}$, update α by maximizing $\frac{1}{m}\sum_{i=1}^m \log q_\alpha(\tilde{x}_i)$ with Eq. (4.5).
8: Given $\{x_i\}$ and $\{\tilde{x}_i\}$, update EBM parameter θ.
9: **until** converged

4.3.2 Understanding the Learned Two Flows

Convergence equations. In the traditional contrastive divergence (CD) (Hin02b) algorithm, MCMC chains are initialized with observed data so that the CD learning seeks to minimize $\mathbb{D}_{\text{KL}}(p_{\text{data}}(x)||p_\theta(x)) - \mathbb{D}_{\text{KL}}((\mathcal{K}_\theta p_{\text{data}})(x)||p_\theta(x))$, where $(\mathcal{K}_\theta p_{\text{data}})(x)$ denotes the marginal distribution obtained by running the Markov transition $\mathcal{K}_\theta$, which is specified by the Langevin flow, from the data distribution p_{data}. In the CoopFlow algorithm, the learning of the EBM (or the Langevin flow model) follows a modified contrastive divergence, where the initial distribution of the Langevin flow is modified to be a normalizing flow q_α. Thus, at iteration t, the update of θ follows the gradient of $\mathbb{D}_{\text{KL}}(p_{\text{data}}||p_\theta) - \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}||p_\theta)$ with respect to θ. Compared to the traditional CD loss, the modified one replaces p_{data} by q_α in the second KL divergence term. At iteration t, the update of the normalizing flow q_α follows the gradient of $\mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}||q_\alpha)$ with respect to α. Let (θ^*, α^*) be a

fixed point the learning algorithm converges to, then we have the following convergence equations

$$\theta^* = \arg\min_{\theta} \mathbb{D}_{\text{KL}}(p_{\text{data}}||p_\theta) - \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^*} q_{\alpha^*}||p_\theta), \tag{4.7}$$

$$\alpha^* = \arg\min_{\alpha} \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^*} q_{\alpha^*}||q_\alpha). \tag{4.8}$$

Ideal case analysis. In the idealized scenario where the normalizing flow q_α has infinite capacity and the Langevin sampling can mix and converge to the sampled EBM, Eq. (4.8) means that q_{α^*} attempts to be the stationary distribution of $\mathcal{K}_{\theta^*}$, which is p_{θ^*} because $\mathcal{K}_{\theta^*}$ is the Markov transition kernel of a convergent MCMC sampled from p_{θ^*}. That is, $\min_\alpha \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^*} q_{\alpha^*}||q_\alpha) = 0$, thus $q_{\alpha^*} = p_{\theta^*}$. With the second term becoming 0 and vanishing, Eq. (4.7) degrades to $\min_\theta \mathbb{D}_{\text{KL}}(p_{\text{data}}||p_\theta)$ and thus θ^* is the maximum likelihood estimate. On the other hand, the normalizing flow q_α chases the EBM p_θ toward p_{data}, thus α^* is also a maximum likelihood estimate.

Moment matching estimator. In the practical scenario where the Langevin sampling is not mixing, the CoopFlow model $\pi_t = \mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}$ is an interpolation between the learned $q_{\alpha^{(t)}}$ and $p_{\theta^{(t)}}$, and it converges to $\pi^* = \mathcal{K}_{\theta^*} q_{\alpha^*}$, which is an interpolation between q_{α^*} and p_{θ^*}. π^* is the short-run Langevin flow starting from q_{α^*} towards EBM $p_{\theta *}$. π^* is a legitimate generator because $\mathbb{E}_{p_{\text{data}}}[\nabla_\theta f_{\theta^*}(x)] = \mathbb{E}_{\pi^*}[\nabla_\theta f_{\theta^*}(x)]$ at convergence. That is, π^* leads to moment matching in the feature statistics $\nabla_\theta f_{\theta^*}(x)$. In other words, π^* satisfies the above estimating equation.

Understanding via information geometry. Consider a simple EBM with $f_\theta(x) = \langle \theta, h(x) \rangle$, where $h(x)$ is the feature statistics. Since $\nabla_\theta f_\theta(x) = h(x)$, the MLE of the EBM $p_{\theta_{\text{MLE}}}$ is a moment matching estimator due to $\mathbb{E}_{p_{\text{data}}}[h(x)] = \mathbb{E}_{p_{\theta_{\text{MLE}}}}[h(x)]$. The CoopFlow π^* also converges to a moment matching estimator, i.e., $\mathbb{E}_{p_{\text{data}}}[h(x)] = \mathbb{E}_{\pi^*}[h(x)]$. Figure 4.1 is an illustration of model distributions that correspond to different parameters at convergence. We first introduce three families of distributions: $\Omega = \{p : \mathbb{E}_p[h(x)] = \mathbb{E}_{p_{\text{data}}}[h(x)]\}$, $\Theta = \{p_\theta(x) = \exp(\langle \theta, h(x) \rangle)/Z(\theta), \forall \theta\}$, and $A = \{q_\alpha, \forall \alpha\}$, which are shown by the red, blue and green curves respectively in Figure 4.1. Ω is the set of distributions that reproduce statistical property $h(x)$ of the data distribution. Obviously, $p_{\theta_{\text{MLE}}}$, p_{data}, and $\pi^* = \mathcal{K}_{\theta^*} q_{\alpha^*}$ are included in Ω. Θ is the set of EBMs with different values of θ,

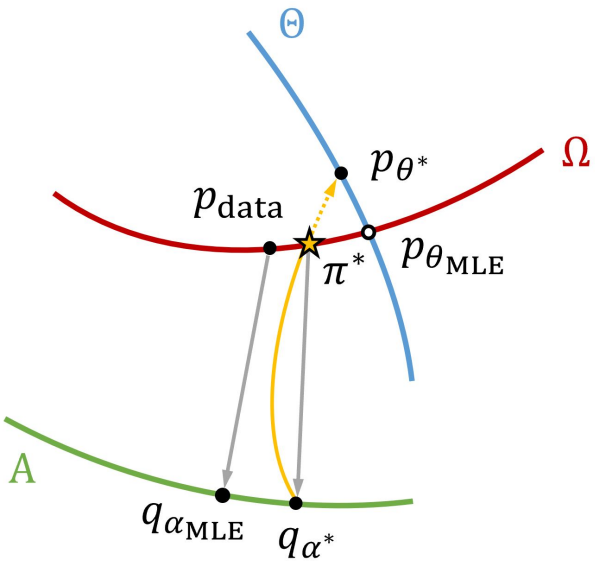


Figure 4.1: An illustration of convergence of the CoopFlow algorithm.

thus $p_{\theta_{\text{MLE}}}$ and p_{θ^*} belong to Θ. Because of the short-run Langevin flow, p_{θ^*} is not a valid model that matches the data distribution in terms of $h(x)$, and thus p_{θ^*} is not in Ω. A is the set of normalizing flow models with different values of α, thus q_{α^*} and $q_{\alpha_{\text{MLE}}}$ belong to A. The yellow line shows the MCMC trajectory. The solid segment of the yellow line, starting from q_{α^*} to $\mathcal{K}_{\theta^*}q_{\alpha^*}$, illustrates the short-run non-mixing MCMC that is initialized by the normalizing flow q_{α^*} in A and arrives at $\pi^* = \mathcal{K}_{\theta^*}q_{\alpha^*}$ in Ω. The dotted segment of the yellow line, starting from $\pi^* = \mathcal{K}_{\theta^*}q_{\alpha^*}$ in Ω to p_{θ^*} in Θ, shows the potential long-run MCMC trajectory, which is not really realized because we stop short in MCMC. If we increase the number of steps of short-run Langevin flow, $\mathbb{D}_{\text{KL}}(\pi^*||p_{\theta^*})$ will be monotonically decreasing to 0. Though π^* stops midway in the path toward $p_{\theta*}$, π^* is still a valid generator because it is in Ω.

Perturbation of MLE. $p_{\theta_{\text{MLE}}}$ is the intersection between Θ and Ω. It is the projection of p_{data} onto Θ because $\theta_{\text{MLE}} = \arg\min_\theta \mathbb{D}_{\text{KL}}(p_{\text{data}}||p_\theta)$. $q_{\alpha_{\text{MLE}}}$ is also a projection of p_{data} onto A because $\alpha_{\text{MLE}} = \arg\min_\alpha \mathbb{D}_{\text{KL}}(p_{\text{data}}||q_\alpha)$. Generally, $q_{\alpha_{\text{MLE}}}$ is far from p_{data} due to its restricted form network, whereas $p_{\theta_{\text{MLE}}}$ is very close to p_{data} due to its scalar-valued free form network. Because the short-run MCMC is not mixing, $\theta^* \neq \theta_{\text{MLE}}$ and $\alpha^* \neq \alpha_{\text{MLE}}$. θ^* and α^* are perturbations of

θ_{MLE} and α_{MLE}, respectively. $p_{\theta_{\text{MLE}}}$ shown by an empty dot is not attainable unfortunately. We want to point out that, as T goes to infinity, $p_{\theta^*} = p_{\theta_{\text{MLE}}}$, and $q_{\alpha^*} = q_{\alpha_{\text{MLE}}}$. Note that Ω, Θ and A are high-dimensional manifolds instead of 1D curves as depicted in Figure 4.1, and π^* may be farther away from p_{data} than $p_{\theta_{\text{MLE}}}$ is. During learning, $q_{\alpha^{(t+1)}}$ is the projection of $\mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}$ on A. At convergence, q_{α^*} is the projection of $\pi^* = \mathcal{K}_{\theta^*} q_{\alpha^*}$ on A. There is an infinite looping between q_{α^*} and $\pi^* = \mathcal{K}_{\theta^*} q_{\alpha^*}$ at convergence of the CoopFlow algorithm, i.e., π^* lifts q_{α^*} off the ground A, and the projection drops π^* back to q_{α^*}. Although p_{θ^*} and q_{α^*} are biased, they are not wrong. Many useful models and algorithms, e.g., variational autoencoder and contrastive divergence are also biased to MLE. Their learning objectives also follow perturbation of MLE.

4.3.3 Comparison Between CoopFlow and Short-Run EBM via Information Geometry

Figure 4.2 illustrates the convergences of both the CoopFlow and the EBM with a short-run MCMC starting from an initial noise distribution q_0. We define the short-run EBM in a more generic form (XLZ16a) as follows

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp[f_\theta(x)] q_0(x), \tag{4.9}$$

which is an exponential tilting of a known reference distribution $q_0(x)$. In general, the reference distribution can be either the Gaussian distribution or the uniform distribution. When the reference distribution is the uniform distribution, q_0 can be removed in Eq. (4.9). Since the initial distribution of the CoopFlow is the Gaussian distribution q_0, which is actually the prior distribution of the normalizing flow. For a convenient and fair comparison, we will use the Gaussian distribution as the reference distribution of the EBM in Eq. (4.9). And the CoopFlow and the baseline short-run EBM will use the same EBM defined in Eq. (4.9) in their frameworks. We will use $\bar{p}_{\bar{\theta}}$ to denote the baseline short-run EBM and keep using p_θ to denote the EBM component in the CoopFlow.

There are three families of distributions: $\Omega = \{p : \mathbb{E}_p[h(x)] = \mathbb{E}_{p_{\text{data}}}[h(x)]\}$, $\Theta = \{p_\theta(x) = \exp(\langle \theta, h(x) \rangle) q_0(x)/Z(\theta), \forall \theta\}$, and $A = \{q_\alpha, \forall \alpha\}$, which are shown by the red, blue and green

curves respectively in Figure 4.2, which is an extension of Figure 4.1 by adding the following elements:

- q_0, which is the initial distribution for both the CoopFlow and the short-run EBM. It belongs to Θ because it corresponds to $\theta = 0$. q_0 is just a noise distribution thus it is far under the green curve. That is, it is very far from $q_{\alpha *}$ because $q_{\alpha *}$ has been already a good approximation of p_{θ^*}.

- g_{α^*}, which is the learned transformation of the normalizing flow q_α, and is visualized as a mapping from q_0 to q_{α^*} by a directed brown line segment.

- The MCMC trajectory of the baseline short-run EBM $\bar{p}_{\bar{\theta}}$, which is shown by the yellow line on the right hand side of the blue curve. The solid part of the yellow line, starting from q_0 to $\bar{\pi}^* = \mathcal{K}_{\bar{\theta}^*} q_0$, shows the short-run non-mixing MCMC starting from the initial Gaussian distribution q_0 in Θ and arriving at $\bar{\pi}^*$ in Ω. The dotted part of the yellow line is the potential long-run MCMC trajectory that is unrealized.

By comparing the MCMC trajectories of the CoopFlow and the short-run EBM in Figure 4.2, we can find that the CoopFlow has a much shorter MCMC trajectory than that of the short-run EBM, since the normalizing flow g_{α^*} in the CoopFlow amortizes the sampling workload for the Langevin flow in the CoopFlow model.

4.3.4 Convergence Analysis

The CoopFlow algorithm simply involves two MLE learning algorithms: (i) the MLE learning of the EBM p_θ and (ii) the MLE learning of the normalizing flow q_α. The convergence of each of the two learning algorithms has been well studied and verified in the existing literature, e.g., (You99; XLZ16a; KD18). That is, each of them has a fixed point. The only interaction between these two MLE algorithms in the proposed CoopFlow algorithm is that, in each learning iteration, they feed each other with their synthesized examples and use the cooperatively synthesized examples

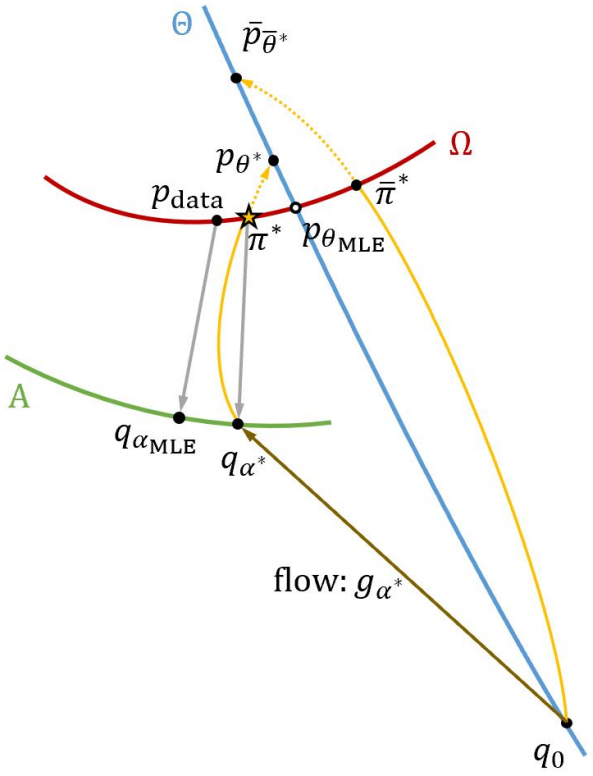


Figure 4.2: A comparison between the CoopFlow and the short-run EBM.

in their parameter update formulas. To be specific, the normalizing flow uses its synthesized examples to initialize the MCMC of the EBM, while the EBM feeds the normalizing flow with its synthesized examples as training examples. The synthesized examples from the Langevin flow are considered the cooperatively synthesized examples by the two models, and are used to compute their learning gradients. Unlike other amortized sampling methods (e.g., (HNF19b; GKH21b)) that uses variational learning, the EBM and normalizing flow in our framework do not back-propagate each other through the cooperatively synthesized examples. They just feed each other with some input data for their own training algorithms. That is, each learning algorithm will still converge to a fixed point.

Now let us analyze the convergence of the whole CoopFlow algorithm that alternates two maximum likelihood learning algorithms. We first analyze the convergence of the objective function at each learning step, and then we conclude the convergence of the whole algorithm.

The convergence of CD learning of EBM. The learning objective of the EBM is to minimize the KL divergence between the EBM p_θ and the data distribution p_{data}. Since the MCMC of the EBM in our model is initialized by the normalizing flow q_α, it follows a modified contrastive divergence algorithm. That is, at iteration t, it has the following objective,

$$\theta^{(t+1)} = \arg\min_\theta \mathbb{D}_{\text{KL}}(p_{\text{data}}||p_\theta) - \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}||p_\theta). \tag{4.10}$$

No matter what kind of distribution is used to initialize the MCMC, it will has a fixed point when the learning gradient of θ equals to 0, i.e., $L'(\theta) = \frac{1}{n}\sum_{i=1}^n \nabla_\theta f_\theta(x_i) - \frac{1}{n}\sum_{i=1}^n \nabla_\theta f_\theta(\tilde{x}_i) = 0$. The initialization of the MCMC only affects the location of the fixed point of the learning algorithm. The convergence and the analysis of the fixed point of the contrastive divergence algorithm has been studied by (Hin02b; CH05).

The convergence of MLE learning of normalizing flow. The objective of the normalizing flow is to learn to minimize the KL divergence between the normalizing flow and the Langevin flow (or the EBM) because, in each learning iteration, the normalizing flow uses the synthesized examples generated from the Langevin dynamics as training data. At iteration t, it has the following objective

$$\alpha^{(t+1)} = \arg\min_\alpha \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}||q_\alpha), \tag{4.11}$$

which is a convergent algorithm at each t. The convergence has been studied by (You99).

