A TALE OF TWO FLOWS: COOPERATIVE LEARNING OF LANGEVIN FLOW AND NORMALIZING FLOW TOWARD ENERGY-BASED MODEL

Jianwen Xie, Yaxuan Zhu, Jun Li, Ping Li

Cognitive Computing Lab
Baidu Research
10900 NE 8th St. Bellevue, WA 98004, USA
{jianwen.kenny, yaxuanzhu.alvin, junli.sh04, pingli98}@gmail.com

ABSTRACT

This paper studies the cooperative learning of two generative flow models, in which the two models are iteratively updated based on the jointly synthesized examples. The first flow model is a normalizing flow that transforms an initial simple density into a target density by applying a sequence of invertible transformations. The second flow model is a Langevin flow that runs finite steps of gradient-based MCMC toward an energy-based model. We start from proposing a generative framework that trains an energy-based model with a normalizing flow as an amortized sampler to initialize the MCMC chains of the energy-based model. In each learning iteration, we generate synthesized examples by using a normalizing flow initialization followed by a short-run Langevin flow revision toward the current energy-based model. 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. We provide an understating of the CoopFlow algorithm by information geometry and show that it is a valid generator as it converges to a moment matching estimator. We demonstrate that the trained CoopFlow is capable of synthesizing realistic images, reconstructing images, and interpolating between images.

1 INTRODUCTION AND MOTIVATION

Normalizing flows (Dinh et al., 2015; 2017; Kingma & Dhariwal, 2018) are a family of generative models that construct a complex distribution by transforming a simple probability density, such as Gaussian distribution, through a sequence of invertible and differentiable mappings. Due to the tractability of the exact log-likelihood and the efficiency of the inference and synthesis, normalizing flows have gained popularity in density estimation (Kingma & Dhariwal, 2018; Ho et al., 2019; Yang et al., 2019; Prenger et al., 2019; Kumar et al., 2020) and variational inference (Rezende & Mohamed, 2015; Kingma et al., 2016). However, for the sake of ensuring the favorable property of closed-form density evaluation, the normalizing flows typically require special designs of the sequence of transformations, which, in general, constrain the expressive power of the models.

Energy-based models (EBMs) (Zhu et al., 1998; LeCun et al., 2006; Hinton, 2012) define an unnormalized probability density function of data, which is the exponential of the negative energy function. The energy function is directly defined on data domain, and assigns each input configuration with a scalar energy value, with lower energy values indicating more likely configurations. Recently, with the energy function parameterized by a modern deep network such as a ConvNet, the ConvNet-EBMs (Xie et al., 2016) have gained unprecedented success in modeling large-scale data sets (Gao et al., 2018; Nijkamp et al., 2019; Du & Mordatch, 2019; Grathwohl et al., 2020; Gao

et al., 2021; Zhao et al., 2021; Zheng et al., 2021) and exhibited stunning performance in synthesizing different modalities of data, e.g., videos (Xie et al., 2017; 2021c), volumetric shapes (Xie et al., 2018b; 2020b), point clouds (Xie et al., 2021a) and molecules (Du et al., 2020). The parameters of the energy function can be trained by maximum likelihood. However, due to the intractable integral in computing the normalizing constant, the evaluation of the gradient of the log-likelihood typically requires Markov chain Monte Carlo (MCMC) sampling (Barbu & Zhu, 2020), e.g., Langevin dynamics (Neal, 2011), to generate samples from the current model. However, the Langevin sampling on a highly multi-modal energy function, due to the use of deep network parameterization, is generally not mixing. When sampling from a density with a multi-modal landscape, the Langevin dynamics, which follows the gradient information, is apt to get trapped by local modes of the density and is unlikely to jump out to explore other isolated modes. Relying on non-mixing MCMC samples, the estimated gradient of the likelihood can be biased, and the resulting learned energy-based model may become an invalid model, which is unable to approximate the data distribution as expected.

Recently, Nijkamp et al. (2019) propose to train an EBM with a short-run non-convergent Langevin dynamics, and show that even though the energy function is invalid, the short-run MCMC can be treated as a valid flow model that generates realistic examples. This not only provides an explanation of why EBMs with non-convergent MCMC are still capable of synthesizing realistic examples, but also suggests a more practical computationally-affordable way to learn useful generative models under the existing energy-based learning frameworks. Although EBMs have been widely applied to different domains, learning short-run MCMC in the context of EBM is still underexplored.

In this paper, we will accept the fact that MCMC sampling is not mixing in practice, and decide to give up the goal of training a valid EBM of 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 because it can be considered as a noise-injected residual network. Even though implementing a short-run Langevin flow is surprisingly simple, which is nothing but a design of a bottom-up ConvNet for the energy function, it might still require a sufficiently large number of Langevin steps (each Langevin step consists of one step of gradient descent plus a diffusion step) to construct an effective Langevin flow, so that it can be expressive enough to represent and synthesize the data. Motivated by reducing the number of Langevin steps in the Langevin flow for computational efficiency, we propose the CoopFlow model that trains a Langevin flow jointly with a normalizing flow in a cooperative learning scheme, in which the normalizing flow model learns to serve as a rapid sampler to initialize the Langevin flow so that the Langevin flow can be shorter, while the Langevin flow teaches the normalizing flow by short-run MCMC transition toward the EBM so that the normalizing flow can accumulate the temporal difference in the transition to provide better initial samples. Besides, the motivation of using the normalizing flow instead of a generic generator, is because the cooperative learning framework need to incorporate the MLE algorithm of an EBM and the MLE algorithm of a generator. The MLE of a normalizing flow generator will be much more tractable than the MLE of any other generic generator. The latter might resort to either MCMC-based inference to evaluate the posterior distribution or another encoder network for variational inference. In the proposed CoopFlow, the Langevin flow can overcome expressivity limitation of the normalizing flow due to the invertibility constraint. Such a benefit also motivates us to study the CoopFlow framework. To be specific, we understand the dynamics of cooperative learning with short-run non-mixing MCMC by information geometry. We provide justifications that the CoopFlow trained in the context of EBM with non-mixing MCMC is a valid generator, because it converges to a moment matching estimator. Experiments, including image generation, image reconstruction, and latent space interpolation, are conducted to support our justification.

2 Contributions and Related Work

Our paper studies amortized sampling for training a short-run non-mixing Langevin sampler toward an EBM. We propose a novel framework, the CoopFlow, which cooperatively trains a short-run Langevin flow as a valid generator and a normalizing flow as an amortized sampler for image representation. We provide both theoretical and empirical justifications for the proposed CoopFlow algorithm. The following are some closely related work. We will point out the differences between our work and the prior arts to further highlight the contributions and novelty of our paper.