The convergence of CoopFlow. The CoopFlow alternates the above two learning algorithms. The EBM learning seeks to reduce the KL divergence between the EBM and the data, i.e., $p_\theta \to p_{data}$; while the MLE learning of normalizing flow seeks to reduce the KL divergence between the normalizing flow and the EBM, i.e., $q_\alpha \to p_\theta$. Therefore, the normalizing flow will chase the EBM toward the data distribution gradually. Because the process $p_\theta \to p_{\text{data}}$ will stop at a fixed point, therefore $q_\alpha \to p_\theta$ will also stop at a fixed point obviously. Such a chasing game is a contraction algorithm, therefore the fixed point of the CoopFlow exists. Empirical evidence also support our

claim. If we use (θ^*, α^*) to denote the fixed point of the CoopFlow, according to the definition of a fixed point, (θ^*, α^*) will satisfy

$$\theta^* = \arg\min_{\theta} \mathbb{D}_{\text{KL}}(p_{\text{data}}||p_\theta) - \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^*} q_{\alpha^*}||p_\theta), \tag{4.12}$$

$$\alpha^* = \arg\min_{\alpha} \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^*} q_{\alpha^*}||q_\alpha). \tag{4.13}$$

The convergence of the cooperative learning framework (CoopNets) that integrates the MLE algorithm of an EBM and the MLE algorithm of a generic generator has been verified and discussed in (XLG20). The CoopFlow that uses a normalizing flow instead of a generic generator has the same convergence property as that of the original CoopNets. The major contribution in our paper is to start from the above fixed point equation to analyze where the fixed point will be in our learning algorithm, especially when the MCMC is non-mixing and non-convergent. This goes beyond all the prior works about cooperative learning.

4.4 Experiments

We showcase experiment results on various tasks. We start from a toy example to illustrate the basic idea of the CoopFlow in Section 4.4.1. We show image generation results in Section 4.4.2. Section 4.4.3 demonstrates the learned CoopFlow is useful for image reconstruction and inpainting, while Section 4.4.4 shows that the learned latent space is meaningful so that it can be used for interpolation.

4.4.1 Toy Example Study

We first demonstrate our idea using a two-dimensional toy example where data lie on a spiral. We train three CoopFlow models with different lengths of Langevin flows. As shown in Figure 4.3, the rightmost box shows the results obtained with 10,000 Langevin steps. We can see that both the normalizing flow q_α and the EBM p_θ can fit the ground truth density, which is displayed in the red

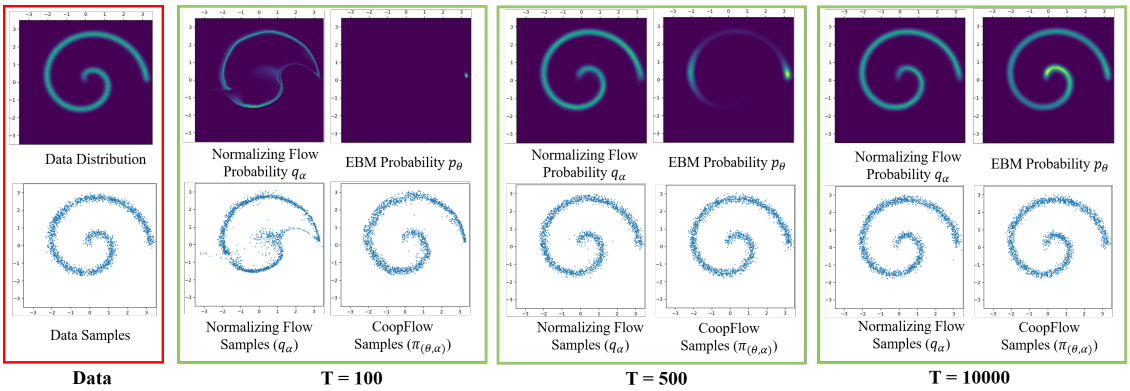


Figure 4.3: Learning CoopFlows on two-dimensional data. The ground truth data distribution is shown in the red box and the models trained with different Langevin steps are in the green boxes. In each green box, the first row shows the learned distributions of the normalizing flow and the EBM, and the second row shows the samples from the learned normalizing flow and the learned CoopFlow.

box, perfectly. This validates that, with a sufficiently long Langevin flow, the CoopFlow algorithm can learn both a valid q_α and a valid p_θ. The leftmost green box represents the model trained with 100 Langevin steps. This is a typical non-convergent short-run MCMC setting. In this case, neither q_α nor p_θ is valid, but their cooperation is. The short-run Langevin dynamics toward the EBM actually works as a flow-like generator that modifies the initial proposal by the normalizing flow. We can see that the samples from q_α are not perfect, but after the modification, the samples from $\pi_{(\theta,\alpha)}$ fit the ground truth distribution very well. The third box shows the results obtained with 500 Langevin steps. This is still a short-run setting, even though it uses more Langevin steps. p_θ becomes better than that with 100 steps, but it is still invalid. With an increased number of Langevin steps, samples from both q_α and $\pi_{(\theta,\alpha)}$ are improved and comparable to those in the long-run setting with 10,000 steps. The results verify that the CoopFlow might learn a biased BEM and a biased normalizing flow if the Langevin flow is non-convergent. However, the non-convergent Langevin flow together with the biased normalizing flow can still form a valid generator that synthesizes valid examples.

4.4.2 Image Generation

We test our model on three image datasets for image synthesis. (i) CIFAR-10 (KH09) is a dataset containing 50k training images and 10k testing images in 10 classes; (ii) SVHN (NWC11) is a

dataset containing over 70k training images and over 20k testing images of house numbers; (iii) CelebA (LLW15) is a celebrity facial dataset containing over 200k images. We downsample all the images to the resolution of 32×32.

Table 4.1: FID scores on the CIFAR-10, SVHN and CelebA datasets (32×32 pixels).

	Models	FID ↓
VAE	VAE (KW14b)	78.41
Autoregressive	PixelCNN (SKC17)	65.93
	PixelIQN (ODM18)	49.46
GAN	WGAN-GP(GAA17)	36.40
	SN-GAN (MKK18)	21.70
	StyleGAN2-ADA (KAH20)	2.92
Score-Based	NCSN (SE19)	25.32
	NCSN-v2 (SE20)	31.75
	NCSN++ (SSK21)	2.20
Flow	Glow (KD18)	45.99
	Residual Flow (CBD19)	46.37
EBM	LP-EBM (PHN20)	70.15
	EBM-SR (NHZ19)	44.50
	EBM-IG (DM19)	38.20
	CoopVAEBM (XZL21)	36.20
	CoopNets (XLG20)	33.61
	Divergence Triangle (HNZ20)	30.10
	VARA (GKH21b)	27.50
	EBM-CD (DLT21b)	25.10
	GEBM (AZG21)	19.31
	CF-EBM (ZXL21)	16.71
	VAEBM (XKK21)	12.16
	EBM-Diffusion (GSP21b)	9.58
Flow+EBM	NT-EBM (NGS22)	78.12
	EBM-FCE (GNK20)	37.30
Ours	CoopFlow(T=30)	21.16
	CoopFlow(T=200)	18.89
	CoopFlow(Pre)	15.80

(a) CIFAR-10

Models	FID ↓
ABP (HLZ17)	49.71
ABP-SRI (NPH20)	35.23
ABP-OT (AXL21)	19.48
VAE (KW14b)	46.78
SNGE (DAS20)	42.83
RAE (GSV20)	40.02
Glow (KD18)	41.70
DCGAN (RMC15)	21.40
NT-EBM (NGS22)	48.01
LP-EBM (PHN20)	29.44
EBM-FCE (GNK20)	20.19
CoopFlow(T=30)	18.11
CoopFlow(T=200)	16.97
CoopFlow(Pre)	15.32

(b) SVHN

Models	FID ↓
ABP (HLZ17)	51.50
ABP-SRI (NPH20)	36.84
VAE (KW14b)	38.76
Glow (KD18)	23.32
DCGAN (RMC15)	12.50
EBM-FCE (GNK20)	12.21
GEBM (AZG21)	5.21
CoopFlow(T=30)	6.44
CoopFlow(T=200)	4.90
CoopFlow(Pre)	4.15

(c) CelebA

For our model, we show results of three different settings. *CoopFlow(T=30)* denotes the setting where we train a normalizing flow and a Langevin flow together from scratch and use 30 Langevin steps. *CoopFlow(T=200)* denotes the setting where we increase the number of Langevin steps to 200. In the *CoopFlow(Pre)* setting, we first pretrain a normalizing flow from observed data, and then train the CoopFlow with the parameters of the normalizing flow being initialized by the pretrained

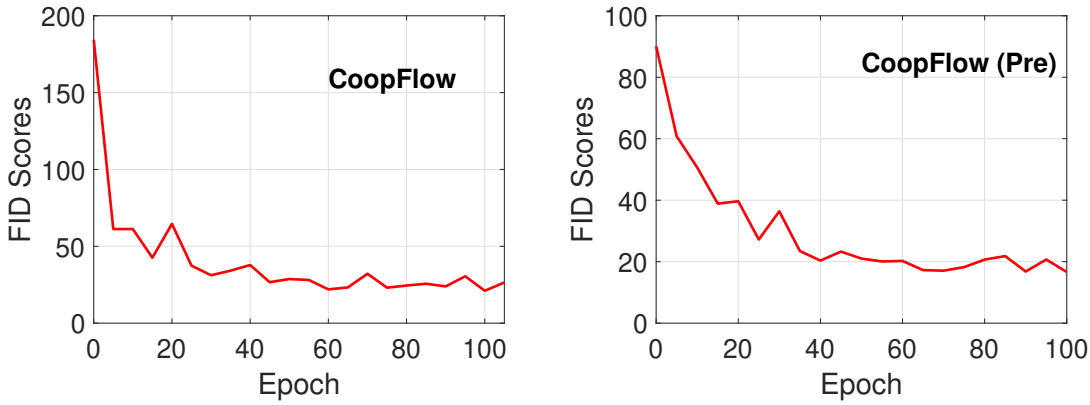
one. We use a 30-step Langevin flow in this setting. For all the three settings, we slightly increase the Langevin step size at the testing stage for better performance. We show both qualitative results in Figure 4.4 and quantitative results in Table 4.1. To calculate FID (HRU17a) scores, we generate 50,000 samples on each dataset. Our models outperform most of the baseline algorithms. We get lower FID scores comparing to the individual normalizing flows and the prior works that jointly train a normalizing flow with an EBM, e.g., (GNK20) and (NGS22). We also achieve comparable results with the state-of-the-art EBMs. We can see using more Langevin steps or a pretrained normalizing flow can help improve the performance of the CoopFlow. The former enhances the expressive power, while the latter stabilizes the training.

(a) CIFAR-10 (b) SVHN (c) CelebA

Figure 4.4: Generated examples (32×32 pixels) by the CoopFlow models trained on the CIFAR-10, SVHN and CelebA datasets respectively. Samples are obtained from the setting of *CoopFlow(pre)*.

Figure 4.5 shows the FID trends during the training of the CoopFlow models on the CIFAR-10 dataset. Each curve represents the FID scores over training epochs. We train the CoopFlow models using the settings of *CoopFlow(T=30)* and *CoopFlow(pre)*. For each of them, we can observe that, as the cooperative learning proceeds, the FID keeps decreasing and converges to a low value. This is an empirical evidence to show that the proposed CoopFlow algorithm is a convergent algorithm.

4.4.3 Image Reconstruction and Inpainting



(a) FID curve for *CoopFlow(T=30)*. (b) FID curve for *CoopFlow(Pre)*.

Figure 4.5: FID curves on the CIFAR-10 dataset. The FID score is reported every 5 epochs.

We show that the learned CoopFlow model is able to reconstruct observed images. We may consider the CoopFlow model $\pi_{(\theta,\alpha)}(x)$ a latent variable generative model: $z \sim q_0(z); \hat{x} = g_\alpha(z); x = F_\theta(\hat{x}, e)$, where z denotes the latent variables, e denotes all the injected noises in the Langevin flow, and F_θ denotes the mapping realized by a T-step Langevin flow that is actually a T-layer noise-injected residual network. Since the Langevin flow is not mixing, x is dependent on $\hat{x}$ in the Langevin flow, thus also dependent on z. The CoopFlow is a generator $x = F_\theta(g_\alpha(z), e)$, so we can reconstruct any x by inferring the corresponding latent variables z using gradient descent on $L(z) = ||x - F_\theta(g_\alpha(z), e)||^2$, with z being initialized by q_0. However, g is an invertible transformation, so we can infer z by an efficient way, i.e., we first find $\hat{x}$ by gradient descent on $L(\hat{x}) = ||x - F_\theta(\hat{x}, e)||^2$, with $\hat{x}$ being initialized by $\hat{x}_0 = g_\alpha(z)$ where $z \sim q_0(z)$ and e being set to be 0, and then use $z = g_\alpha^{-1}(\hat{x})$ to get the latent variables. These two methods are equivalent, but the latter one is computationally efficient, since computing the gradient on the whole two-flow generator $F_\theta(g_\alpha(z), e)$ is difficult and time-consuming. Let $\hat{x}^* = \arg\min_{\hat{x}} L(\hat{x})$. The reconstruction is given by $F_\theta(\hat{x}^*)$. The optimization is done using 200 steps of gradient descent over $\hat{x}$. The reconstruction results are shown in Figure 4.6. We use the CIFAR-10 testing set. The right column displays the original images x that need to be reconstructed. The left and the middle columns display $\hat{x}^*$ and $F_\theta(\hat{x}^*)$, respectively.

We also provide quantitative results for the image reconstruction experiment. Following (NHZ19), we calculate the per-pixel mean squared error (MSE) on 1,000 examples in the testing set of the CIFAR-10 dataset. We use a 200-step gradient descent to minimize the reconstruction loss. We plot the reconstruction error curve showing the MSEs over iterations in Figure 4.7 and report the final per-pixel MSE in Table 4.2. For a baseline, we train an EBM using a 100-step short-run MCMC and the resulting model is the short-run Langevin flow. We then apply it to the reconstruction task of the same 1,000 images by following (NHZ19). The baseline EBM has the same network architecture as that of the EBM component in our CoopFlow model for fair comparison. The experiment results show that the CoopFlow works better than the individual short-run Langevin flow in this image reconstruction task.

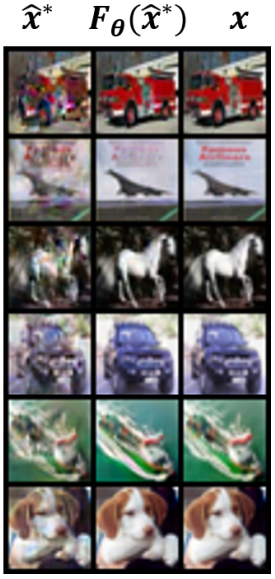


Figure 4.6: Image reconstruction on the CIFAR-10.

Model	MSE
Langevin Flow / EBM with short-run MCMC	0.1083
CoopFlow (ours)	0.0254

Table 4.2: Reconstruction error (MSE per pixel).

The aforementioned results demonstrate that our model can successfully reconstruct the observed images, verifying that the CoopFlow with a non-mixing MCMC is indeed a valid latent variable model.

We further show that our model is also capable of doing image inpainting. Similar to image reconstruction, given a masked observation x_{mask} along with a binary matrix M indicating the positions of the unmasked pixels, we optimize $\hat{x}$ to minimize the reconstruction error between $F_\theta(\hat{x})$ and x_{mask} in the unmasked area, i.e., $L(\hat{x}) = ||M \odot (x_{\text{mask}} - F_\theta(\hat{x}))||^2$, where $\odot$ is the element-wise multiplication operator. $\hat{x}$ is still initialized by the normalizing flow. We do experiments on the

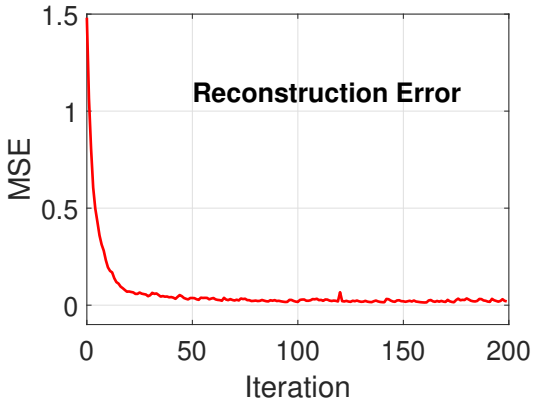


Figure 4.7: Reconstruction errors (MSE per pixel) over iterations.

CelebA training set. In Figure 4.8, the two subfigures correspond to two target images to be inpainted. Each row in the same subfigure shows one example of inpainting to the same target image with a different initialization of $\hat{x}$ provided by the normalizing flow. The first 17 columns display the inpainting results at different optimization iterations. The last two columns show the masked images and the original images respectively. We can see that our model can reconstruct the unmasked areas faithfully and simultaneously fill in the blank areas of the input images. With different initializations, our model can inpaint diversified and meaningful patterns.

4.4.4 Interpolation in the Latent Space

The CoopFlow model is capable of doing interpolation in the latent space z. Given an image x, we first find its corresponding $\hat{x}^*$ using the reconstruction method described in Section 4.4.3. We then infer z by the inversion of the normalizing flow $z^* = g_\alpha^{-1}(\hat{x}^*)$.

Figure 4.9 shows two examples of interpolation between two latent vectors inferred from observed images. For each row, the images at the two ends are observed. Each image in the middle is obtained by first interpolating the latent vectors of the two end images, and then generating the image using the CoopFlow generator. This experiment shows that the CoopFlow can learn a smooth latent space that traces the data manifold.

4.5 Implementation Details

4.5.1 Network Architecture of the CoopFlow

We use the same network architecture for all the experiments. For the normalizing flow $g_\alpha(z)$ in our CoopFlow framework, we use the Flow++ (HCS19) network architecture that was originally designed for the CIFAR-10 (32×32 pixels) dataset in (HCS19). As to the EBM in our CoopFlow, we use the architecture shown in Table 4.3 to design the negative energy function $f_\theta(x)$.

3×3 Conv2d, *str*=1, *pad*=1, *ch*=128; Swish
Residual Block, *ch*=256
Residual Block, *ch*=512
Residual Block, *ch*=1,024
3×3 Conv2d, *str*=4, *pad*=0, *ch*=100; Swish
Sum over channel dimension

(a) EBM architecture

3×3 Conv2d, *str*=1, *pad*=1; Swish
3×3 Conv2d, *str*=1, *pad*=1; Swish
+ 3×3 Conv2d, *str*=1, *pad*=1; Swish (input)
2×2 Average pooling

(b) Residual Block architecture

Table 4.3: Network architecture of the EBM in the CoopFlow (*str*: stride, *pad*: padding, *ch*: channel).

4.5.2 Experimental Details

We have three different settings for the CoopFlow model. In the *CoopFlow(T=30)* setting and the *CoopFlow(T=200)* setting, we train both the normalizing flow and the Langevin flow from scratch. The difference between them are only the number of the Langevin steps. The *CoopFlow(T=200)* uses a longer Langevin flow than the *CoopFlow(T=30)*. We follow (HCS19) and use the data-dependent parameter initialization method (SK16) for our normalizing flow in both settings *CoopFlow(T=30)* and *CoopFlow(T=200)*. On the other hand, as to the *CoopFlow(Pre)* setting, we first pretrain a normalizing flow on training examples, and then train a 30-step Langevin flow, whose parameters are initialized randomly, together with the pretrained normalizing flow by following Algorithm 3. The cooperation between the pretrained normalizing flow and the untrained Langevin flow would be

difficult and unstable because the Langevin flow is not knowledgeable at all to teach the normalizing flow. To stabilize the cooperative training and make a smooth transition for the normalizing flow, we include a warm-up phase in the CoopFlow algorithm. During this phase, instead of updating both the normalizing flow and the Langevin flow, we fix the parameters of the pretrained normalizing flow and only update the parameters of the Langevin flow. After a certain number of learning epochs, the Langevin flow may get used to the normalizing flow initialization, and learn to cooperate with it. Then we begin to update both two flows as described in Algorithm 3. This strategy is effective in preventing the Langevin flow from generating bad synthesized examples at the beginning of the CoopFlow algorithm to ruin the pretrained normalizing flow.

We use the Adam optimizer (KB15b) for training. We set learning rates $\eta_\alpha = 0.0001$ and $\eta_\theta = 0.0001$ for the normalizing flow and the Langevin flow, respectively. We use $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the normalizing flow and $\beta_1 = 0.5$ and $\beta_2 = 0.5$ for the Langevin flow. In the Adam optimizer, β_1 is the exponential decay rate for the first moment estimates, and β_2 is the exponential decay rate for the second-moment estimates. We adopt random horizontal flip as data augmentation only for the CIFAR-10 dataset. We remove the noise term in each Langevin update by following (ZXL21). We also propose an alternative strategy to gradually decay the effect of the noise term in Section 4.9. The batch sizes for settings *CoopFlow(T=30)*, *CoopFlow(T=200)*, and *CoopFlow(Pre)* are 28, 32 and 28. The values of other hyperparameters can be found in Table 4.4.

4.5.3 Analysis of Hyperparameters of Langevin Flow

We investigate the influence of the Langevin step size δ and the number of Langevin steps T on the CIFAR-10 dataset using the *CoopFlow(Pre)* setting. We first fix the Langevin step size to be 0.03 and vary the number of Langevin steps from 10 to 50. The results are shown in Table 4.5. On the other hand, we show the influence of the Langevin step size in Table 4.6, where we fix the number of Langevin steps to be 30 and vary the Langevin step size used in training. When synthesizing examples from the learned models in testing, we slightly increase the Langevin step size by a ratio of $4/3$ for better performance. We can see that our choices of 30 as the number of Langevin steps

Model	Dataset	# of epochs to pretrain Normal. Flow	# of warm-up epochs for Lang. Flow	# of epochs for CoopFlow	# of Langevin steps	Langevin step size (train)	Langevin step size (test)
CoopFlow (T=30)	CIFAR-10	0	0	100	30	0.03	0.04
	SVHN	0	0	100	30	0.03	0.035
	CelebA	0	0	100	30	0.03	0.035
CoopFlow (T=200)	CIFAR-10	0	0	100	200	0.01	0.012
	SVHN	0	0	100	200	0.011	0.0125
	CelebA	0	0	100	200	0.011	0.013
CoopFlow (Pre)	CIFAR-10	300	25	75	30	0.03	0.04
	SVHN	200	10	90	30	0.03	0.035
	CelebA	80	10	90	30	0.03	0.035

Table 4.4: Hyperparameter setting of the CoopFlow in our experiments.

and 0.03 as the Langevin step size are reasonable. Increasing the number of Langevin steps can improve the performance in terms of FID, but also be computationally expensive. The choice of $T = 30$ is a trade-off between the synthesis performance and the computation efficiency.

# of Langevin steps	10	20	30	40	50
FID ↓	16.46	15.20	15.80	16.80	15.64

Table 4.5: FID scores over the numbers of Langevin steps of *CoopFlow(Pre)* on the CIFAR-10 dataset.

Langevin step size (train)	0.01	0.02	0.03	0.04	0.05	0.10
Langevin step size (test)	0.013	0.026	0.04	0.053	0.067	0.13
FID ↓	15.99	16.32	15.80	16.52	18.17	19.82

Table 4.6: FID scores of *CoopFlow(Pre)* on the CIFAR-10 dataset under different Langevin step sizes.

4.6 Ablation Study

To show the effect of the cooperative training, we compare a CoopFlow model with an individual normalizing flow and an individual Langevin flow. For fair comparison, the normalizing flow component in the CoopFlow has the same network architecture as that in the individual normalizing flow, while the Langevin flow component in the CoopFlow also uses the same network architecture as that in the individual Langevin flow. We train the individual normalizing flow by following (HCS19) and train the individual Langevin flow by following (NHZ19). All three models are trained on the CIFAR-10 dataset. We present a comparison of these three models in terms of FID in Table 4.7, and also show generated samples in Figure 4.10. From Table 4.7, we can see that the CoopFlow model outperforms both the normalizing flow and the Langevin flow by a large margin, which verifies the effectiveness of the proposed CoopFlow algorithm.

Model	Normalizing flow	Langevin flow	CoopFlow
FID $\downarrow$	92.10	49.51	21.16

Table 4.7: A FID comparison among the normalizing flow, the Langevin flow and the CoopFlow.

4.7 Model Complexity

In Table 4.8, we present a comparison of different models in terms of model size and FID score. Here we mainly compare those models that have a normalizing flow component, e.g., EBM-FCE, NT-EBM, GLOW, Flow++, as well as an EBM jointly trained with a VAE generator, e.g., VAEBM. We can see the CoopFlow model has a good balance between model complexity and performance. It is noteworthy that both the CoopFlow and the EBM-FCE consist of an EBM and a normalizing flow, and their model sizes are also similar, but the CoopFlow achieves a much lower FID than the EBM-FCE. Note that the Flow++ baseline uses the same structure as that in our CoopFlow. By comparing the Flow++ and the CoopFlow, we can find that recruiting an extra Langevin flow can help improve the performance of the normalizing flow in terms of FID. On the other hand, although

the VAEBM model achieves a better FID than ours, but it relies on a much larger pretrained NVAE model (VK20a) that significantly increases its model complexity.

Model	# of Parameters	FID ↓
NT-EBM (NGS22)	23.8M	78.12
GLOW (KD18)	44.2M	45.99
EBM-FCE (GNK20)	44.9M	37.30
Flow++ (HCS19)	28.8M	92.10
VAEBM (XKK21)	135.1M	12.16
CoopFlow(T=30) (ours)	45.9M	21.16
CoopFlow(T=200) (ours)	45.9M	18.89
CoopFlow(pre) (ours)	45.9M	15.80

Table 4.8: A comparison of model sizes and FID scores among different models. FID scores are reported on the CIFAR-10 dataset.

4.8 Comparison with Models Using Short-Run MCMC

Our method is relevant to the short-run MCMC. In this section, we compare the CoopFlow model with other models that use a short-run MCMC as a flow-like generator. The baselines include (1) the single EBM with short-run MCMC starting from the noise distribution (NHZ19), and (ii) cooperative training of an EBM and a generic generator (XLG20). In Table 4.9, we report the FID scores of different methods over different numbers of MCMC steps. With the same number of Langevin steps, the CoopFlow can generate much more realistic image patterns than the other two baselines. Further, the results show that the CoopFlow can use less Langevin steps (i.e., a shorter Langevin flow) to achieve better performance.

4.9 Noise Term in the Langevin Dynamics

While for the experiments shown in the main text, we completely disable the noise term $\delta\epsilon$ of the Langevin equation presented in Eq. (4.2) by following (ZXL21) to achieve better results, here we

Model \ # of MCMC steps	10	20	30	40	50	200
Short-run EBM	421.3	194.88	117.02	140.79	198.09	54.23
CoopNets	33.74	33.48	34.12	33.85	42.99	38.88
CoopFlow(Pre) (ours)	16.46	15.20	15.80	16.80	15.64	17.94

Table 4.9: A comparison of FID scores of the short-run EBM, the CoopNets and the CoopFlow under different numbers of Langevin steps on the CIFAR-10 dataset.

try an alternative way where we gradually decay the effect of the noise term toward zero during the training process. The decay ratio for the noise term can be computed by the following:

$$\text{decay ratio} = \max((1.0 - \frac{\text{epoch}}{K})^{20}, 0.0) \tag{4.14}$$

where K is a hyper-parameter controlling the decay speed of the noise term. Such a noise decay strategy enables the model to do more exploration in the sampling space at the beginning of the training and then gradually focus on the basins of the reachable local modes for better synthesis quality when the model is about to converge. Note that we only decay the noise term during the training stage and still remove the noise term during the testing stage, including image generation and FID calculation. We carry out experiments on the CIFAR-10 and SVHN datasets using the *CoopFlow(Pre)* setting. The results are shown in Table 4.10.

Dataset	K	FID (no noise)	FID (decreasing noise)
CIFAR-10	30	15.80	14.55
SVHN	15	15.32	15.74

Table 4.10: FID scores for the CoopFlow models trained with gradually reducing noise term in the Langevin dynamics.

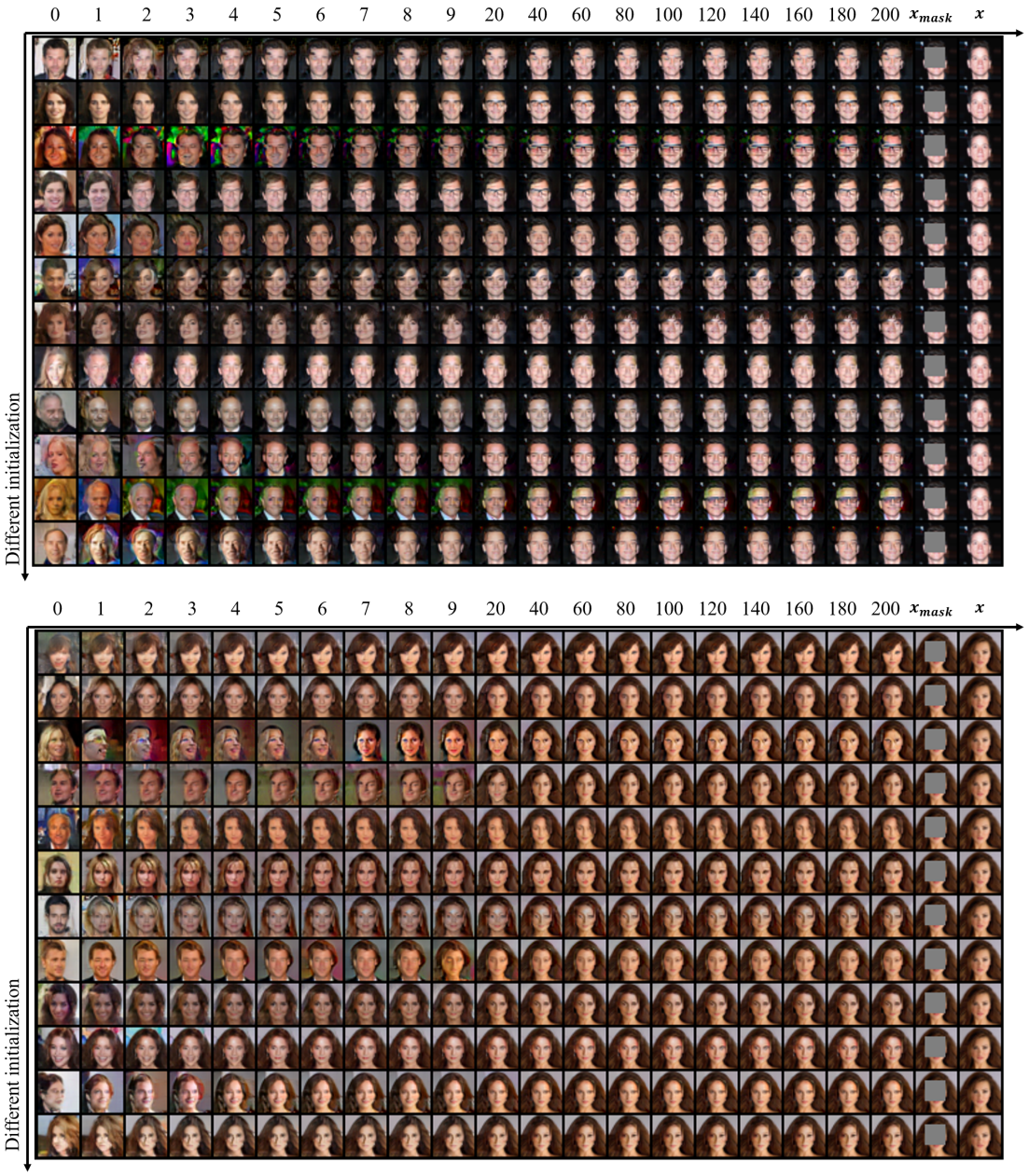


Figure 4.8: More results on image inpainting.

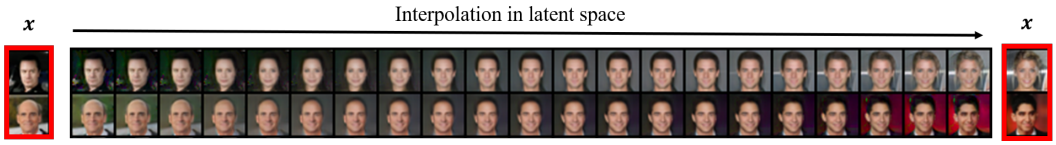
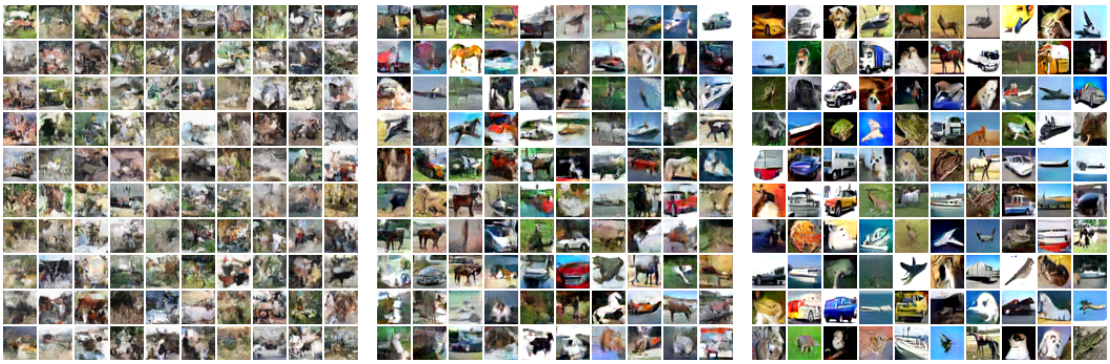


Figure 4.9: Image interpolation results on the CelebA dataset (32×32 pixels). The leftmost and rightmost columns display the images we observed. The columns in the middle represent the interploation results between the inferred latent vectors of the two end observed images.



(a) Normalizing flow (b) Langevin flow (c) CoopFlow(T=30)

Figure 4.10: Generated examples by (a) the individual normalizing flow, (b) the individual Langevin flow, and (c) the CoopFlow, which are trained on the CIFAR-10 dataset.

CHAPTER 5

Go One Step Further from CoopFlow: Learning Energy-Based Models by Cooperative Diffusion Recovery Likelihood

5.1 Motivation and Introduction

The success of CoopFlow demonstrates the effectiveness of cooperatively learning an initializer and an EBM. However, there is still a performance gap between CoopFlow with other STOA generative frameworks. And the energy function learned by CoopFlow can be incorrect. Therefore, in this chapter, we discuss how we can further resolve these problems.

Recently, Diffusion Recovery Likelihood (DRL) (GSP21a) has emerged as a powerful framework for estimating EBMs. Drawing inspiration from diffusion models (SWM15; HJA20; SE19), DRL employs a sequence of EBMs to model the marginal distributions of samples perturbed by a Gaussian diffusion process. Each EBM is trained using recovery likelihood, which maximizes the conditional probability of the data at the current noise level given their noisier versions at a higher noise level. This method is more tractable than maximizing regular likelihood because sampling from the conditional distribution is simpler than from the marginal distribution. Despite DRL's strong performance among EBM-based generative models, it still lags in sample quality when compared to other generative frameworks like GANs or diffusion models. Moreover, DRL requires about 30 MCMC sampling steps at each noise level to generate valid samples, which can be time-consuming during both training and sampling.