Learning short-run MCMC as a generator. Recently, Nijkamp et al. (2019) propose to learn an EBM with short-run non-convergent MCMC that samples from the model, and they eventually

keep the short-run MCMC as a valid generator and discard the biased EBM. Nijkamp et al. (2020b) use short-run MCMC to sample the latent space of a top-down generative model in a variational learning framework. An et al. (2021a) proposes to correct the bias of short-run MCMC inference by optimal transport in training latent variable model. Pang et al. (2020) adopts short-run MCMC to sample from both the EBM prior and the posterior of the latent variables. Our work studies learning a normalizing flow to amortize the sampling cost of a short-run non-mixing MCMC sampler (i.e., Langevin flow) in data space, which makes a further step forward in this underexplored theme.

Cooperative learning with MCMC teaching. Our learning algorithm is inspired by the cooperative learning or the CoopNets (Xie et al., 2018a; 2020a), in which an EBM and a top-down generator (Han et al., 2017) are jointly trained by jump-starting their maximum learning algorithms. Xie et al. (2021b) replaces the generator in the original CoopNets by a VAE (Kingma & Welling, 2014) for efficient inference. The proposed CoopFlow algorithm is different from the above prior work in the following two aspects. (i) In the idealized long-run mixing MCMC scenario, our algorithm is a cooperative learning framework that trains an unbiased EBM and a normalizing flow via MCMC teaching, where the updating of the normalizing flow with a tractable density is more efficient and less biased than updating a generic generator via variational inference as in Xie et al. (2021b) or MCMC-based inference as in Xie et al. (2020a). (ii) Our paper has a novel emphasis on studying cooperative learning with short-run non-mixing MCMC, which is more practical and common in reality. Our algorithm actually trains a short-run Langevin flow and a normalizing flow together toward a biased EBM for image generation. We use information geometry to understand the learning dynamics and show that the learned two-flow generator (i.e., the CoopFlow) is a valid generative model, even though the learned EBM is biased.

Joint training of EBM and normalizing flow. There are other two works studying EBM and normalizing flow together. To avoid MCMC, Gao et al. (2020) propose to train an EBM using noise contrastive estimation (Gutmann & Hyvärinen, 2010), where the noise distribution is a normalization flow. Nijkamp et al. (2020a) propose to learn an EBM as an exponential tilting of a pretrained normalizing flow, so that neural transport MCMC sampling in the latent space of the normalizing flow can mix well. Our paper proposes to train an EBM and a normalizing flow via short-run MCMC teaching. More specially, we focus on short-run non-mixing MCMC and treat it as a valid flow-like model (i.e., short-run Langevin flow) that is guided by the EBM. Disregarding the biased EBM, the resulting valid generator is the combination of the short-run Langevin flow and the normalizing flow, where the latter serves as a rapid initializer of the former. The form of this two-flow generator shares a similar fashion with the Stochastic Normalizing Flow (Wu et al., 2020) that consists of a sequence of deterministic invertible transformations and stochastic sampling blocks.

3 Two Flow Models

3.1 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:

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

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.

Maximum likelihood learning. Suppose we observe unlabeled training examples $\{x_i, i = 1, ..., n\}$ from unknown data distribution $p_{\text{data}}(x)$, the energy-based model in Eq. (1) can be trained from $\{x_i\}$ by Markov chain Monte Carlo (MCMC)-based maximum likelihood estimation, in which MCMC samples are drawn from the model $p_\theta(x)$ to approximate the gradient of the log-likelihood function for updating the model parameters θ. Specifically, the log-likelihood is $L(\theta) = \frac{1}{n}\sum_{i=1}^n \log p_\theta(x_i)$. For a large n, maximizing $L(\theta)$ is equivalent to minimizing the Kullback-Leibler (KL) divergence $\mathbb{D}_{\text{KL}}(p_{\text{data}}||p_\theta)$. The learning gradient is given by

$$L'(\theta) = \mathbb{E}_{p_{\text{data}}}[\nabla_\theta f_\theta(x)] - \mathbb{E}_{p_\theta}[\nabla_\theta f_\theta(x)] \approx \frac{1}{n}\sum_{i=1}^n \nabla_\theta f_\theta(x_i) - \frac{1}{n}\sum_{i=1}^n \nabla_\theta f_\theta(\tilde{x}_i), \tag{2}$$

where the expectations are approximated by averaging over the observed examples $\{x_i\}$ and the synthesized examples $\{\tilde{x}_i\}$ generated from the current model $p_\theta(x)$, respectively.

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

$$x_{t+1} = x_t + \frac{\delta^2}{2}\nabla_x f_\theta(x_t) + \delta\epsilon_t; \quad x_0 \sim p_0(x), \quad \epsilon_t \sim \mathcal{N}(0, I_D), \quad t = 1, \cdots, T, \tag{3}$$

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 the 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 the 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 update starting from $x_0 \sim p_0(x)$. Due to fixed initial distribution $p_0(x)$ and fixed K and δ, the distribution $\tilde{p}_\theta(x)$ is well defined, which can be implicitly expressed by

$$\tilde{p}_\theta(x) = (\mathcal{K}_\theta p_0)(x) = \int p_0(z)\mathcal{K}_\theta(x|z)dz, \tag{4}$$

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 s, 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, $x_{t+1} = x_t + \delta^2/2\nabla_x f_\theta(x_t)$, 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 (Cover & Thomas, 2006), $\mathbb{D}_{\text{KL}}(\tilde{p}_\theta(x)||p_\theta(x))$ decreases to zero monotonically as $T \to \infty$.

3.2 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

$$x = g_\alpha(z); z \sim q_0(z), \tag{5}$$

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 variable z that follows a simple distribution q_0 into a flexible distribution. Under the change-of-variables law, the resulting random variable $x = g_\alpha(z)$ 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 $z \xleftrightarrow{g_1} h_1 \xleftrightarrow{g_2} h_2 \cdots \xleftrightarrow{g_L} x$, 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

$$\log q_\alpha(x) = \log q_0(z) + \sum_{l=1}^{L} \log \left|\det\left(\frac{\partial h_{l-1}}{\partial h_l}\right)\right|. \tag{6}$$

With some smart designs of the sequence of transformations $g_\alpha = \{g_l, l = 1, ..., L\}$, the log-determinant in Eq. (6) can be easily computed, then the flow-based model can be trained by maximizing the data log-likelihood $L(\alpha) = \sum_i^n \log q_\alpha(x_i)$.