To further narrow the performance gap and speed up EBM training and sampling with fewer MCMC steps, we introduce Cooperative Diffusion Recovery Likelihood (CDRL). This method

jointly estimates a sequence of EBMs and MCMC initializers on data perturbed by a diffusion process. At each noise level, both the initializer and EBM are updated through a cooperative training scheme (XLG18). The initializer model proposes initial samples by predicting the samples at the current noise level based on their noisier versions from a higher noise level. These initial samples are then refined using a few MCMC sampling steps from the conditional distribution defined by the EBM. The EBM is updated by maximizing recovery likelihood with the refined samples, and the initializer is updated to account for the differences between the initial and refined samples. The initializer models learn to accumulate the MCMC transitions of the EBMs and reproduce them through direct ancestral sampling. By incorporating a new noise schedule and a variance reduction technique, we achieve significantly better performance than existing EBM estimation methods. Additionally, we integrate classifier-free guidance (CFG) (HS22) to enhance conditional generation performance, observing similar trade-offs between sample quality and diversity as seen in CFG for diffusion models when adjusting the guidance strength. Furthermore, our approach is applicable to several useful downstream tasks, including compositional generation, image inpainting, and out-of-distribution detection.

In this chapter, our main achievements are as follows:

- We propose Cooperative Diffusion Recovery Likelihood (CDRL), which efficiently learns and samples from a sequence of EBMs and initializers.
- We make several practical design choices related to noise scheduling, MCMC sampling, and noise variance reduction for EBM training.
- Empirically, we demonstrate that CDRL significantly improves sample quality compared to existing EBM approaches on CIFAR-10 and ImageNet 32×32 datasets.
- We show that CDRL has great potential to enable more efficient sampling with sampling adjustment techniques.
- We demonstrate CDRL's capabilities in compositional generation, image inpainting, and

out-of-distribution (OOD) detection, as well as its compatibility with classifier-free guidance for conditional generation.

5.2 Related Works

5.2.1 Denoising Diffusion Models

Diffusion models, initially introduced by (SWM15) and further developed in subsequent works such as (SE20; HJA20), generate samples by progressively denoising them from a high noise level to clean data. These models have demonstrated significant success in generating high-quality samples from complex distributions, owing to a range of architectural and algorithmic innovations (HJA20; SME21; KSS21; SSK21; DN21; KAA22; HS22). Notably, (DN21) emphasizes that the generative performance of diffusion models can be enhanced with the aid of a classifier, while (HS22) further demonstrates that this guided scoring can be estimated by the differential scores of a conditional model versus an unconditional model. Enhancements in sampling speed have been realized through distillation techniques (SH22) and the development of fast SDE/ODE samplers (SME21; KAA22; LZB22). Recent advancements (RBL22; SCS22; RDN22) have successfully applied conditional diffusion models to the task of text-to-image generation, achieving significant breakthroughs.

EBM shares a close relationship with diffusion models, as both frameworks can provide a score to guide the generation process, whether through Langevin dynamics or SDE/ODE solvers. As (SH21) discusses, the distinction between these two models lies in their implementation approaches: EBMs model the log-likelihood directly, while diffusion models focus on the gradient of the log-likelihood. This distinction brings advantages to EBMs, such as their compatibility with advanced sampling techniques (DDS23), potential conversion into classifiers (GMJ23), and capability to detect abnormal samples through estimated likelihood (GWJ20; LWO20).

The primary focus of this work is to advance the development of EBMs. Our approach

connects with diffusion models (HJA20; XKV22) by training a sequence of EBMs and MCMC initializers to reverse the diffusion process. In contrast to (HJA20), our framework employs more expressive conditional EBMs instead of normal distributions to represent the denoising distribution. Additionally, (XKV22) also suggests multimodal distributions, trained by generative adversarial networks (GPM20), for the reverse process.

5.3 Cooperative Diffusion Recovery Likelihood

5.3.1 Diffusion recovery likelihood

Given the difficulty of sampling from the marginal distribution $p(\mathbf{x})$ defined by an EBM, we could instead estimate a sequence of EBMs defined on increasingly noisy versions of the data and jointly estimate them by maximizing *recovery likelihood*. Specifically, assume a sequence of noisy training examples perturbed by a Gaussian diffusion process: $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_T$ such that $\mathbf{x}_0 \sim p_{\text{data}}$; $\mathbf{x}_{t+1} = \alpha_{t+1}\mathbf{x}_t + \sigma_{t+1}\boldsymbol{\epsilon}$. Denote $\mathbf{y}_t = \alpha_{t+1}\mathbf{x}_t$ for notation simplicity. The marginal distributions of $\{\mathbf{y}_t; t = 1, ..., T\}$ are modeled by a sequence of EBMs: $p_\theta(\mathbf{y}_t) = \frac{1}{Z_{\theta,t}} \exp(f_\theta(\mathbf{y}_t; t))$. Then the conditional EBM of $\mathbf{y}_t$ given the sample $\mathbf{x}_{t+1}$ at a higher noise level can be derived as

$$p_\theta(\mathbf{y}_t|\mathbf{x}_{t+1}) = \frac{1}{\tilde{Z}_{\theta,t}(\mathbf{x}_{t+1})} \exp\left(f_\theta(\mathbf{y}_t; t) - \frac{1}{2\sigma_{t+1}^2}\|\mathbf{y}_t - \mathbf{x}_{t+1}\|^2\right), \tag{5.1}$$

where $\tilde{Z}_{\theta,t}(\mathbf{x}_{t+1})$ is the partition function of the conditional EBM dependent on $\mathbf{x}_{t+1}$. Compared with the marginal EBM $p_\theta(\mathbf{y}_t)$, when σ_{t+1} is small, the extra quadratic term in $p_\theta(\mathbf{y}_t|\mathbf{x}_{t+1})$ constrains the conditional energy landscape to be localized around $\mathbf{x}_{t+1}$, making the latter less multi-modal and easier to sample from with MCMC. In the extreme case when σ_{t+1} is infinitesimal, $p_\theta(\mathbf{y}_t|\mathbf{x}_{t+1})$ is approximately a Gaussian distribution that can be tractably sampled from and has a close connection to diffusion models (GSP21a). In the other extreme case when $\sigma_{t+1} \to \infty$, the conditional distribution falls back to the marginal distribution, and we lose the advantage of being more MCMC friendly for the conditional distribution. Therefore, we need to maintain a small σ_{t+1} between

adjacent time steps, and to equip the model with the ability of generating new samples from white noises, we end up with estimating a sequence of EBMs defined on the diffusion process. We use the variance-preserving noise schedule (SSK21), under which case we have $\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \bar{\sigma}_t \boldsymbol{\epsilon}$, where $\bar{\alpha}_t = \prod_{t=1}^{T} \alpha_t$ and $\bar{\sigma}_t = \sqrt{1 - \bar{\alpha}_t^2}$.

We estimate each EBM by maximizing the following recovery log-likelihood function at each noise level (BYA13):

$$\mathcal{J}_t(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(\mathbf{y}_{t,i} | \mathbf{x}_{t+1,i}), \tag{5.2}$$

where $\{\mathbf{y}_{t,i}, \mathbf{x}_{t+1,i}\}$ are pair of samples at time steps t and $t+1$. Sampling from $p_\theta(\mathbf{y}_t | \mathbf{x}_{t+1})$ can be achieved by running K steps of Langevin dynamics from the initialization point $\tilde{\mathbf{y}}_t^0 = \mathbf{x}_{t+1,i}$ and iterating

$$\tilde{\mathbf{y}}_t^{\tau+1} = \tilde{\mathbf{y}}_t^{\tau} + \frac{s_t^2}{2} \left(\nabla_{\mathbf{y}} f_\theta(\tilde{\mathbf{y}}_t^{\tau}; t) - \frac{1}{\sigma_{t+1}^2} (\tilde{\mathbf{y}}_t^{\tau} - \mathbf{x}_{t+1}) \right) + s_t \boldsymbol{\epsilon}^{\tau}, \tag{5.3}$$

where s_t is the step size and τ is the index of MCMC sampling step. With the samples, the updating of EBMs then follows the same learning gradients as MLE, as the extra quadratic term $-\frac{1}{2\sigma_{t+1}^2} \|\mathbf{y}_t - \mathbf{x}_{t+1}\|^2$ in $p_\theta(\mathbf{y}_t | \mathbf{x}_{t+1})$ does not involve learnable parameters. It is worth noting that maximizing recovery likelihood still guarantees an unbiased estimator of the true parameters of the *marginal distribution* of the data.

5.3.2 Amortizing MCMC sampling with initializer models

Although $p_\theta(\mathbf{y}_t | \mathbf{x}_{t+1})$ is easier to sample from than $p_\theta(\mathbf{y}_t)$, when σ_{t+1} is not infinitesimal, the initialization of MCMC sampling, $\mathbf{x}_{t+1}$, may still be far from the data manifold of $\mathbf{y}_t$. This necessitates a certain amount of MCMC sampling steps at each noise level (e.g., 30 steps of Langevin dynamics in (GSP21a)). Naively reducing the number of sampling steps would lead to training divergence or performance degradation.

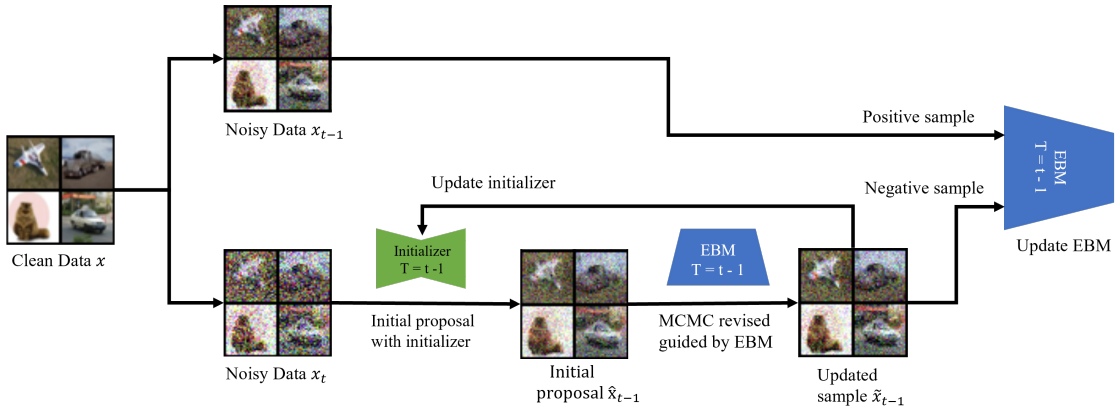
To address this issue, we propose to learn an initializer model jointly with the EBM at each noise level, which maps $\mathbf{x}_{t+1}$ closer to the manifold of $\mathbf{y}_t$. Our work is inspired by the CoopNets work (XLG18; XZL21; XZL22), which shows that jointly training a top-down generator via MCMC teaching will help the training of a single EBM model. We take this idea and generalize it to the recovery-likelihood model. Specifically, the initializer model at noise level t is defined as

$$q_\phi(\mathbf{y}_t|\mathbf{x}_{t+1}) \sim \mathcal{N}(\boldsymbol{g}_\phi(\mathbf{x}_{t+1}; t), \tilde{\sigma}_t^2 \boldsymbol{I}). \tag{5.4}$$

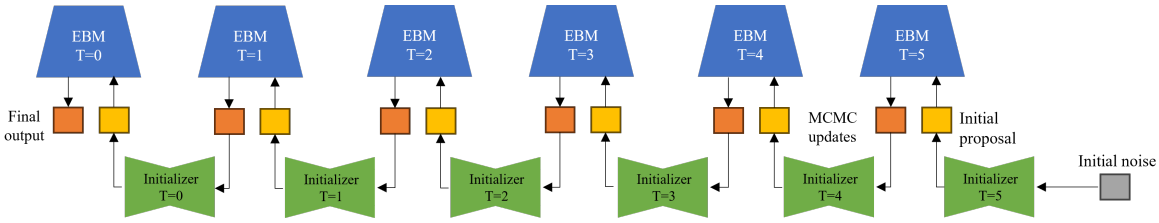
It serves as a coarse approximation to $p_\theta(\mathbf{y}_t|\mathbf{x}_{t+1})$, as the former is a single-mode Gaussian distribution while the latter can be multi-modal. A more general formulation would be to involve latent variables $\mathbf{z}_t$ following a certain simple prior $p(\mathbf{z}_t)$ into $\boldsymbol{g}_\phi$. Then $q_\phi(\mathbf{y}_t, t|\mathbf{x}_{t+1}) = \mathbb{E}_{p(\mathbf{z}_t)}\left[q_\phi(\mathbf{y}_t, \mathbf{z}_t, t|\mathbf{x}_{t+1})\right]$ can be non-Gaussian (XKV22). However, we empirically find that the simple initializer in Equation 5.4 works well. Compared with the more general formulation, the simple initializer avoids the inference of $\mathbf{z}_t$ which may again require MCMC sampling, and leads to more stable training. Different from (XKV22), samples from the initializer just serves as the starting points and are refined by sampling from the EBM, instead of being treated as the final samples. We follow (HJA20) to set $\tilde{\sigma}_t = \sqrt{\frac{1-\bar{\alpha}_t^2}{1-\bar{\alpha}_{t+1}^2}}\sigma_t$. If we treat the sequence of initializers as the reverse process, such choice of $\tilde{\sigma}_t$ corresponds to the lower bound of the standard deviation given by p_{data} being a delta function (SWM15).

5.3.3 Cooperative training

We jointly train the sequence of EBMs and intializers in a cooperative fashion. Specifically, at each iteration, for a randomly sampled noise level t, we obtain an initial sample $\hat{\mathbf{y}}_t$ from the intializer model. Then a synthesized sample $\tilde{\mathbf{y}}_t$ from $p(\mathbf{y}_t|\mathbf{x}_{t+1})$ is generated by initializing from $\hat{\mathbf{y}}_t$ and running a few steps of Langevin dynamics (Equation 5.3). The parameters of EBM are then updated by maximizing the recovery log-likelihood function (Equation 5.2). The learning gradient of EBM



(a) CDRL training process. In the training phase, we start by selecting a pair of images at noise levels t and $t-1$. The image at noise level t is then fed into the initializer to generate an initial proposal. Subsequently, this initial proposal undergoes refinement through the MCMC process guided by the underlying energy function. The refined sample obtained from this process is utilized to update both the energy function and the initializer.



(b) CDRL Sampling process. The sampling phase starts from Gaussian noise. Starting from the highest noise level, an initial proposal is generated by the initializer that corresponds to that noise level. Subsequently, the samples undergo refinement through MCMC sampling. This denoising process is iteratively repeated to push the noisy image towards lower noise levels until the lowest noise level is reached.

Figure 5.1: Illustration of the Cooperative Diffusion Recovery Likelihood (CDRL) framework

is

$$\nabla_\theta \mathcal{J}_t(\theta) = \nabla_\theta \left[\frac{1}{n} \sum_{i=1}^{n} f_\theta(\mathbf{y}_{t,i}; t) - \frac{1}{n} \sum_{i=1}^{n} f_\theta(\tilde{\mathbf{y}}_{t,i}; t) \right]. \tag{5.5}$$

To train the intializer model that amortizes the MCMC sampling process, we treat the revised sample $\tilde{\mathbf{y}}_t$ by the EBM as the observed data of the initializer model, and estimate the parameters of

the initializer by maximizing log-likelihood:

$$\mathcal{L}_t(\phi) = \frac{1}{n}\sum_{i=1}^{n}\left[-\frac{1}{2\tilde{\sigma}_t^2}\|\tilde{\mathbf{y}}_{t,i} - \boldsymbol{g}_\phi(\mathbf{x}_{t+1,i}; t)\|^2\right]. \tag{5.6}$$

That is, the initializer model learns to absorb the difference between $\hat{\mathbf{y}}_t$ and $\tilde{\mathbf{y}}_t$ at each iteration so that $\hat{\mathbf{y}}_t$ is getting closer to the samples from $p_\theta(\mathbf{y}_t|\mathbf{x}_{t+1})$. In practice, we re-weight $\mathcal{L}_t(\phi)$ across different noise levels by removing the coefficient $\frac{1}{2\tilde{\sigma}_t^2}$, similar to the "simple loss" in diffusion models. The training algorithm is summarized in Algorithm 4 and illustrated in Figure 5.1a

After training, we generate new samples by starting from Gaussian white noise and progressively samples $p_\theta(\mathbf{y}_t|\mathbf{x}_{t+1})$ at decreasingly lower noise levels. For each noise level, an initial proposal is generated from the intializer model, followed by a few steps of Langevin dynamics from the EBM. Algorithm 5 summarizes the sampling algorithm and Figure 5.1b illustrate the sampling process.

5.3.4 Noise variance reduction

We further propose a simple way to reduce the variance of training gradients. In principle, the pair of $\mathbf{x}_t$ (or $\mathbf{y}_t$) and $\mathbf{x}_{t+1}$ is generated by $\mathbf{x}_t \sim \mathcal{N}(\bar{\alpha}_t\mathbf{x}_0, \bar{\sigma}_t^2\boldsymbol{I})$ and $\mathbf{x}_{t+1} \sim \mathcal{N}(\alpha_{t+1}\mathbf{x}_t, \sigma_{t+1}^2\boldsymbol{I})$. Alternatively, we can fix the Gaussian white noise $\mathbf{e} \sim \mathcal{N}(0, \boldsymbol{I})$, and sample pair $(\mathbf{x}'_t, \mathbf{x}'_{t+1})$ by

$$\begin{aligned}\mathbf{x}'_t &= \bar{\alpha}_t\mathbf{x}_0 + \bar{\sigma}_t\mathbf{e}\\ \mathbf{x}'_{t+1} &= \bar{\alpha}_{t+1}\mathbf{x}'_t + \bar{\sigma}_{t+1}\mathbf{e}.\end{aligned} \tag{5.7}$$

In other words, both $\mathbf{x}'_t$ and $\mathbf{x}'_{t+1}$ are linear interpolation between the clean sample $\mathbf{x}_0$ and a sampled white noise image $\mathbf{e}$. $\mathbf{x}'_t$ and $\mathbf{x}'_{t+1}$ have the same marginal distributions as $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$. But $\mathbf{x}'_t$ is deterministic given $\mathbf{x}_0$ and $\mathbf{x}'_{t+1}$, while there's still variance for $\mathbf{x}_t$ given $\mathbf{x}_0$ and $\mathbf{x}_{t+1}$. This schedule is related to the ODE forward process used in flow matching (LCB22) and rectified flow (LGL22).