4 CoopFlow: Cooperative Training of Two Flows

4.1 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 treats 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 Langevin flow and 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 as the synthesized examples that are generated by the CoopFlow models. 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. Algorithm 1 presents a description of the proposed CoopFlow algorithm. The advantage of this training scheme is that we only need to minimally modify the existing code for the MLE training of EBM p_θ and normalizing flow q_α. The density of the CoopFlow $\pi_{(\theta,\alpha)}(x)$ is well defined, which can be implicitly expressed by

$$\pi_{(\theta,\alpha)}(x) = (\mathcal{K}_\theta q_\alpha)(x) = \int q_\alpha(x')\mathcal{K}_\theta(x|x')dx'. \tag{7}$$

If we increase the length of Langevin flow, $\pi_{(\theta,\alpha)}$ will converge to EBM $p_\theta(x)$. The network $f_\theta(x)$ in Langevin flow is scalar valued and is of free form, whereas the network $g_\alpha(x)$ in normalizing flow has high-dimensional output and is of a severely constrained form. Thus Langevin flow can potentially provide a tighter fit to $p_{\text{data}}(x)$ than normalizing flow. On the flip side, sampling from Langevin flow requires MCMC, whereas normalizing flow can synthesize examples directly via ancestral sampling. It is thus desirable to train these two flows, where normalizing flow serves as an approximate sampler to amortize the sampling cost of Langevin flow. Meanwhile the normalizing flow can be updated by a temporal difference MCMC teaching scheme provided by Langevin flow.

Algorithm 1 CoopFlow Algorithm

Input: (1) Observed images $\{x_i\}_i^n$; (2) Numbers of Langevin steps T; (4) Langevin step size δ; (5) Learning rate for Langevin flow η_θ; (6) Learning rates for normalizing flow η_α; 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 $\{z_i\}_{i=1}^m \sim p_0(z)$,
5: Start from z_i, generate $\hat{x}_i = g_\alpha(z_i)$ via normalizing flow.
6: Start from $\hat{x}_i$, run a T-step Langevin flow to obtain $\tilde{x}_i$ by following Eq. (3).
7: Given $\{\tilde{x}_i\}$, update α by maximizing $\sum_i \log q_\alpha(\tilde{x}_i)$ in Eq. (6).
8: Given $\{x_i\}$ and $\{\tilde{x}_i\}$, update θ by following the gradient $\nabla\theta$ in Eq. (2).
9: **until** converged

4.2 Understanding the Learned Two Flows

Convergence equations. In the traditional contrastive divergence (CD) 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 $\mathcal{K}$, which is specified by the Langevin flow, from the data distribution. In the proposed 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

$$\theta^* = \arg\min_\theta \mathbb{D}_{\text{KL}}(p_{\text{data}}||p_\theta) - \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^*} q_{\alpha^*}||p_\theta), \tag{8}$$

$$\alpha^* = \arg\min_\alpha \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^*} q_{\alpha^*}||q_\alpha). \tag{9}$$

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. (9) 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. (8) 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.

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 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 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, 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, $\text{KL}(\pi^* || p_{\theta^*})$ 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.

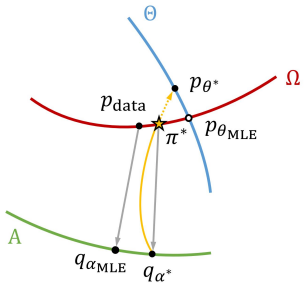


Figure 1: Illustration of convergence of CoopFlow.

The EBM with MLE $p_{\theta_{\text{MLE}}}$ is the intersection between Θ and Ω. It is the projection of p_{data} onto Θ because $\theta_{\text{MLE}} = \arg\min_\theta \mathbb{D}_{\text{KL}}(p_{\text{data}} || p_\theta)$. The normalizing flow with MLE $q_{\alpha_{\text{MLE}}}$ 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 Langevin flow is not mixing, $\theta^* \neq \theta_{\text{MLE}}$ and $\alpha^* \neq \alpha_{\text{MLE}}$. Specifically, 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. Note that Ω, Θ and A, is high-dimensional manifold instead of 1D curve as depicted in the Figure, 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 of the learning algorithm, there is an infinite looping, i.e., $\mathcal{K}_{\theta^*} q_{\alpha^*}$ (i.e., π^*) lifts q_{α^*} off the ground A, and the projection drops $\mathcal{K}_{\theta^*} q_{\alpha^*}$ back to q_{α^*}.

5 Experiments

We showcase experiment results on various tasks. We start from a toy example to illustrate the basic idea of the CoopFlow in Section 5.1. We show image generation results in Section 5.2. Section 5.3 demonstrates the learned CoopFlow is useful for image reconstruction and inpainting, while Section 5.4 shows that the learned latent space is meaningful so that it can be used for interpolation.

5.1 Toy Example Study

We first demonstrate our idea using a two-dimensional toy example where data lies on a spiral. We train 3 CoopFlow models with different lengths of Langevin flows. As shown in Figure 2, the right-

most box shows the results obtained with 10,000 Langevin steps. We can see in this situation, both the normalizing flow q_α and the EBM model p_θ can capture the underlying distribution perfectly. This validates that, with a sufficiently long Langevin flow, the CoopFlow algorithm can learn both a valid EBM and a valid normalizing flow. The leftmost green box represents the model trains with 100-step Langevin updates. This is a more typical non-convergent MCMC setting. In this case, neither q_α nor p_θ are perfect. Instead, they need to work together and the Langevin updates process over EBM model actually works as an extra generator that modifies the initial proposal by the normalizing flow model. We can see that the samples from q_α are not perfect, but after the modification, the samples from $\pi_{(\theta,\alpha)}$ follow the underlying distribution well. The middle green box shows the results obtained with 500-step Langevin steps. It is in the middle of short-run and long-run. p_θ becomes better but still not perfect. Using the toy example here, we want to make the points that short-run non-convergent MCMC situation might still exist even when we have a generator for initial proposal and using relatively long MCMC chain. Besides, if we use the MCMC on EBM as a special generator, i.e. Langevin Flow, then together with the normalizing flow, we can get pretty good samples even with few MCMC updates.

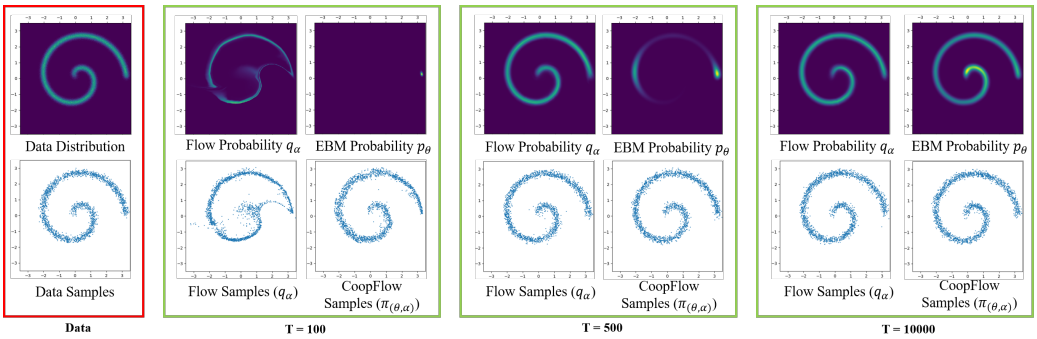


Figure 2: Toy example of using CoopFlow on two-dimensional data. We show the underlying data distribution in the red box and results of models trained with different Langevin steps in the three green boxes. In each green box, we show the final fitted probability of normalizing flow and EBM in the first row. Then in the second row, we show the samples proposed by normalizing flow and the samples modified by Langevin Flow based on normalizing flow's proposal.

5.2 Image Generation

We test our model on 3 image datasets for image synthesis. (i) CIFAR-10 (Krizhevsky & Hinton, 2009) is a dataset containing 50k training images and 10k testing images in 10 classes; (ii) SVHN (Netzer et al., 2011) is a dataset containing over 70k training images and over 20k testing images of house numbers; (iii) CelebA (Liu et al., 2015) is a celebrity facial dataset containing over 200k imagess. We downsample all the images to the resolution of 32×32. For our model, we show results of three different settings. CoopFlow denotes the setting where we train a normalizing flow and a Langevin flow together from scratch and use 30 Langevin steps. CoopFlow(long) denotes the setting that we increase the number of Langevin updates to 200. In CoopFlow(Pre) setting, we first pretrained a normalizing flow from observed data, and then we train the CoopFlow with the parameters of the normalizing flow being initialized by the pretrained one. We use a 30-step long Langevin flow in this setting. For all the three settings, we slightly increase the step size of Langevin update at the test stage for better performance. For more training details, please refer to Appendix.

We show both the qualitative results in Figure 3 and quantitative results in Table 1. To calculate FID Heusel et al. (2017) scores, we generate 50,000 samples on each dataset. From these results, we can see that our models outperform most of the baseline algorithms. We get lower FID scores comparing to the purely normalizing flow models and the prior works that try to jointly train a normalizing flow with an EBM, e.g., Gao et al. (2020); Nijkamp et al. (2020a). We also achieve comparable results with the SOTA EBMs. Among our different settings, we can see using a longer Langevin flow consistently improves the performance. And using a pretrained normalizing flow can provide the lowest FID scores among all the three settings. More results can be found in Appendix.

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

Figure 3: Generated examples (32 × 32 pixels) by CoopFlow models trained from CIFAR-10, SVHN and Celeba datasets respectively. Samples are obtained from the setting of CoopFlow(pre).

Approach	Models	FID ↓
VAE	VAE (Kingma & Welling, 2014)	78.41
Autoregressive	PixelCNN (Salimans et al., 2017)	65.93
	PixelIQN (Ostrovski et al., 2018)	49.46
GAN	WGAN-GP(Gulrajani et al., 2017)	36.40
	SN-GAN (Miyato et al., 2018)	21.70
	StyleGAN2-ADA (Karras et al., 2020)	2.92
Score Based	NCSN (Song & Ermon, 2019)	25.32
	NCSN-v2 (Song & Ermon, 2020)	31.75
	NCSN++ (Song et al., 2021)	2.20
Flow	Glow (Kingma & Dhariwal, 2018)	45.99
	Residual Flow (Chen et al., 2019)	46.37
EBM	LP-EBM (Pang et al., 2020)	70.15
	EBM-SR (Nijkamp et al., 2019)	44.50
	EBM-IG (Du & Mordatch, 2019)	38.20
	CoopVAEBM (Xie et al., 2021b)	36.20
	CoopNets (Xie et al., 2020a)	33.61
	Divergence Triangle (Han et al., 2020)	30.10
	VARA (Grathwohl et al., 2021)	27.50
	EBM-CD (Du et al., 2021)	25.10
	GEBM (Arbel et al., 2021)	19.31
	CF-EBM Zhao et al. (2021)	16.71
	VAEBM (Xiao et al., 2021)	12.16
	EBM-Diffusion (Gao et al., 2021)	9.60
Flow+EBM	NT-EBM (Nijkamp et al., 2020a)	78.12
	EBM-FCE (Gao et al., 2020)	37.30
Ours	CoopFlow	21.16
	CoopFlow(Long)	18.89
	CoopFlow(Pre)	15.80

(a) CIFAR 10

Models	FID ↓
ABP (Han et al., 2017)	49.71
ABP-SRI (Nijkamp et al., 2020b)	35.23
ABP-OT (An et al., 2021b)	19.48
VAE (Kingma & Welling, 2014)	46.78
2sVAE (Dai & Wipf, 2019)	42.81
RAE (Ghosh et al., 2020)	40.02
Glow (Kingma & Dhariwal, 2018)	41.70
DCGAN (Radford et al., 2016)	21.40
NT-EBM (Nijkamp et al., 2020a)	48.01
LP-EBM (Pang et al., 2020)	29.44
EBM-FCE (Gao et al., 2020)	20.19
CoopFlow	18.11
CoopFlow(Long)	16.97
CoopFlow(Pre)	15.32

(b) SVHN

Models	FID ↓
ABP (Han et al., 2017)	51.50
ABP-SRI (Nijkamp et al., 2020b)	36.84
VAE (Kingma & Welling, 2014)	38.76
Glow (Kingma & Dhariwal, 2018)	23.32
DCGAN (Radford et al., 2016)	12.50
EBM-FCE (Gao et al., 2020)	12.21
GEBM (Arbel et al., 2021)	5.21
CoopFlow	6.44
CoopFlow(Long)	4.90
CoopFlow(Pre)	4.15

(c) Celeba

Table 1: FID scores on CIFAR-10, SVHN and Celeba datasets. Images are resized to 32×32 pixels.