Algorithm 4 CDRL Training

Input: (1) observed data $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$; (2) Number of noise levels T; (3) Number of Langevin sampling steps K per noise level; (4) Langevin step size at each noise level s_t; (5) Learning rate η_θ for EBM f_θ; (6) Learning rate η_ϕ for initializer g_ϕ;
Output: Parameters θ, ϕ

Randomly initialize θ and ϕ.
repeat
Sample noise level t from $\{0, 1, ..., T-1\}$.
Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$. Let $\mathbf{x}_{t+1} = \bar{\alpha}_{t+1}\mathbf{x}_0 + \bar{\sigma}_{t+1}\boldsymbol{\epsilon}$, $\mathbf{y}_t = \alpha_{t+1}(\bar{\alpha}_t\mathbf{x}_0 + \bar{\sigma}_t\boldsymbol{\epsilon})$.
Generate the initial sample $\hat{\mathbf{y}}_t$ following Equation 5.4.
Generate the refined sample $\mathbf{y}_t$ by running K steps of Langevin dynamics starting from $\hat{\mathbf{y}}_t$ following Equation 5.3.
Update EBM parameter θ following the gradients in Equation 5.5.
Update initializer parameter ϕ by maximizing Equation 5.6.
until converged

Algorithm 5 CDRL Sampling

Input: (1) Number of noise levels T; (2) Number of Langevin sampling steps K at each noise level; (3) Langevin step size at each noise level δ_t; (4) Trained EBM f_θ; (5) Trained initializer g_ϕ;
Output: Samples $\tilde{\mathbf{x}}_0$

Randomly initialize $\mathbf{x}_T \sim \mathcal{N}(0, \boldsymbol{I})$.
for $t = T-1$ **to** 0 **do**
Generate initial proposal $\hat{\mathbf{y}}_t$ following Equation 5.4.
Update $\hat{\mathbf{y}}_t$ to $\tilde{\mathbf{y}}_t$ by K iterations of Equation 5.3.
Let $\tilde{\mathbf{x}}_t = \tilde{\mathbf{y}}_t / \alpha_{t+1}$.
end for

5.3.5 Conditional generation and classifier-free guidance

(HS22) proposed classifier-free guidance that greatly improves the sample quality of conditional diffusion models, and trades-off between sample quality and sample diversity by adjusting the guidance strength. Given the close connection between EBMs and diffusion models, we show that it is possible to apply classifier-free guidance in CDRL as well. Specifically, suppose c is the context (e.g., a label or a text description). At each noise level we jointly estimate an unconditional EBM $p_\theta(\mathbf{y}_t) \propto \exp(f_\theta(\mathbf{y}_t; t))$ and a conditional EBM $p_\theta(\mathbf{y}_t|c) \propto \exp(f_\theta(\mathbf{y}_t; c, t))$. By Bayes rule:

$$p_\theta(c|\mathbf{y}_t) = \frac{p_\theta(c, \mathbf{y}_t)}{p_\theta(\mathbf{y}_t)} = \frac{p_\theta(\mathbf{y}_t|c)p(c)}{p_\theta(\mathbf{y}_t)}. \tag{5.8}$$

With classifier-free guidance, we assume that the log-density of $\mathbf{y}_t$ is scaled to

$$\log \tilde{p}_\theta(\mathbf{y}_t|c) = \log\left[p_\theta(\mathbf{y}_t|c)p_\theta(c|\mathbf{y}_t)^w\right] + \text{const.} = (w+1)f_\theta(\mathbf{y}_t; c, t) - wf_\theta(\mathbf{y}_t; t) + \text{const.}, \tag{5.9}$$

where w controls the guidance strength. Similarly, for the initializer model, we jointly estimate an unconditional model $q_\phi(\mathbf{y}_t|\mathbf{x}_{t+1}) \sim \mathcal{N}(\boldsymbol{g}_\phi(\mathbf{x}_{t+1}; t), \tilde{\sigma}_t^2\boldsymbol{I})$ and a conditional model $q_\phi(\mathbf{y}_t|c, \mathbf{x}_{t+1}) \sim \mathcal{N}(\boldsymbol{g}_\phi(\mathbf{x}_{t+1}; c, t), \tilde{\sigma}_t^2\boldsymbol{I})$. Since both models follow Gaussian distributions, the scaled conditional distribution with classifier-free guidance is still a Gaussian (DN21):

$$\tilde{q}_\phi(\mathbf{y}_t|c, \mathbf{x}_{t+1}) \propto q_\phi(\mathbf{y}_t|c, \mathbf{x}_{t+1})q_\phi(c|\mathbf{y}_t, \mathbf{x}_{t+1})^w \sim \mathcal{N}\left((w+1)\boldsymbol{g}_\phi(\mathbf{x}_{t+1}; c, t) - w\boldsymbol{g}_\phi(\mathbf{x}_{t+1}; t), \tilde{\sigma}_t^2\boldsymbol{I}\right). \tag{5.10}$$

5.3.6 Compositionality in energy-based model

One attractive property of EBMs is compositionality: one can combine multiple EBMs conditioned on individual concepts, and re-normalize it to create a new distribution conditioned on the intersection of those concepts. Specifically, given two EBMs $p_\theta(\mathbf{x}|c_1) \propto \exp(f_\theta(\mathbf{x}; c_1))$ and $p_\theta(\mathbf{x}|c_2) \propto \exp(f_\theta(\mathbf{x}; c_2))$ that are conditional on two seperate concepts, (DLM20; LJP23) constructs a new EBM conditional on both concepts as $p_\theta(\mathbf{x}|c_1, c_2) \propto \exp(f_\theta(\mathbf{x}; c_1) + f_\theta(\mathbf{x}; c_2))$ based on the production of expert (Hin02a). Specifically, suppose the two concepts c_1 and c_2 are conditionally independent given the observed data $\mathbf{x}$. Then we have

$$\begin{aligned}
\log p_\theta(\mathbf{x}|c_1, c_2) &= \log p_\theta(c_1, c_2|\mathbf{x}) + \log p_\theta(\mathbf{x}) + \text{const.} \\
&= \log p_\theta(c_1|\mathbf{x}) + \log p_\theta(c_2|\mathbf{x}) + \log p_\theta(\mathbf{x}) + \text{const.} \\
&= \log p_\theta(\mathbf{x}|c_1) + \log p_\theta(\mathbf{x}|c_2) - \log p_\theta(\mathbf{x}) + \text{const.}
\end{aligned}$$

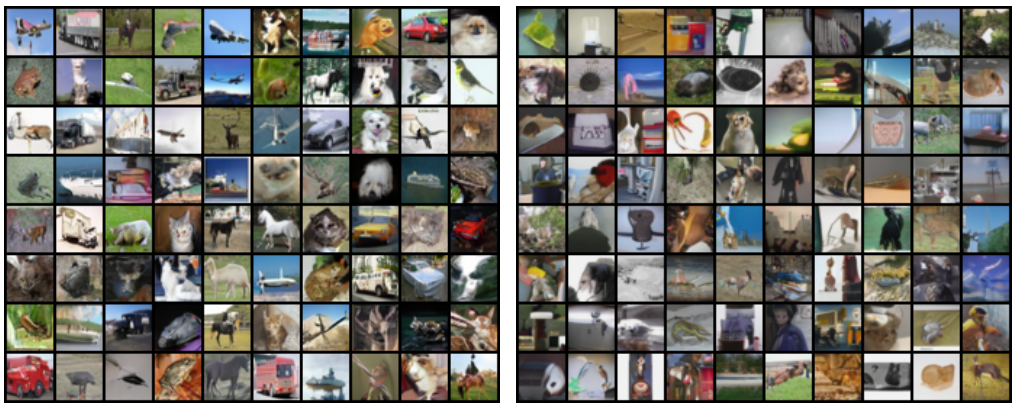
The composition can be generalized to include arbitrary number of concepts. Suppose we have

M conditionally independent concepts $c_i, i = 1, ..., M$, then

$$\log p_\theta(\mathbf{x}|c_i, i = 1, ..., M) = \sum_{i=1}^{M} \log p_\theta(\mathbf{x}|c_i) - (M-1) \log p_\theta(\mathbf{x}) + \text{const.} \tag{5.11}$$

We can combine the compositional log-density (Equation 5.11) with classifier-free guidance (Equation 5.9) to further improve the alignment of generated samples with given concepts. The scaled log-density function is given by

$$\begin{aligned} &\log \left[p(\mathbf{x}|c_i, i = 1, ..., M) p(c_i, i = 1, ..., M|\mathbf{x})^w \right] \\ &= (w+1) \sum_{i=1}^{M} \log p_\theta(\mathbf{x}|c_i) - (Mw + M - 1) \log p(\mathbf{x}) + \text{const.} \end{aligned} \tag{5.12}$$



(a) CIFAR-10 (b) ImageNet (32×32)

Figure 5.2: Unconditional generated examples on CIFAR-10 and ImageNet (32×32) datasets.

5.4 Experiments

We evaluate the performance of our model across various scenarios. Specifically, Section 5.4.1 demonstrates the capacity of unconditional generation. Section 5.4.2 highlights the potential of our model to further optimize sampling efficiency. The focus shifts to conditional generation and classifier-free guidance in Section 5.4.3. Section 5.4.5 elucidates the power of our model in performing likelihood estimation and OOD detection. Section 5.4.6 showcases compositional

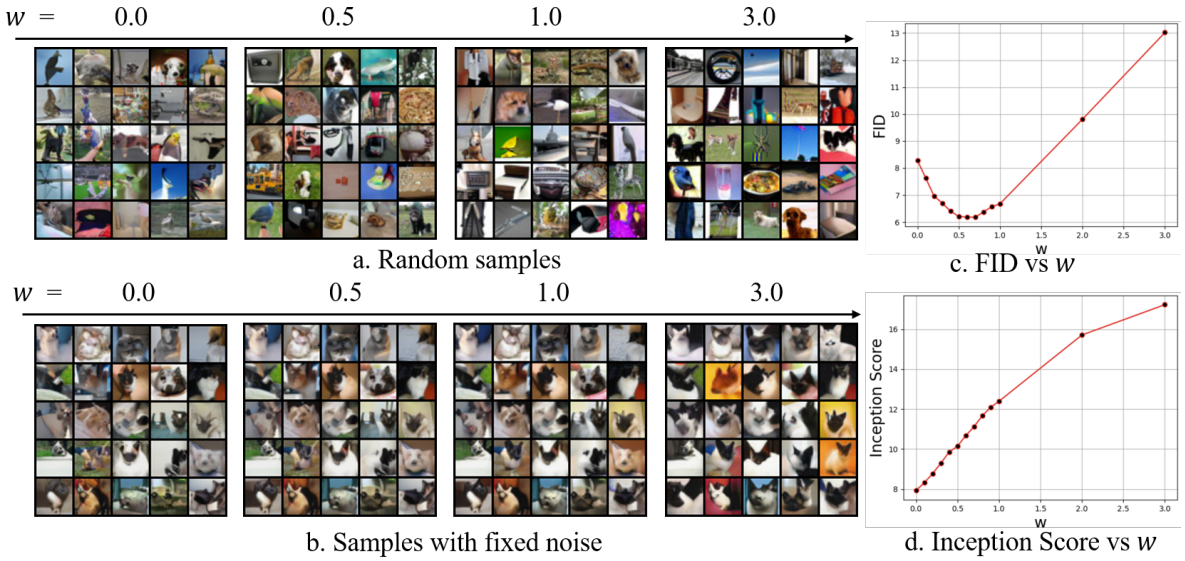


Figure 5.3: Conditional generation on ImageNet (32×32) dataset with a classifier-free guidance. (a) Random image samples generated with different guided weights $w = 0.0, 0.5, 1.0$ and 3.0; (b) Samples generated with a fixed noise under different guided weights. The class label is set to be the category of Siamese Cat. Sub-images presented at the same position depict samples with identical random noise and class label, differing only in their guided weights; (c) A curve of FID scores across different guided weights; (d) A curve of Inception scores across different guided weights.

generation. Section 5.4.7 showcases image inpainting results with our trained models. Section 5.4.8 discusses our implementation details. Section 5.4.9 compares the sampling time between our approach and other EBM models. Section 5.4.10 provides experimental results for understanding the role of EBM and initializer in the generation process. Section 5.4.11 contains the ablation study. We designate our approach as "CDRL" in the following sections.

Our experiments primarily involve three datasets: (i) CIFAR-10 (KH09) comprises images from 10 categories, with 50k training samples and 10k test samples at a resolution of 32×32 pixels. We use its training set for evaluating our model in the task of unconditional generation. (ii) ImageNet (DDS09) contains approximately 1.28M images from 1000 categories. We use its training set for both conditional and unconditional generation, focusing on a downsampled version (32×32) of the dataset. (iii) CelebA (LLW15) consists of around 200k human face images, each annotated with attributes. We downsample each image of the dataset to the size of 64×64 pixels and utilize the

resized dataset for compositionality and image inpainting tasks.

5.4.1 Unconditional image generation

We first showcase our model's capabilities in unconditional image generation on CIFAR-10 and ImageNet datasets. The resolution of each image is 32×32 pixels. FID scores (HRU17b) on these two datasets are reported in Tables 5.1 and 5.3, respectively, with generated examples displayed in Figure 5.2. We adopt the EBM architecture proposed in (GSP21a). Additionally, we utilize a larger version called "CDRL-large", which incorporates twice as many channels in each layer. For the initializer network, we follow the structure of (ND21), utilizing a U-Net (RFB15) but halving the number of channels. Compared to (GSP21a), CDRL achieves significant improvements in FID scores. Furthermore, CDRL uses the same number of noise levels (6 in total) as DRL but requires only half the MCMC steps at each noise level, reducing it from 30 to 15. This substantial reduction in computational costs is noteworthy. With the large architecture, CDRL achieves a FID score of 3.68 on CIFAR-10 and 9.35 on ImageNet (32×32). These results, to the best of our knowledge, are the state-of-the-art among existing EBM frameworks and are competitive with other strong generative model classes such as GANs and diffusion models.

5.4.2 Sampling efficiency

Similar to the sampling acceleration techniques employed in the diffusion model (SME21; LRL22; LZB22), we foresee the development of post-training techniques to further accelerate CDRL sampling. Although designing an advanced MCMC sampling algorithm could be a standalone project, we present a straightforward yet effective sampling adjustment technique to demonstrate CDRL's potential in further reducing sampling time. Specifically, we propose to decrease the number of sampling steps while simultaneously adjusting the MCMC sampling step size to be inversely proportional to the square root of the number of sampling steps. As shown in Table 5.2, while we train CDRL with 15 MCMC steps at each noise level, we can reduce the number of

Table 5.1: Comparison of FID scores for unconditional generation on CIFAR-10.

Models	**FID** $\downarrow$
EBM based method	
NT-EBM (NGS22)	78.12
LP-EBM (PHN20)	70.15
Adaptive CE (XH22)	65.01
EBM-SR (NHZ19)	44.50
JEM (GWJ20)	38.40
EBM-IG (DM19)	38.20
EBM-FCE (GNK20)	37.30
CoopVAEBM (XZL21)	36.20
CoopNets (XLG18)	33.61
Divergence Triangle (HNZ20)	30.10
VARA (GKH21a)	27.50
EBM-CD (DLT21a)	25.10
GEBM (AZG21)	19.31
HAT-EBM (HNM22)	19.30
CF-EBM (ZXL21)	16.71
CoopFlow (XZL22)	15.80
CLEL-base (LJP23)	15.27
VAEBM (XKK21)	12.16
DRL (GSP21a)	9.58
CLEL-large (LJP23)	8.61
EGC (Unsupervised) (GMJ23)	5.36
CDRL (Ours)	4.31
CDRL-large (Ours)	3.68

Models	**FID** $\downarrow$
Other likelihood based method	
VAE (KW14b)	78.41
PixelCNN (SKC17)	65.93
PixelIQN (ODM18)	49.46
Residual Flow (CBD19)	47.37
Glow (KD18)	45.99
DC-VAE (PLL21)	17.90
GAN based method	
WGAN-GP(GAA17)	36.40
SN-GAN (MKK18)	21.70
BigGAN (BDS19)	14.80
StyleGAN2-DiffAugment (ZLL20)	5.79
Diffusion-GAN (XKV22)	3.75
StyleGAN2-ADA (KAH20)	2.92
Score based and Diffusion method	
NCSN (SE19)	25.32
NCSN-v2 (SE20)	10.87
NCSN++ (SSK21)	2.20
DDPM Distillation (LL21)	9.36
DDPM++(VP, NLL) (KSS21)	3.45
DDPM (HJA20)	3.17
DDPM++(VP, FID) (KSS21)	2.47

MCMC steps to 8, 5, and 3 during the inference stage, without sacrificing much perceptual quality.

5.4.3 Conditional synthesis with classifier-free guidance

We evaluate our model for conditional generation on the ImageNet32 dataset, employing classifier-free guidance as outlined in Section 5.3.5. Generation results for varying guided weights w are displayed in Figure 5.3. As the value of w increases, the quality of samples improves, and the conditioned class features become more prominent, although diversity may decrease. This trend is also evident from the FID and Inception Score (SGZ16) curves shown in Figures 5.3(c) and 5.3(d). While the Inception Score consistently increases (improving quality), the FID metric first drops

(improving quality) and then increases (worsening quality), obtaining the optimal value of 6.18 (lowest value) at a guidance weight of 0.7.

Table 5.2: FID for CIFAR-10 with sampling adjustment.