5.3 Image Reconstruction and Inpainting

We show that the learned CoopFlow model is able to reconstruct observed images. We may consider the CoopFlow model $\pi_{(\theta,\alpha)}(x)$ to be a latent variable generative model: $z \sim p_0(z); \hat{x} = g_\alpha(z); x = F_\theta(\hat{x}, e)$, 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 p_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 $z \sim p_0(z)$ 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 results are

shown in Figure 4. We do experiment on CIFAR10 test set. The right column displays the original images x that need to be reconstructed. The left column and the middle column display $\hat{x}^*$ and $F_\theta(\hat{x}^*)$, respectively. We can see that our model can successfully reconstruct the observed images, which verifies that the CoopFlow with a non-mixing MCMC is a valid latent variable generator.

We further show that our model is also capable of doing image inpainting. Similar to image reconstruction, given a masked observation x_{mask} 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 experiment on Celeba training set. In Figure 5, each row shows one example of inpainting with a different initialization of $\hat{x}$ provided by the normalizing flow and the first 17 columns display the inpainting results at a different optimization iteration. The last two columns show the masked images and the original images respectively. We can see that our model can reconstruct the unmasked area faithfully and simultaneously fill in the blank areas in the input images. The recovered images are meaningful and given different initializations, our model can inpaint diversified patterns.

Figure 4: Reconstruction of images

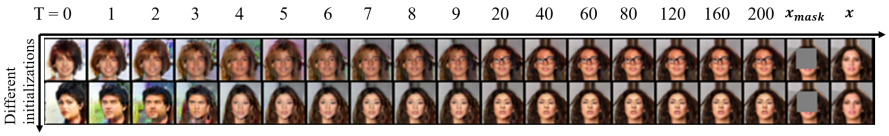


Figure 5: Image inpainting on Celeba dataset (32×32). Each row represents one different initialization. The last two columns display the masked and original images respectively. From the 1st column to the 17th column, we show the inpainted images at different optimization iterations.

5.4 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 5.3. We then infer z by the inversion of the normalizing flow $z^* = g_\alpha^{-1}(\hat{x}^*)$. The results of interpolation in latent space experiments on Celeba (training set) are shown in Figure 6. Comparing the first and second columns to the left (or to the right), We can see that our reconstructions of the original images are good. Further, our interpolations are smooth and the generated images are meaningful.

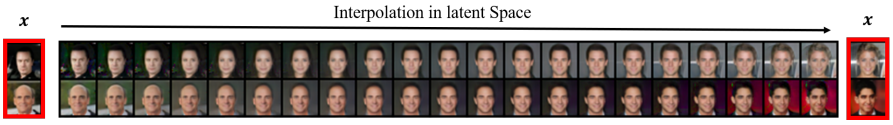


Figure 6: Image interpolation results on Celeba dataset (32×32). The leftmost and rightmost columns represent the images we observed. The columns in the middle represent the interploation results between the inferred latent vectors of the observed images.

6 Conclusion

This paper studies an interesting problem of learning two types of deep flow models in the context of energy-based framework for image representation and generation. One is the normalizing flow model that generates synthesized examples by transforming Gaussian noise examples through a sequence of invertible transformations, while the other is the Langevin flow that generates synthesized examples by running a non-mixing, non-convergent short-run MCMC toward an EBM. We propose the CoopFlow algorithm to train the short-run Langevin flow model jointly with the normalizing flow serving as a rapid initializer in a cooperative manner. The experiments show that the CoopFlow is a valid generative model that can be useful for image generation, reconstruction, and interpolation.

REFERENCES

Dongsheng An, Jianwen Xie, and Ping Li. Learning deep latent variable models by short-run MCMC inference with optimal transport correction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15415–15424, Virtual Event, 2021a.

Dongsheng An, Jianwen Xie, and Ping Li. Learning deep latent variable models by short-run MCMC inference with optimal transport correction. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15415–15424, Virtual Event, 2021b.

Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, Virtual Event, 2021.

Adrian Barbu and Song-Chun Zhu. *Monte Carlo Methods*. Springer Nature, 2020.

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.

Tian Qi Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 9913–9923, Vancouver, Canada, 2019.

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

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.

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.

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.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3603–3613, Vancouver, Canada, 2019.

Yilun Du, Joshua Meier, Jerry Ma, Rob Fergus, and Alexander Rives. Energy-based models for atomic-resolution protein conformations. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.

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.

Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning generative convnets via multi-grid modeling and sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9155–9164, 2018.

Ruiqi Gao, Erik Nijkamp, Diederik P. Kingma, Zhen Xu, Andrew M. Dai, and Ying Nian Wu. Flow contrastive estimation of energy-based models. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7515–7525, Seattle, WA, 2020.

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.

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.

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 *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.

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.

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 5767–5777, Long Beach, CA, 2017.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 297–304, Chia Laguna Resort, Sardinia, Italy, 2010.

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, San Francisco, CA, 2017.

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.

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 *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7975–7984, Seattle, WA, 2020.

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.

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

Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pp. 599–619. Springer, 2012.

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.

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)*, Virtual Event, 2020.

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.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 10236–10245, Montréal, Canada, 2018.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.

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.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

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.

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

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the 2015 International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *Proceeding of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

Radford M Neal. MCMC using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2 (11):2, 2011.

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

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 (NeurIPS)*, pp. 5233–5243, Vancouver, Canada, 2019.

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. *arXiv preprint arXiv:2006.06897*, 2020a.

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 *Proceedings of the 16th European Conference on Computer Vision (ECCV, Part VI)*, pp. 361–378, Glasgow, UK, 2020b.

Georg Ostrovski, Will Dabney, and Rémi Munos. Autoregressive quantile networks for generative modeling. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 3933–3942, Stockholmsmässan, Stockholm, Sweden, 2018.

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)*, Virtual Event, 2020.

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.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

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.

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)*, pp. 901, Barcelona, Spain, 2016.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *Proceeding of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 11895–11907, Vancouver, Canada, 2019.

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

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Proceeding of the 9th International Conference on Learning Representations (ICLR)*, Virtual Event, 2021.

Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems (NeurIPS)*, Virtual Event, 2020.

Hao Wu, Jonas Köhler, and Frank Noé. Stochastic normalizing flows. In *Advances in Neural Information Processing (NeurIPS)*, Virtual Event, 2020.

Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. VAEBM: A symbiosis between variational autoencoders and energy-based models. In *Proceeding of the 9th International Conference on Learning Representations (ICLR)*, Virtual Event, 2021.

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.

Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative convnet. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1061–1069, Honolulu, HI, 2017.

Jianwen Xie, Yang Lu, Ruiqi Gao, and Ying Nian Wu. Cooperative learning of energy-based model and latent variable model via MCMC teaching. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, (AAAI)*, pp. 4292–4301, New Orleans, Louisiana, USA, 2018a.

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 Recognition (CVPR)*, pp. 8629–8638, Salt Lake City, UT, 2018b.

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, 2020a.

Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Generative voxelnet: learning energy-based models for 3d shape synthesis and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020b.

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)*, pp. 14976–14985, Virtual Event, 2021a.

Jianwen Xie, Zilong Zheng, and Ping Li. Learning energy-based model with variational auto-encoder as amortized sampler. In *Proceeding of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 10441–10451, Virtual Event, 2021b.

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, 2021c.

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/CVF International Conference on Computer Vision (ICCV)*, pp. 4540–4549, Seoul, Korea, 2019.

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.

Yang Zhao, Jianwen Xie, and Ping Li. Learning energy-based generative models via coarse-to-fine expanding and sampling. In *Proceeding of the 9th International Conference on Learning Representations (ICLR)*, Virtual Event, 2021.

Zilong Zheng, Jianwen Xie, and Ping Li. Patchwise generative convnet: Training energy-based models from a single natural image for internal learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2961–2970, virtual Event, 2021.

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.

A APPENDIX

A.1 MODEL ARCHITECTURE

We use the same model structure for all the experiments. For the normalizing flow model, we use the Flow++ (Ho et al., 2019) structure designed for the CIFAR-10 dataset. For the EBM model, we use the structure shown in Table 2.

3×3 Conv2d, s=1, pad=1, ch=128; Swish
Res Block, ch=256
Res Block, ch=512
Res Block, ch=1024
3×3 Conv2d, s=4, pad=0, ch=100; Swish
Sum over channel dimension

(a) EBM model architecture

3×3 Conv2d, s=1, pad=1; Swish
3×3 Conv2d, s=1, pad=1; Swish
+ 3×3 Conv2d, s=1, pad=1; Swish(input)
2×2 Average Pooling

(b) Res Block architecture

Table 2: Model architecture

A.2 EXPERIMENTAL SETTING

We have tried three different settings. In CoopFlow setting and CoopFlow(long) setting, we train both normalizing flow model and Langevin flow model from scratch. The difference between them are the number of Langevin steps use. The CoopFlow(Long) use a relatively long chain of 200 steps MCMC updates. Following Ho et al. (2019), we use data-dependent parameter initipalization (Salimans & Kingma, 2016) for our normalizing flow model. On the other hand, in the CoopFlow(Pre) setting, we first pretrained the normalizing flow model using true observed data. Then we use this pretrained normalizing flow model as initialization and train it together with our Langevin flow model using Algorithm 1. To make this switching smoother, we include a warm-up phase for Langevin flow model. During this phase, instead of updating both the normalizing flow model and Langevin flow model from the beginning, we first fix the weight of pretrained normalizing flow model and only update the weight of EBM. After a certain number of epochs of updating, the Langevin flow model itself has been able to roughly capture the underlying data distribuion. Then we beginning to update both the normalizing flow and Langevin flow together. This may prevent the Langevin flow from proposing bad samples at the begining of training and affecting the pretrained normalizing flow model.

During training, we use 1e-4 as learning rate for both the normalizing flow and Langevin flow model, we use Adam (Kingma & Ba, 2015) as our optimizer. We use $\beta s = (0.9, 0.999)$ for normalizing flow model and $\beta s = (0.5, 0.5)$ for Langevin flow. We keep monitoring the FID scores during training and pick the best one over the training trajectory. For data augmentation, we only include random horizontal flip for the CIFAR-10 dataset and do not apply other tricks. Given we are actually using the short-run MCMC setting, we remove the noise term in Langevin updates so that the updates becomes gradient descent over the energy function. The other hyper-parameter setting can be found in Table 3.

Model	Dataset	# of flow pretrained epoch	# of Langevin flow warm-up epoch	# of Cooperative training epoch	# of Langevin updates	Langevin step size (train)	Langevin step size (test)
CoopFlow	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(Long)	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 3: Hyper-parameter setting for our experiments.

A.3 Analysis of hyperparameters of Langevin flow

We further show the influence of Langevin step size and number of Langevin steps on the CIFAR-10 dataset using the CoopFlow(Pre) model. We first fix the Langevin step size to be 0.03 and try different number of Langevin updates from 10 to 50. The results are shown in Table 4. On the other hand, we show the influence of Langevin Step size in Table 5. Here, we keep the number of Langevin updates to be 30 and we adjust the training Langevin step size. We keep the testing step size and training step size to have a ratio of $\frac{4}{3}$. We can see that our choice of 30 as number of Langevin steps and 0.03 as Langevin step size is reasonable. Based on the comparison between CoopFlow and CoopFlow(Long), further increase the number of Langevin steps to a relatively long chain may helps with the FID score, but given long MCMC chain can be computationally expensive, we keep to use 30 steps here.

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

Table 4: FID score under different number of Langevin updates of CoopFlow(Pre) on CIFAR-10.

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 5: FID score on CIFAR-10 dataset under different Langevin step size.

A.4 Ablation Study

Here we compare the results of training normalizing flow and Langevin flow using or Cooperative algorithm with training them separately. Training flow itself follows the original normalizing flow training algorithm (we use the flow++ (Ho et al., 2019) structure here, which is the same as the one used in our CoopFlow model). And for training Langevin flow individually, we use the short-run MCMC proposed by Nijkamp et al. (2019) with our own EBM structure. Through our experiments, we find that removing the noise injection to both observed data and Langevin updates can give us better result. We show generated samples of normalizing flow and short-run EBM on CIFAR-10 dataset in Figure 7 and FID scores in Table 6. We can see that the CoopFlow outperforms both the normalizing flow and short-run EBM by a large margin.

Model	Flow++ only	Short-run EBM	CoopFlow
FID ↓	92.10	49.51	21.16

Table 6: FID score of normalizing flow and short-run EBM on CIFAR-10 dataset.

A.5 More Generation Results

In Section 5.2, we show the samples from CoopFlow(Pre). Here, we show the samples generated by CoopFlow in Figure 8 and samples generated by CoopFlow(Long) in Figure 9. We can see that all the samples are meaningful. On the other hand, we the initial samples from the normalizing flow trained under the CoopFlow setting in Figure 10. Comparing Figure 3 and Figure 10, we can see that there is a gap between the normalizing flow samples and the final CoopFlow outputs. The initial samples from normalizing flow are blurred and after modified by the Langevin flow, the samples become sharp and clear. This agrees with our discussion in Section 4.2. The CoopFlow(Long) setting and CoopFlow(pre) setting give similar observations. We show more image inpainting results in Figure 11.