Models	**Number of noise level $\times$ Number of MCMC steps**	**FID $\downarrow$**
DRL (GSP21a)	$6 \times 30 = 180$	9.58
CDRL	$6 \times 15 = 90$	4.31
CDRL (step 8)	$6 \times 8 = 48$	4.58
CDRL (step 5)	$6 \times 5 = 30$	5.37
CDRL (step 3)	$6 \times 3 = 18$	9.67

Table 5.3: FID for ImageNet (32×32) unconditional generation.

Models	**FID $\downarrow$**
EBM-IG (DM19)	60.23
PixelCNN (SKC17)	40.51
EBM-CD (DLT21a)	32.48
CF-EBM (ZXL21)	26.31
CLEL-base (LJP23)	22.16
DRL (GSP21a)	- (not converge)
DDPM++(VP, NLL) (KSS21)	8.42
CDRL (Ours)	9.35

5.4.4 Generating High-Resolution Images

The recent trend in generative modeling of high-resolution images involves either utilizing the latent space of a VAE, as demonstrated in latent diffusion (RBL22), or initially generating a low-resolution image and then gradually expanding it, as exemplified by techniques like Imagen (SCS22). This process often reduce the modeled space to dimensions such as 32×32 or 64×64, which aligns with the resolutions that we used in our experiments in the main text. Here, we conduct additional experiments by learning CDRL following (RBL22). We conduct experiments on the CelebA-HQ dataset, and the generated samples are shown in Figure 5.4. Additionally, we report the FIDs in Table 5.4.

Table 5.4: Comparison of FIDs on the CelebA-HQ (256 x 256) dataset

Model	**FID score**
GLOW (KD18)	68.93
VAEBM (XKK21)	20.38
ATEBM (YLR22)	17.31
VQGAN+Transformer (ERO21)	10.2
LDM (RBL22)	5.11
CDRL(ours)	10.74

5.4.5 Likelihood Estimation and Out-Of-Distribution Detection

A distinctive feature of the EBM is its ability to model the unnormalized log-likelihood directly using the energy function. This capability enables it to perform tasks beyond generation. In this section, we first showcase the capability of the CDRL in estimating the density of a 2D checkerboard distribution. Experimental results are presented in Figure 5.5, where we illustrate observed samples, the fitted density, and the generated samples at each noise level, respectively. These results confirm CDRL's ability to accurately estimate log-likelihood while simultaneously generating valid samples.

Moreover, we demonstrate CDRL's utility in out-of-distribution (OOD) detection tasks. For this endeavor, we employ the model trained on CIFAR-10 as a detector and use the energy at the lowest noise level to serve as the OOD prediction score. The AUROC score of our CDRL model, with CIFAR-10 interpolation, CIFAR-100, and CelebA data as OOD samples, is provided in Table 5.5. CDRL achieves strong results in OOD detection comparing with the baseline approaches.

5.4.6 Compositionality

To evaluate the compositionality of EBMs, we conduct experiments on CelebA (64×64) datasets with *Male*, *Smile*, and *Young* as the three conditional concepts. We estimate EBMs conditional on each single concept separately, and assume simple unconditional initializer models. Classifier-free guidance is adopted when conducting compositional generation (Equation 5.12). Specifically, we treat images with a certain attribute value as individual classes. We randomly assign each image

Figure 5.4: Samples generated by CDRL model trained on the CelebAHQ (256×256) dataset.

in a training batch to a class based on the controlled attribute value. For example, an image with Male=True and Smile=True may be assigned to class 0 if the Male attribute is picked or class 2 if the Smile attribute is picked. For the conditional network structure, we make EBM f_θ conditional on attributes c_i and use an unconditional initializer model g_ϕ to propose the initial distribution. We focus on showcasing the compositionality ability of EBM itself, although it is also possible to use a conditional initializer model similar to Section 5.3.5. Our results are displayed in Figure 5.6, with images generated at a guided weight of $w = 3.0$. Images generated with composed attributes following Equation 5.12 contain features of both attributes, and increasing the guided weight makes the corresponding attribute more prominent. This demonstrates CDRL's ability and the effectiveness

Table 5.5: AUROC scores in OOD detection using CDRL and other explicit density models on CIFAR-10

	Cifar-10 interpolation	Cifar-100	CelebA
PixelCNN (SKC17)	0.71	0.63	-
GLOW (KD18)	0.51	0.55	0.57
NVAE (VK20b)	0.64	0.56	0.68
EBM-IG (DM19)	0.70	0.50	0.70
VAEBM (XKK21)	0.70	0.62	0.77
EBM-CD (DLT21a)	0.65	0.83	-
CLEL-Base (LJP23)	0.72	0.72	0.77
CDRL (ours)	0.75	0.78	0.84

of Equation 5.12.

5.4.7 Image Inpainting

We demonstrate the inpainting ability of our learned model on the 64×64 CelebA dataset. Each image is masked, and our model is tasked with filling in the masked area. We gradually add noise to the masked image up to the final noise level, allowing the model to denoise the image progressively,

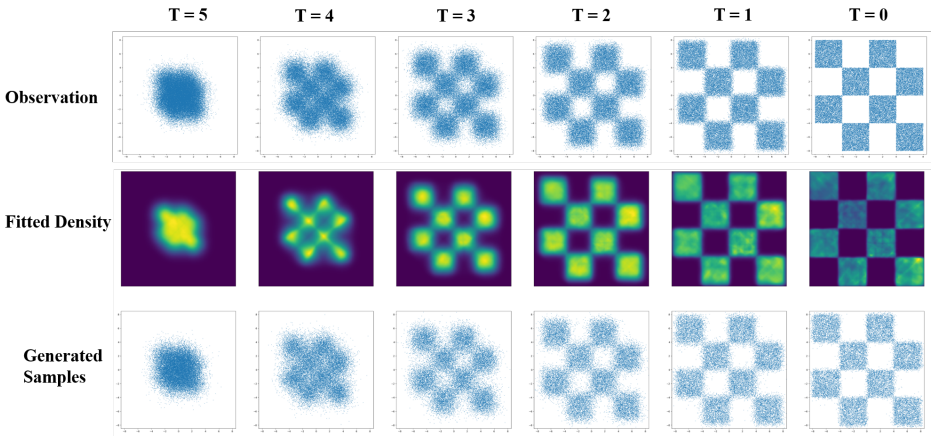
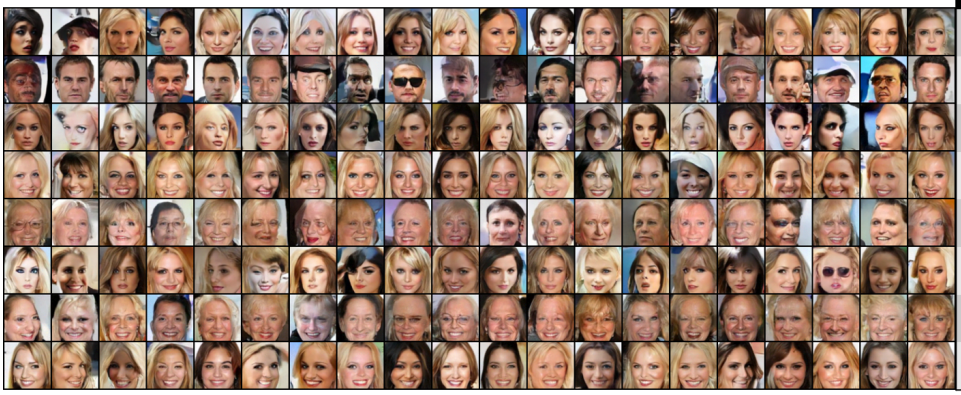


Figure 5.5: The results of density estimation using CDRL for a 2D checkerboard distribution. The number of noise levels in the CDRL is set to be 5. Top: observed samples at each noise level. Middle: density fitted by CDRL at each noise level. Bottom: generated samples at each noise level.



Male	Smile	Young
×	-	-
√	-	-
×	×	-
×	√	-
×	-	×
×	-	√
×	√	×
×	√	√

Figure 5.6: Results of attribute-compositional generation on CelebA (64×64) with guided weight $w = 3$. Left: generated samples under different attribute compositions. Right: control attributes ("$\surd$", "$\times$" and "-" indicate "True", "False" and "No Control" respectively).

similar to the standard generation process. During inpainting, only the masked area is updated, while the values in the unmasked area are retained. This is achieved by resetting the unmasked area values to the current noisy version after each Langevin update step of the EBM or initializer proposal step. Our results, depicted in Figure 5.7, include two types of masking: a regular square mask and an irregularly shaped mask. In Figure 5.7, the first two columns respectively display the original images and the masked images, while the other columns show the corresponding inpainting results. CDRL successfully inpaints valid and diverse values in the masked area, producing inpainted results that differ from the observations. This suggest that CDRL does not merely memorize data because it fills novel and meaningful content into unobserved areas based on the statistical features of the dataset.

5.4.8 Training Details

5.4.8.1 Network Architectures

We adopt the EBM architecture from (GSP21a), starting with a 3×3 convolution layer with 128 channels (The number of channels is doubled to 256 in the CDRL-large configuration). We use several downsample blocks for resolution adjustments, each containing multiple residual blocks.

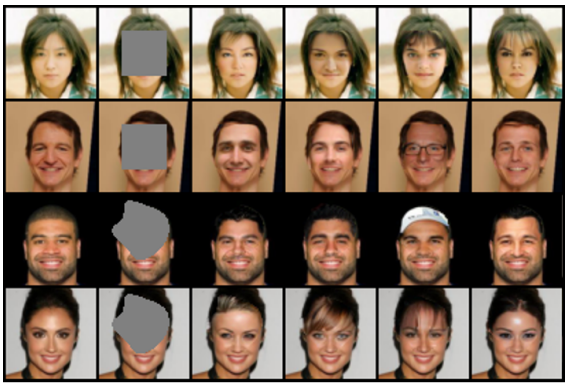


Figure 5.7: Results of Image inpainting on CelebA (64×64) dataset. The first two rows utilize square masks, while the last two rows use irregular masks. The first column displays the original images. The second column shows the masked images. Columns three to six display inpainted images using different initialization noises.

All downsampling blocks, except the last one, include a 2×2 average pooling layer. Spectral normalization is applied to all convolution layers for stability, while ReLU activation is applied to the final feature map. The energy output is obtained by summing the values over spatial and channel dimensions. The architectures of EBM building blocks are shown in Table 5.6, and the hyperparameters of network architecture are displayed in Table 5.7.

For the initializer network, we follow (ND21) to utilize a U-Net (RFB15) while halving the number of channels. This reduction effectively decreases the size of the initializer model. For an image with a resolution of 32×32 pixels, we have feature map resolutions of 32×32, 16×16, and 4×4. When dealing with 64×64 images, we include an additional feature map resolution of 64×64. All feature map channel numbers are set to 64, with attention applied to resolutions of 16×16 and 8×8. Our initializer directly predicts the noised image $\tilde{\mathbf{y}}_t$ at each noise level t, while the DDPM in (HJA20) predicts the total injected noise ϵ.

For the class-conditioned generation task, we map class labels to one-hot vectors and use a fully-connected layer to map these vectors to class embedding vectors with the same dimensions as time embedding vectors. The class embedding is then added to the time embedding. We set the time embedding dimension to 512 for EBM and 256 for the initializer in the CDRL setting. In the CDRL-large setting, the time embedding dimension increases to 1024 for EBM, while the one in

the initializer remains unchanged.

Table 5.6: Building blocks of the EBM in CDRL.

(a) ResBlock

leakyReLU, 3×3 Conv2D
+ Dense(leakyReLU(temb))
leakyReLU, 3×3 Conv2D
+ Input

(b) Downsample Block

N ResBlocks
Downsample 2×2

(c) Time Embedding

Sinusoidal Embedding
Dense, leakyReLU
Dense

Table 5.7: Hyperparameters for EBM architectures in different settings.

Model	# of Downsample Blocks	N (# of Resblocks in Downsample Block)	# of channels in each resolution
CDRL (32×32)	4	8	(128, 256, 256, 256)
CDRL-large (32×32)	4	8	(256, 512, 512, 512)
Compositionality Experiment	5	2	(128, 256, 256, 256, 256)
Inpainting Experiment	5	8	(128, 256, 256, 256, 256)

5.4.8.2 Hyperparameters

We set the learning rate of EBM to be $\eta_\theta = 1e-4$ and the learning rate of initializer to be $\eta_\phi = 1e-5$. We use linear warm up for both EBM and initializer and let the initializer to start earlier than EBM. More specifically, given training iteration iter, we have:

$$\begin{aligned} \eta_\theta &= \min(1.0, \frac{\mathrm{iter}}{10000}) \times 1e-4 \\ \eta_\phi &= \min(1.0, \frac{\mathrm{iter}+500}{10000}) \times 1e-5 \end{aligned} \tag{5.13}$$

We use the Adam optimizer (KB15c; LH19) to train both the EBM and the initializer, with $\beta = (0.9, 0.999)$ and a weight decay equal to 0.0. We also apply exponential moving average with a decay rate equal to 0.9999 to both the EBM and the initializer. Training is conducted across 8

Nvidia A100 GPUs, typically requiring approximately 400k iterations, which spans approximately 6 days.

Following (GSP21a), we use a re-parameterization trick to calculate the energy term. Our EBM is constructed across noise levels $t = 0, 1, 2, 3, 4, 5$ and we assume the distribution at noise level $t = 6$ is a simple Normal distribution during sampling. Given $\mathbf{y}_t$ under noise level t, suppose we denote the output of the EBM network as $\hat{f}_\theta(\mathbf{y}_t, t)$, then the true energy term is given by $f_\theta(\mathbf{y}_t, t) = \frac{\hat{f}_\theta(\mathbf{y}_t, t)}{s_t^2}$, where s_t is the Langevin step size at noise level t. In other words, we parameterize the energy as the product of the EBM network output and a noise-level dependent coefficient, setting this coefficient equal to the square of the Langevin step size. We use 15 steps of Langevin updates at each noise level, with the Langevin step size at noise level t given by

$$s_t^2 = 0.054 \times \bar{\sigma}_t \times \sigma_{t+1}^2, \tag{5.14}$$

where σ_{t+1}^2 is the variance of the added noise at noise level $t + 1$ and $\bar{\sigma}_t$ is the standard deviation of the accumulative noise at noise level t. During the generation process, we begin by randomly sampling $\mathbf{x}_6 \sim \mathcal{N}(0, \boldsymbol{I})$ and perform denoising using both the initializer and the Langevin Dynamics of the EBM, which follows Algorithm 5. After obtaining samples $\mathbf{x}_0$ at the lowest noise level $t = 0$, we perform an additional denoising step, where we disable the noise term in the Langevin step, to further enhance its quality. More specifically, we follow Tweedie's formula (Efr11; Rob92), which states that given $\mathbf{x} \sim p_{data}(\mathbf{x})$ and a noisy version image $\mathbf{x}'$ with conditional distribution $p(\mathbf{x}'|\mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma^2 \boldsymbol{I})$, the marginal distribution can be defined as $p(\mathbf{x}') = \int p_{data}(\mathbf{x}) p(\mathbf{x}'|\mathbf{x}) d\mathbf{x}$. Consequently, we have

$$\mathbb{E}(\mathbf{x}|\mathbf{x}') = \mathbf{x}' + \sigma^2 \nabla_{\mathbf{x}'} \log p(\mathbf{x}'). \tag{5.15}$$

In our case, we have $p(\mathbf{x}_t|\bar{\alpha}_t \mathbf{x}_0) = \mathcal{N}(\bar{\alpha}_t \mathbf{x}_0, \bar{\sigma}_t^2 \boldsymbol{I})$ and we use EBM to model the marginal distribu-

tion of $\mathbf{x}_t$ as $p_{\theta,t}(\mathbf{x}_t)$, thus

$$\begin{aligned}\mathbb{E}(\bar{\alpha}_t \mathbf{x}_0 | \mathbf{x}_t) &= \mathbf{x}_t + \bar{\sigma}_t^2 \nabla_{\mathbf{x}_t} \log p_{\theta,t}(\mathbf{x}_t), \\ \mathbb{E}(\mathbf{x}_0 | \mathbf{x}_t) &= \frac{\mathbf{x}_t + \bar{\sigma}_t^2 \nabla_{\mathbf{x}_t} \log p_{\theta,t}(\mathbf{x}_t)}{\bar{\alpha}_t}.\end{aligned} \tag{5.16}$$

Suppose the samples we obtain at $t = 0$ are denoted as $\mathbf{x}_0$. These samples actually contains a small amount of noise corresponding to $\bar{\alpha}_0$, thus, we may use Equation 5.16 to further denoise them. In practice, we find that enlarging the denoising step by multiplying the gradient term $\nabla_{\mathbf{x}_t} \log p_{\theta,t}(\mathbf{x}_t)$ by a coefficient larger than 1.0 yields better results. We set this coefficient to be 2.0 in our experiments.

5.4.8.3 Noise Schedule and Conditioning Input

We improve upon the noise schedule and the conditioning input of DRL (GSP21a). Let $\lambda_t = \log \frac{\bar{\alpha}_t^2}{\bar{\sigma}_t^2}$ represent the logarithm of signal-to-noise ratio at noise level t. Inspired by (KSP21), we utilize λ_t as the conditioning input of the noise level and feed it to the networks f_θ and $\boldsymbol{g}_\phi$ instead of directly using t.

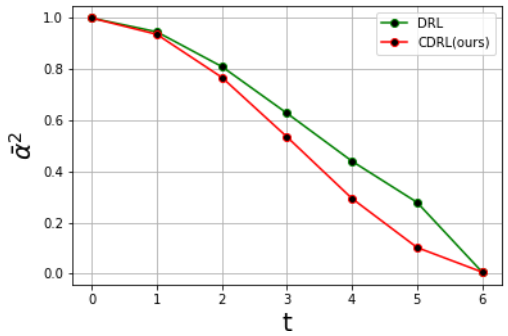


Figure 5.8: Noise schedule. The green line represents the noise schedule used by DRL (GSP21a) while the red line depicts the noise schedule employed by our CDRL.

For the noise schedule, we keep the design of using 6 noise levels as in DRL. Inspired by (ND21), we construct a cosine schedule such that λ_t is defined as $\lambda_t = -2\log(\tan(at + b))$, where

a and b are calculated from the maximum log SNR (denoted as $\lambda_{\max}$) and the minimum log SNR (denoted as $\lambda_{\min}$) using

$$b = \arctan(\exp(-0.5\lambda_{\max})), \tag{5.17}$$

$$a = \arctan(\exp(-0.5\lambda_{\min})) - b. \tag{5.18}$$

We set $\lambda_{\max} = 9.8$ and $\lambda_{\min} = -5.1$ to correspond with the standard deviation $\bar{\alpha}_t$ of the accumulative noise in the original Recovery Likelihood model (T6 setting) at the highest and lowest noise levels. Figure 5.8 illustrates the noise schedule of DRL alongside our proposed schedule. In contrast to the DRL's original schedule, our proposed schedule places more emphasis on regions with lower signal-to-noise ratios, which are vital for generating low-frequency, high-level concepts in samples.

5.4.9 Sampling Time

In this section, we measure the sampling time of CDRL and compare it with the following models: 1) CoopFlow (XZL22), which composes a EBM and a Normalizing Flow model; 2) VAEBM (XKK21), which composes a VAE with an EBM and achieves strong generation performance; 3) The original DRL (GSP21a) model with 30 step MCMC steps at each noise level. We run the sampling process of each model individually on a single A6000 GPU to generate a batch of 100 samples on the Cifar10 dataset. Our CDRL model generates samples with better quality with relatively less time. And after applying the sampling adjustment techniques, the sampling time can be further reduced without hurting much sampling quality.

5.4.10 Analyzing the Effects of the Initializer and the EBM

To gain deeper insights into the roles of the initializer and the EBM in the CDRL in image generation, we conduct two additional experiments using a pretrained CDRL model on the ImageNet Dataset (32×32). We evaluate two generation options: (a) images generated using only the initializer's proposal, without the EBM's Langevin Dynamics at each noise level, and (b) images generated

Table 5.8: Comparison of different EBMs in terms of sampling time and number of MCMC steps. The sampling time are measured in second.

Method	**Number of MCMC steps**	**Sampling Time**	**FID** $\downarrow$
CoopFlow (XZL22)	30	2.5	15.80
VAEBM (XKK21)	16	21.3	12.16
DRL (GSP21a)	$6 \times 30 = 180$	23.9	9.58
CDRL	$6 \times 15 = 90$	12.2	4.31
CDRL (8 steps)	$6 \times 8 = 48$	6.5	4.58
CDRL (5 steps)	$6 \times 5 = 30$	4.2	5.37
CDRL (3 steps)	$6 \times 3 = 18$	2.6	9.67

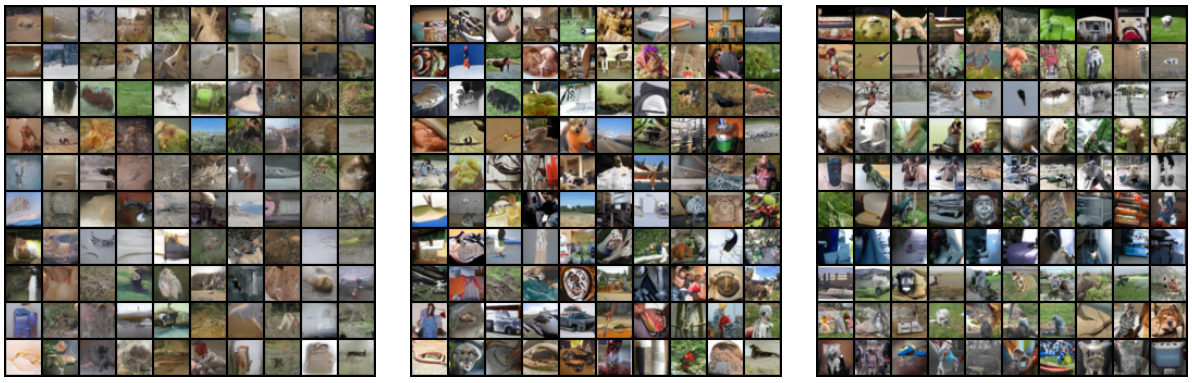
with the full CDRL model, which includes the initializer's proposal and 15-step Langevin updates at each noise level. As shown in Figure 5.9a and 5.9b, the initializer captures the rough outline of the object, while the Langevin updates by the EBM improve the details of the object. Furthermore, in Figure 5.9c, we display samples generated by fixing the initial noise image and sample noise of each initializer proposal step. The outcomes demonstrate that images generated with the same initialization noises share basic elements but differ in details, highlighting the impact of both the initializer and the Langevin sampling. The initializer provides a starting point, while the Langevin sampling process enriches details.

5.4.11 Ablation Study

In this section, we conduct an ablation study to analyze the effectiveness of each component of our CDRL model. We have previously described three main techniques in our main paper that contribute significantly to our CDRL model: (1) the new noise schedule design, (2) the cooperative training algorithm, and (3) noise variance reduction. We demonstrate the impact of each of these techniques by comparing our CDRL model with the following models:

1. The original diffusion recovery likelihood (DRL) model (GSP21a) as a baseline.

2. A model trained without using the cooperative training. This corresponds to the DRL but



(a) Initializer only (b) Full CDRL model (c) Fixing the initialization noise

Figure 5.9: Illustration of the effects of the initializer and the EBM on the image generation process using a CDRL model pretrained on the ImageNet Dataset (32×32). (a) Samples generated using only the proposal of the initializer; (b) Samples generated by the full CDRL model; (c) Samples generated by fixing the initial noise image and the sample noise of each initialization proposal step. Each row of images shared the same initial noise image and the sample noise of each initialization proposal step, but differed in the noises of Langevin sampling process at each noise level.

using the same noise schedule and conditioning input as CDRL.

3. CDRL without using noise reduction.

4. Similar to (XKV22), we use the initializer to predict the clean image $\hat{\mathbf{x}}_0$ and then transform it to $\hat{\mathbf{y}}_t$. Note that our CDRL uses the initializer to directly predict $\hat{\mathbf{y}}_t$.

5. Similar to (HJA20), we use initializer to directly output the prediction of total added noise $\hat{\epsilon}$ and then transform it to $\hat{\mathbf{y}}_t$.

6. Compared with the noise schedule used in the original DRL(GSP21a) paper, the proposed one used in our CDRL places more emphasis on the high-noise area where $\bar{\alpha}$ is close to 0. We train a CDRL model with the original DRL noise schedule but with 2 additional noise levels in the high-noise region for comparison.

7. As depicted in Equation 5.6 in the main paper, our cooperative training algorithm involves the initializer learning from the revised sample $\tilde{\mathbf{y}}_t$ at each step. A natural question arises: should

we instead regress it directly on the data $\mathbf{y}_t$? To answer this, we train a model, in which the initializer directly learns from $\mathbf{y}_t$ at each step.

We ensure that all models share the same network structure and training settings on the CIFAR-10 dataset and differ only in the aforementioned ways. As shown in Table 5.9, our full model performs the best among these settings, which justifies our design choices.

Table 5.9: Ablation study on the CIFAR-10 dataset.

Models	**FID** $\downarrow$
DRL (GSP21a)	9.58
CDRL without cooperative training	6.47
CDRL without noise reduction	5.51
CDRL with an initializer that predicts $\hat{\mathbf{x}}_0$	5.17
CDRL with an initializer that predicts $\hat{\epsilon}$	4.95
CDRL using a noise schedule in DRL-T8	4.94
CDRL with an initializer that learns from $\mathbf{y}_t$	5.95
CDRL (full)	4.31

5.4.12 Effects of Number of Noise Levels and Number of Langevin Steps

We test whether the noise level can be further reduced. The results in Table 5.10a show that further reducing noise level to 4 can make model more unstable, even if we increase the number of the Langevin sample steps K. On the other hand, reducing T to 5 yields reasonable but slightly worse results. In Table 5.10b, we show the effect of changing the number of Langevin steps K. The results show that, on one hand, decreasing K to 10 yields comparable but slightly worse results. On the other hand, increasing K to 30 doesn't lead to better results. This observation aligns wit the finding from (GSP21a). The observation of changing K implies that simply increasing the number of Langevin steps doesn't significantly enhance sample quality, thereby verifying the effectiveness of the initializer in our model.

Table 5.10: Comparison of CDRL models with varying numbers of noise levels T and varying numbers of Langevin steps K. FIDs are reported on the Cifar-10 dataset.

(a) Results of CDRL models with varying T

Model	FID ↓
$T = 4$ ($K = 15, 20, 30$)	not converge
$T = 5$ ($K = 15$)	5.08
$T = 6$ ($K = 15$)	4.31

(b) Results of CDRL models with varying K

Model	FID ↓
$T = 6$ ($K = 10$)	4.50
$T = 6$ ($K = 15$)	4.31
$T = 6$ ($K = 30$)	5.08

CHAPTER 6

Conclusion and Future Work

6.1 Research Summary

In this dissertation, we delve into the exploration of Energy-Based Models (EBM), employing a case study focused on 3D modeling to illustrate the application of EBM in addressing real-world challenges. This investigation not only highlights the efficacy of EBM but also sheds light on the complexities associated with its training processes. Drawing on these insights, we have developed algorithms aimed at enhancing the training efficiency of EBM. Specifically, we introduce two innovative algorithms: CoopFlow and CDRL. CoopFlow enhances the generative capabilities of EBM, showcasing the benefits of a cooperative training approach. Meanwhile, CDRL bridges the divide between EBM and other generative frameworks, demonstrating its versatility across a variety of tasks. Below, we succinctly summarize the key findings and contributions of three pivotal chapters:

6.1.1 Likelihood-Based Generative Radiance Field with Latent Space Energy

In this chapter, we explore the application of EBM within the realm of 3D modeling, advancing the development of likelihood-based generative radiance field models for disentangled representation. We introduce the NeRF-LEBM framework, which integrates informative and trainable energy-based priors for the latent variables of object appearance and shape into a NeRF-based 2D image generator. Additionally, we present two maximum likelihood learning algorithms: one leveraging MCMC-based inference and the other utilizing amortized inference. The NeRF-LEBM framework is

formulated under two scenarios: one with known camera poses and the other with unknown camera poses. Through a series of tasks, we demonstrate NeRF-LEBM's capability to generate meaningful samples, infer 3D structures from 2D observations, accommodate incomplete 2D observations, and even learn from images with unspecified camera positions.

6.1.2 Cooperative Learning of Langevin Flow and Normalizing Flow Toward Energy-Based Model

This chapter advances the generative capabilities of EBM by exploring a novel paradigm that involves learning two distinct types of deep flow models within an energy-based framework for image representation and generation. The first model, the normalizing flow model, synthesizes examples by applying a sequence of invertible transformations to Gaussian noise. The second model, the Langevin flow, generates examples through a non-mixing, non-convergent short-run Markov Chain Monte Carlo (MCMC) process aimed at an EBM. We introduce the CoopFlow training algorithm, designed to concurrently train the short-run Langevin flow model and the normalizing flow, with the latter acting as an efficient initializer in a cooperative learning setup. Our experiments demonstrate that CoopFlow is an effective generative model, showcasing its utility in image generation, reconstruction, and interpolation, thereby underscoring its contribution to enhancing the generative performance of EBMs.

6.1.3 Learning Energy-Based Models by Cooperative Diffusion Recovery Likelihood

In this chapter, we propose CDRL, a groundbreaking energy-based generative learning framework that employs cooperative diffusion recovery likelihood to significantly improve EBMs' generative quality. We illustrate CDRL's prowess in compositional generation, out-of-distribution detection, image inpainting, and its compatibility with classifier-free guidance for conditional generation, highlighting its versatility and potential in advancing the field of generative modeling.

6.2 Limitations and Future Work

Within this thesis, we advance the field of Energy-Based Models (EBMs) through the introduction of the CoopFlow and CDRL algorithms. These innovations significantly enhance EBM's ability to model complex distributions, such as images, and reduce the time required for sampling. Despite these advancements, tackling very high-dimensional data, such as high-resolution images or videos, remains a formidable challenge. This necessitates the development of more computationally efficient methods. Employing pre-defined encoder-decoder models (RBL22) or devising MCMC-free algorithms (HNF19a; GNK20) might be promising directions for future research. Additionally, the diverse applications of EBMs warrant further exploration. As illustrated by our case study with NeRF-LEBM, venturing into varied applications demands specific domain knowledge, often requiring collaboration among individuals from different disciplines. Moreover, as a powerful generative model, EBM might also pose potential negative societal impacts, such as the creation of deepfakes, spread of misinformation, privacy infringements, and undermining of public trust. This underscores the imperative for implementing effective preventive measures in both academic research and the industry to mitigate these risks.

Bibliography

[AMG18] Hassan Abu Alhaija, Siva Karthik Mustikovela, Andreas Geiger, and Carsten Rother. "Geometric Image Synthesis." In *Proceedings of the 14th Asian Conference on Computer Vision (ACCV), Part VI*, pp. 85–100, Perth, Australia, 2018.

[AXL21] Dongsheng An, Jianwen Xie, and Ping Li. "Learning Deep Latent Variable Models by Short-Run MCMC Inference With Optimal Transport Correction." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15415–15424, 2021.

[AZG21] Michael Arbel, Liang Zhou, and Arthur Gretton. "Generalized energy based models." In *The 9th International Conference on Learning Representations (ICLR)*, 2021.

[BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis." In *The 7th International Conference on Learning Representations (ICLR)*, 2019.

[BGK18] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. "Geometry-Aware Learning of Maps for Camera Localization." In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2616–2625, Salt Lake City, UT, 2018.

[BYA13] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. "Generalized denoising auto-encoders as generative models." In *Advances in neural information processing systems (NIPS)*, 2013.

[BZ20] Adrian Barbu and Song-Chun Zhu. *Monte Carlo Methods*, volume 35. Springer, 2020.

[CBD19] Ricky TQ Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. "Residual flows for invertible generative modeling." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[CC24] Taoli Cheng and Aaron C Courville. "Versatile Energy-Based Probabilistic Models for High Energy Physics." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[CFG15] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, and Fisher Yu. "ShapeNet: An information-rich 3D model repository." *arXiv preprint arXiv:1512.03012*, 2015.

[CH05] Miguel Á. Carreira-Perpiñán and Geoffrey E. Hinton. "On Contrastive Divergence Learning." In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, Bridgetown, Barbados, 2005.

[CH23] Jiali Cui and Tian Han. "Learning Energy-based Model via Dual-MCMC Teaching." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[CMK21] Eric R. Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. "Pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5799–5809, virtual, 2021.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory, Second Edition*. Wiley, 2006.

[CWD18] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. "Generative adversarial networks: An overview." *IEEE signal processing magazine*, **35**(1):53–65, 2018.

[DDS09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database." In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[DDS23] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Grathwohl. "Reduce, Reuse, Recycle: Compositional Generation with Energy-Based Diffusion Models and MCMC." In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[DFC18] Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. "Hyperspherical Variational Auto-Encoders." In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 856–865, Monterey, CA, 2018.

[DKB15] Laurent Dinh, David Krueger, and Yoshua Bengio. "NICE: Non-linear Independent Components Estimation." In *Workshop Proceedings of the 3rd International Conference on Learning Representations (ICLR Workshop)*, San Diego, CA, 2015.

[DLM20] Yilun Du, Shuang Li, and Igor Mordatch. "Compositional visual generation with energy based models." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[DLT21a] Yilun Du, Shuang Li, Joshua B. Tenenbaum, and Igor Mordatch. "Improved Contrastive Divergence Training of Energy-Based Models." In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[DLT21b] Yilun Du, Shuang Li, Joshua B. Tenenbaum, and Igor Mordatch. "Improved Contrastive Divergence Training of Energy-Based Models." In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 2837–2848, Virtual Event, 2021.

[DM19] Yilun Du and Igor Mordatch. "Implicit generation and generalization in energy-based models." *arXiv preprint arXiv:1903.08689*, 2019.

[DN21] Prafulla Dhariwal and Alexander Quinn Nichol. "Diffusion Models Beat GANs on Image Synthesis." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[DPS18] Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. "Feature-wise transformations." *Distill*, **3**(7):e11, 2018.

[DRC17] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. "CARLA: An Open Urban Driving Simulator." In *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*, pp. 1–16, Mountain View, CA, 2017.

[DSB17] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP." In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[DW19] Bin Dai and David P. Wipf. "Diagnosing and Enhancing VAE Models." In *Proceeding of the 7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, 2019.

[Efr11] Bradley Efron. "Tweedie's formula and selection bias." *Journal of the American Statistical Association*, **106**(496):1602–1614, 2011.

[EJB18] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. "Neural scene representation and rendering." *Science*, **360**(6394):1204–1210, 2018.

[ERO21] Patrick Esser, Robin Rombach, and Bjorn Ommer. "Taming transformers for high-resolution image synthesis." In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[GAA17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. "Improved training of wasserstein gans." In *The 6th International Conference on Learning Representations (ICLR)*, 2017.

[GAD22] Pierre Glaser, Michael Arbel, Arnaud Doucet, and Arthur Gretton. "Maximum Likelihood Learning of Energy-Based Models for Simulation-Based Inference." *arXiv preprint arXiv:2210.14756*, 2022.