(a) Normalizing Flow (b) Short-run EBM

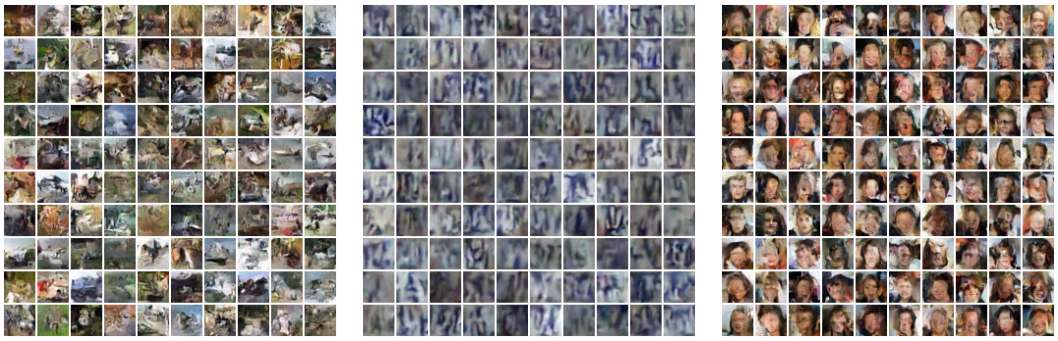
Figure 7: Generated samples of normalizing flow and short-run EBM on CIFAR-10 dataset.

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

Figure 8: Generated samples from CIFAR-10, SVHN and Celeba dataset. All images are at resolution 32×32. Samples are got from the CoopFlow setting.

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

Figure 9: Generated samples from CIFAR-10, SVHN and Celeba dataset. All images are at resolution 32×32. Samples are got from the CoopFlow(Long) setting.



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

Figure 10: The initial proposed samples of normalizing flow model. Samples are got from the CoopFlow setting.

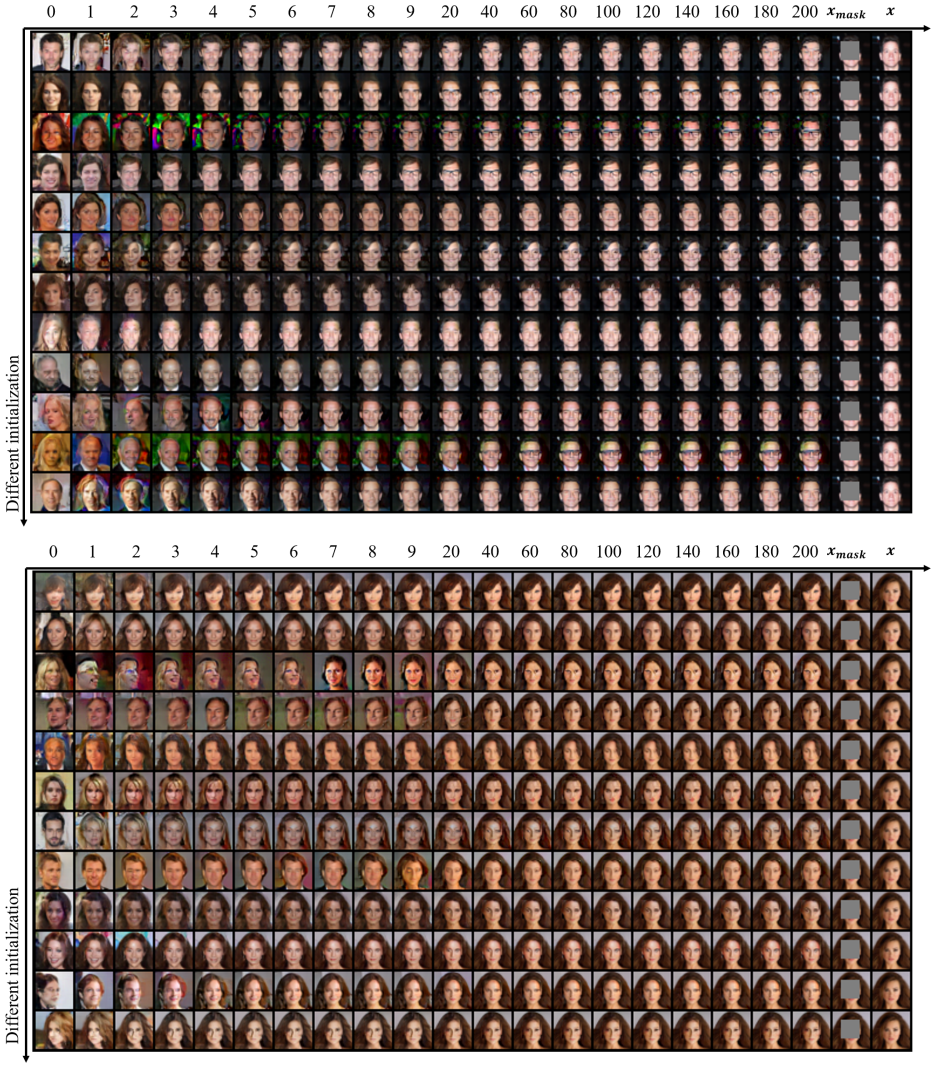
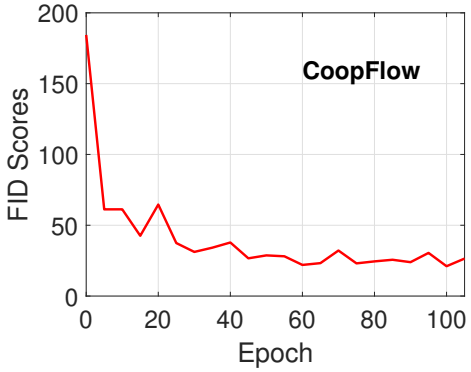
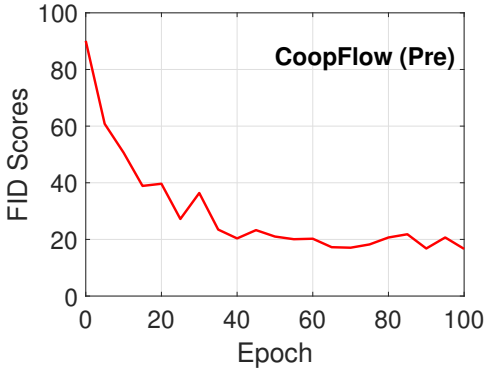


Figure 11: More results on image inpainting.