[GKH21a] Will Sussman Grathwohl, Jacob Jin Kelly, Milad Hashemi, Mohammad Norouzi, Kevin Swersky, and David Duvenaud. "No MCMC for me: Amortized sampling for fast and stable training of energy-based models." In *The 9th International Conference on Learning Representations (ICLR)*, 2021.

[GKH21b] Will Sussman Grathwohl, Jacob Jin Kelly, Milad Hashemi, Mohammad Norouzi, Kevin Swersky, and David Duvenaud. "No MCMC for me: Amortized sampling for fast and stable training of energy-based models." In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, Virtual Event, 2021.

[GLZ18] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. "Learning Generative ConvNets via Multi-Grid Modeling and Sampling." In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[GMJ23] Qiushan Guo, Chuofan Ma, Yi Jiang, Zehuan Yuan, Yizhou Yu, and Ping Luo. "EGC: Image Generation and Classification via a Diffusion Energy-Based Model." *arXiv preprint arXiv:2304.02012*, 2023.

[GNK20] Ruiqi Gao, Erik Nijkamp, Diederik P. Kingma, Zhen Xu, Andrew M Dai, and Ying Nian Wu. "Flow contrastive estimation of energy-based models." In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[GPM14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. "Generative Adversarial

Nets." In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2672–2680, Montreal, Canada, 2014.

[GPM20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial networks." *Communications of the ACM*, **63**(11):139–144, 2020.

[GSP21a] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P. Kingma. "Learning Energy-Based Models by Diffusion Recovery Likelihood." In *The 9th International Conference on Learning Representations (ICLR)*, 2021.

[GSP21b] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P. Kingma. "Learning Energy-Based Models by Diffusion Recovery Likelihood." In *Proceeding of the 9th International Conference on Learning Representations (ICLR)*, Virtual Event, 2021.

[GSV20] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael J. Black, and Bernhard Schölkopf. "From Variational to Deterministic Autoencoders." In *Proceeding of the 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.

[GWJ20] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. "Your classifier is secretly an energy based model and you should treat it like one." In *The 8th International Conference on Learning Representations (ICLR)*, 2020.

[HCS19] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. "Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design." In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 2722–2730, Long Beach, CA, 2019.

[Hin02a] Geoffrey E Hinton. "Training products of experts by minimizing contrastive divergence." *Neural computation*, **14**(8):1771–1800, 2002.

[Hin02b] Geoffrey E. Hinton. "Training Products of Experts by Minimizing Contrastive Divergence." *Neural Computation*, **14**(8):1771–1800, 2002.

[Hin12] Geoffrey E. Hinton. "A Practical Guide to Training Restricted Boltzmann Machines." In *Neural Networks: Tricks of the Trade - Second Edition*, pp. 599–619. Springer, 2012.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[HLZ17] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. "Alternating Back-Propagation for Generator Network." In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1976–1984, 2017.

[HMR19] Philipp Henzler, Niloy J. Mitra, and Tobias Ritschel. "Escaping Plato's Cave: 3D Shape From Adversarial Rendering." In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9983–9992, Seoul, Korea, 2019.

[HNF19a] Tian Han, Erik Nijkamp, Xiaolin Fang, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. "Divergence Triangle for Joint Training of Generator Model, Energy-Based Model, and Inferential Model." In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[HNF19b] Tian Han, Erik Nijkamp, Xiaolin Fang, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. "Divergence Triangle for Joint Training of Generator Model, Energy-Based Model, and Inferential Model." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8670–8679, Long Beach, CA, 2019.

[HNM22] Mitch Hill, Erik Nijkamp, Jonathan Mitchell, Bo Pang, and Song-Chun Zhu. "Learning Probabilistic Models from Generator Latent Spaces with Hat EBM." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[HNZ20] Tian Han, Erik Nijkamp, Linqi Zhou, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. "Joint training of variational auto-encoder and latent energy-based model." In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[HRU17a] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." In *Advances in Neural Information Processing Systems (NIPS)*, pp. 6626–6637, Long Beach, CA, 2017.

[HRU17b] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[HS22] Jonathan Ho and Tim Salimans. "Classifier-free diffusion guidance." *arXiv preprint arXiv:2207.12598*, 2022.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, 2016.

[KAA22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. "Elucidating the design space of diffusion-based generative models." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[KAH20] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. "Training generative adversarial networks with limited data." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[KB15a] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In

Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, 2015.

[KB15b] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, 2015.

[KB15c] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In *The 3rd International Conference on Learning Representations (ICLR)*, 2015.

[KB16] Taesup Kim and Yoshua Bengio. "Deep directed generative models with energy-based probability estimation." *arXiv preprint arXiv:1606.03439*, 2016.

[KBE20] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. "VideoFlow: A Conditional Flow-Based Model for Stochastic Video Generation." In *Proceeding of the 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.

[KCZ22] Lingkai Kong, Jiaming Cui, Yuchen Zhuang, Rui Feng, B. Aditya Prakash, and Chao Zhang. "End-to-end Stochastic Optimization with Energy-based Model." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[KD18] Diederik P. Kingma and Prafulla Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[KH84] James T. Kajiya and Brian Von Herzen. "Ray tracing volume densities." In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 165–174, Minneapolis, MN, 1984.

[KH09] Alex Krizhevsky and Geoffrey Hinton. "Learning multiple layers of features from tiny images." 2009.

[KOG19] Rithesh Kumar, Sherjil Ozair, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. "Maximum entropy generators for energy-based models." *arXiv preprint arXiv:1901.08508*, 2019.

[KSJ16] Diederik P. Kingma, Tim Salimans, Rafal Józefowicz, Xi Chen, Ilya Sutskever, and Max Welling. "Improving Variational Autoencoders with Inverse Autoregressive Flow." In *Advances in Neural Information Processing Systems (NIPS)*, pp. 4736–4744, Barcelona, Spain, 2016.

[KSP21] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. "Variational diffusion models." In *Advances in neural information processing systems (NeurIPS)*, 2021.

[KSS21] Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. "Soft Truncation: A Universal Training Technique of Score-based Diffusion Model for High Precision Score Estimation." In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[KSZ21] Adam R. Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Sona Mokrá, and Danilo Jimenez Rezende. "NeRF-VAE: A Geometry Aware 3D Scene Generative Model." In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 5742–5752, Virtual Event, 2021.

[KUH18] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. "Neural 3D Mesh Renderer." In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3907–3916, Salt Lake City, UT, 2018.

[KW14a] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes." In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.

[KW14b] Diederik P. Kingma and Max Welling. "Auto-encoding variational bayes." In *The 2nd International Conference on Learning Representations (ICLR)*, 2014.

[LCB22] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. "Flow matching for generative modeling." *arXiv preprint arXiv:2210.02747*, 2022.

[LCH06] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. "A tutorial on energy-based learning." *Predicting structured data*, **1**(0), 2006.

[LCL19] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. "Soft Rasterizer: A Differentiable Renderer for Image-Based 3D Reasoning." In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7707–7716, Seoul, Korea, 2019.

[LGL22] Xingchao Liu, Chengyue Gong, and Qiang Liu. "Flow straight and fast: Learning to generate and transfer data with rectified flow." *arXiv preprint arXiv:2209.03003*, 2022.

[LH19] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization." In *The 7th International Conference on Learning Representations (ICLR)*, 2019.

[LJP23] Hankook Lee, Jongheon Jeong, Sejun Park, and Jinwoo Shin. "Guiding Energy-based Models via Contrastive Latent Variables." In *The 11th International Conference on Learning Representations (ICLR)*, 2023.

[LL21] Eric Luhman and Troy Luhman. "Knowledge distillation in iterative generative models for improved sampling speed." *arXiv preprint arXiv:2101.02388*, 2021.

[LLW15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Attributes in the Wild." In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[LRL22] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. "Pseudo numerical methods for diffusion models on manifolds." In *The 10th International Conference on Learning Representations (ICLR)*, 2022.

[LWO20] Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li. "Energy-based Out-of-distribution Detection." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[LYO21] Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. "GraphEBM: Molecular Graph Generation with Energy-Based Models." In *ICLR workshop on Energy-Based Model*, 2021.

[LZB22] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. "DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[LZW16] Yang Lu, Song-Chun Zhu, and Ying Wu. "Learning FRAME models using CNN filters." In *Proceeding of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

[Mar75] Kantilal Varichand Mardia. "Statistics of directional data." *Journal of the Royal Statistical Society: Series B (Methodological)*, **37**(3):349–371, 1975.

[MKK18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. "Spectral normalization for generative adversarial networks." In *The 6th International Conference on Learning Representations (ICLR)*, 2018.

[MST22] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. "NeRF: representing scenes as neural radiance fields for view synthesis." *Commun. ACM*, **65**(1):99–106, 2022.

[ND21] Alexander Quinn Nichol and Prafulla Dhariwal. "Improved Denoising Diffusion Probabilistic Models." In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[Nea11] Radford M Neal. "MCMC using Hamiltonian dynamics." *Handbook of Markov Chain Monte Carlo*, **2**(11):2, 2011.

[NG21] Michael Niemeyer and Andreas Geiger. "GIRAFFE: Representing Scenes As Compositional Generative Neural Feature Fields." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11453–11464, virtual, 2021.

[NGS22] Erik Nijkamp, Ruiqi Gao, Pavel Sountsov, Srinivas Vasudevan, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. "Learning energy-based model with flow-based backbone by neural transport mcmc." In *The 9th International Conference on Learning Representations (ICLR)*, 2022.

[NHZ19] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. "Learning Non-Convergent Non-Persistent Short-Run MCMC Toward Energy-Based Model." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[NMJ19] K L Navaneet, Priyanka Mandikal, Varun Jampani, and R. Venkatesh Babu. "DIFFER: Moving Beyond 3D Reconstruction with Differentiable Feature Rendering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, pp. 18–24, Long Beach, CA, 2019.

[NPH20] Erik Nijkamp, Bo Pang, Tian Han, Linqi Zhou, Song-Chun Zhu, and Ying Nian Wu. "Learning Multi-layer Latent Variable Model via Variational Optimization of Short Run MCMC for Approximate Inference." In *European Conference on Computer Vision (ECCV)*, volume 12351, pp. 361–378, 2020.

[NWC11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. "Reading digits in natural images with unsupervised feature learning." *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[ODM18] Georg Ostrovski, Will Dabney, and Rémi Munos. "Autoregressive quantile networks

for generative modeling." In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[OMN19] Michael Oechsle, Lars M. Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. "Texture Fields: Learning Texture Representations in Function Space." In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4530–4539, Seoul, Korea, 2019.

[PHN20] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. "Learning latent space energy-based prior model." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[PLL21] Gaurav Parmar, Dacheng Li, Kwonjoon Lee, and Zhuowen Tu. "Dual Contradistinctive Generative Autoencoder." In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[PSV18] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. "FiLM: Visual Reasoning with a General Conditioning Layer." In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, pp. 3942–3951, New Orleans, LA, 2018.

[PVC19] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. "Waveglow: A Flow-based Generative Network for Speech Synthesis." In *Proceeding of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621, Brighton, United Kingdom, 2019.

[RBL22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-Resolution Image Synthesis with Latent Diffusion Models." In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[RDN22] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen.

"Hierarchical text-conditional image generation with clip latents." *arXiv preprint arXiv:2204.06125*, 2022.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[RM15] Danilo Jimenez Rezende and Shakir Mohamed. "Variational Inference with Normalizing Flows." In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 1530–1538, Lille, France, 2015.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434*, 2015.

[Rob92] Herbert E Robbins. *An empirical Bayes approach to statistics*. Springer, 1992.

[SCS22] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[SE19] Yang Song and Stefano Ermon. "Generative modeling by estimating gradients of the data distribution." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[SE20] Yang Song and Stefano Ermon. "Improved techniques for training score-based generative models." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[SGZ16] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved Techniques for Training GANs." In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[SH21] Tim Salimans and Jonathan Ho. "Should EBMs model the energy or the score?" In *ICLR workshop on Energy-Based Model*, 2021.

[SH22] Tim Salimans and Jonathan Ho. "Progressive Distillation for Fast Sampling of Diffusion Models." In *The 10th International Conference on Learning Representations (ICLR)*, 2022.

[SK16] Tim Salimans and Diederik P. Kingma. "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks." In *Advances in Neural Information Processing Systems 29 (NIPS)*, p. 901, Barcelona, Spain, 2016.

[SKC17] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications." In *The 5th International Conference on Learning Representations (ICLR)*, 2017.

[SLN20] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. "GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis." In *Advances in Neural Information Processing Systems (NeurIPS)*, virtual, 2020.

[SME21] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising Diffusion Implicit Models." In *The 9th International Conference on Learning Representations (ICLR)*, 2021.

[SSK21] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-Based Generative Modeling through Stochastic Differential Equations." In *The 9th International Conference on Learning Representations (ICLR)*, 2021.

[SWM15] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics." In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

[SZW19] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. "Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations." In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1119–1130, Vancouver, Canada, 2019.

[VK20a] Arash Vahdat and Jan Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder." In *Advances in Neural Information Processing Systems (NeurIPS)*, Virtual Event, 2020.

[VK20b] Arash Vahdat and Jan Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[WG16] Xiaolong Wang and Abhinav Gupta. "Generative Image Modeling Using Style and Structure Adversarial Networks." In *Proceedings of the 14th European Conference on Computer Vision (ECCV), Part IV*, pp. 318–335, Amsterdam, The Netherlands, 2016.

[XGZ19] Jianwen Xie, Ruiqi Gao, Zilong Zheng, Song-Chun Zhu, and Ying Nian Wu. "Learning Dynamic Generator Model by Alternating Back-Propagation through Time." In *The Thirty-Third AAAI Conference on Artificial Intelligence, (AAAI)*, pp. 5498–5507, 2019.

[XH22] Zhisheng Xiao and Tian Han. "Adaptive Multi-stage Density Ratio Estimation for Learning Latent Space Energy-based Model." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[XKK21] Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. "VAEBM: A symbiosis between variational autoencoders and energy-based models." In *The 9th International Conference on Learning Representations (ICLR)*, 2021.

[XKV22] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. "Tackling the Generative Learning Trilemma with Denoising Diffusion GANs." In *The 9th International Conference on Learning Representations (ICLR)*, 2022.

[XLG18] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. "Cooperative training of descriptor and generator networks." *IEEE Transactions on Pattern Analysis and Machine intelligence (TPAMI)*, **42**(1):27–45, 2018.

[XLG20] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. "Cooperative Training of Descriptor and Generator Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **42**(1):27–45, 2020.

[XLZ16a] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. "A Theory of Generative ConvNet." In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, 2016.

[XLZ16b] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. "A Theory of Generative ConvNet." In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, pp. 2635–2644, New York City, NY, 2016.

[XXZ21] Jianwen Xie, Yifei Xu, Zilong Zheng, Song-Chun Zhu, and Ying Nian Wu. "Generative PointNet: Deep Energy-Based Learning on Unordered Point Sets for 3D Generation, Reconstruction and Classification." In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[XZG18] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. "Learning Descriptor Networks for 3D Shape Synthesis and Analysis." In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognitio (CVPR)*, 2018.

[XZL21] Jianwen Xie, Zilong Zheng, and Ping Li. "Learning Energy-Based Model with

Variational Auto-Encoder as Amortized Sampler." In *Proceeding of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[XZL22] Jianwen Xie, Yaxuan Zhu, Jun Li, and Ping Li. "A Tale of Two Flows: Cooperative Learning of Langevin Flow and Normalizing Flow Toward Energy-Based Model." In *The 10th International Conference on Learning Representations (ICLR)*, 2022.

[XZW21] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. "Learning Energy-Based Spatial-Temporal Generative ConvNets for Dynamic Patterns." *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **43**(2):516–531, 2021.

[YHH19] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge J. Belongie, and Bharath Hariharan. "PointFlow: 3D Point Cloud Generation With Continuous Normalizing Flows." In *Proceeding of the IEEE International Conference on Computer Vision (ICCV)*, pp. 4540–4549, Seoul, Korea, 2019.

[YLR22] Xuwang Yin, Shiying Li, and Gustavo K Rohde. "Learning energy-based models with adversarial training." In *The European Conference on Computer Vision (ECCV)*, 2022.

[You99] Laurent Younes. "On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates." *Stochastics: An International Journal of Probability and Stochastic Processes*, **65**(3-4):177–228, 1999.

[YZX23] Peiyu Yu, Yaxuan Zhu, Sirui Xie, Xiaojian Shawn Ma, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. "Learning Energy-Based Prior Model with Diffusion-Amortized MCMC." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[ZLL20] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. "Differentiable Augmentation for Data-Efficient GAN Training." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[ZWM98a] Song-Chun Zhu, Ying Nian Wu, and David Mumford. "Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling." *International Journal of Computer Vision (IJCV)*, **27**:107–126, 1998.

[ZWM98b] Song-Chun Zhu, Ying Nian Wu, and David Mumford. "Filters, Random Fields and Maximum Entropy (FRAME): Towards a Unified Theory for Texture Modeling." *International Journal of Computer Vision (IJCV)*, **27**(2):107–126, 1998.

[ZXB23] Jing Zhang, Jianwen Xie, Nick Barnes, and Ping Li. "An Energy-Based Prior for Generative Saliency." *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **45**(11):13100–13116, 2023.

[ZXL21] Yang Zhao, Jianwen Xie, and Ping Li. "Learning Energy-Based Generative Models via Coarse-to-Fine Expanding and Sampling." In *The 9th International Conference on Learning Representations (ICLR)*, 2021.

[ZXL23] Yaxuan Zhu, Jianwen Xie, and Ping Li. "Likelihood-Based Generative Radiance Field with Latent Space Energy-Based Model for 3D-Aware Disentangled Image Representation." In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2023.

[ZZZ18] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. "Visual Object Networks: Image Generation with Disentangled 3D Representations." In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pp. 118–129, Montréal, Canada, 2018.