(a) FID curve for CoopFlow.



(b) FID curve for CoopFlow (Pre).

Figure 12: FID scores on CIFAR-10 dataset at each training epoch. We test the FID score for every 5 epochs.

In Figure 12, we show the FID score curve over training epochs. The FID keep decreasing as the learning proceeds. We can see the FID curves converge to a low value for both CoopFlow setting and CoopFlow(pre) on CIFAR-10 dataset. This is an evidence to show that our algorithm is a convergent algorithm.

A.6 Quantitative Results For The Image Reconstruction Experiment

Following Nijkamp et al. (2019), we calculate the per pixel MSE on 1,000 examples in the leave-out testset of CIFAR-10 dataset. We use a 200-step gradient descent to minimize the reconstruction loss. We plot the MSE per update iteration in Figure 13 and we report the final per pixel MSE in Table 7. For a baseline, we test the reconstruction of the same 1,000 images using the short-run EBM model which has the same EBM structure as our CoopFlow and uses 100 steps of MCMC. We can see the CoopFlow works better than the short-run EBM in this image reconstruction task.

Model	MSE
Short-run EBM	0.1083
CoopFlow (ours)	0.0254

Table 7: Reconstruction error (MSE per pixel).

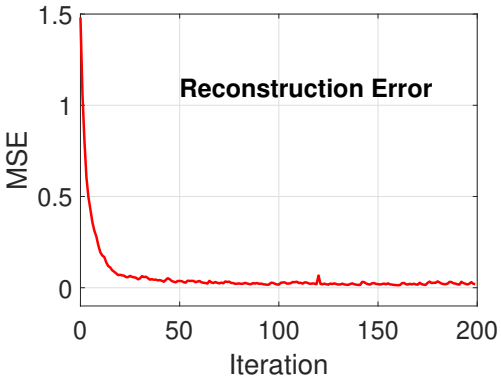


Figure 13: Reconstruction error (MSE per pixel) at each update iteration.

Model	# of Parameters	FID ↓
NT-EBM (Nijkamp et al., 2020a)	23.8M	78.12
GLOW (Kingma & Dhariwal, 2018)	44.2M	45.99
EBM-FCE (Gao et al., 2020)	44.9M	37.30
CoopFlow	45.9M	21.16
CoopFlow(pre)	45.9M	15.80
Flow++ only	28.8M	92.10
VAEBM (Xiao et al., 2021)	135.1M	12.16

Table 8: A comparison of model sizes and FID scores for different models. FID scores are reported on CIFAR-10 dataset.

A.7 Comparison of model sizes

In Table 8, we present an comparison of different models according to their size and the FID scores they got. Here we mainly compare those models that employ a normalizing flow as a generator, e.g. EBM-FCE, NT-EBM, GLOW or EBM jointly trained with other generator structure, e.g., VAEBM. We can see our CoopFlow achieves a good computation and performance balance. On one hand, we have similar number of parameters as EBM-FCE but has a much lower FID. Also, the Flow++ only represents the setting that we train the normalizing flow model with the same structure as the one used in our CoopFlow. Comparing it to CoopFlow, we can see that introducing a Langevin Flow increases the FID a lot. On the other hand, the VAEBM model achieves better FID than us, but they need to use a pretrained NVAE model (Vahdat & Kautz, 2020), which is much larger than the one we used here.

A.8 Analysis of the number of Langevin steps

In this section, we compare our method with other baselines in generating examples with the same number of Langevin steps. We compare our methods with two closely related baselines, (1) the single EBM with short-run MCMC starting from noise distribution (Nijkamp et al., 2019), and (ii) Cooperative learning of an EBM and a generic generator (Xie et al., 2020a). The numbers of MCMC steps and FID reported in Table 9 are collected from the original papers. In Table 10, we also report the FID scores of different baseline methods using the different numbers of MCMC steps. We can see that with the same number of Langevin steps, the CoopFlow can generate much more realistic image patterns than the other two baselines. The results of the experiments show that the proposed CoopFlow can use less number of MCMC steps to achieve better results.

Model	# of MCMC steps	FID ↓
Short-run EBM (Nijkamp et al., 2019)	100	44.50
CoopNet (Xie et al., 2020a)	10	33.61
CoopFlow(Pre)	30	15.80

Table 9: A summary of the reported numbers of the MCMC steps and the corresponding FID scores for Short-run EBM (EBM-SR), CoopNet and CoopFlow model on CIFAR-10 dataset.

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
CoopNet	33.74	33.48	34.12	33.85	42.99	38.88
CoopFlow(Pre)	16.46	15.20	15.80	16.80	15.64	17.94

Table 10: FID of Short-run EBM (EBM-SR), CoopNet and CoopFlow model under different numbers of MCMC steps on CIFAR-10 dataset.

A.9 GRADUALLY REDUCING NOISE INJECTION IN THE LANGEVIN FLOW

While in the paper, we completely remove the noise term in Langevin updates to achieve better results, here we try an alternative way. We add a noise term that is large at the beginning of the training and gradually reduces to 0 as the training goes. The magnitude of the noise follows:

$$\text{magnitude} = \max((1.0 - \frac{\text{epoch}}{K})^{20}, 0.0) \tag{10}$$

where K is a hyper-parameter controlling the decreasing speed of noise term. This gradually reducing noise term enables the model to do more exploration at the beginning of training and does not influence the generation quality when the model is about to converge. Note that we only add decreasing noise during training and still set the noise to zero during image generation and FID calculation. We carried out experiments on CIFAR-10 and SVHN datasets using the CoopFlow(Pre) setting. The results are shown in Table 11. We can see that models trained with reducing noise achieve similar or slightly better results than models trained with no noise.

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

Table 11: FID-score for models trained with gradually reducing noise.

A.10 MORE 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., Younes (1999); Xie et al. (2016); Kingma & Dhariwal (2018). That is, each of them has a fixed point. The only interaction between these two algorithms in the proposed CoopFlow algorithm is that, in each iteration, they feed each other with their synthesized examples and use the cooperatively synthesized examples in their parameter update formulas. To be specific, the normalizing flow uses its synthesized examples to initialize the MCMC of the EBM, and the EBM feed the normalizing flow with its synthesized examples as training examples. The synthesized examples from the Langevin flow are considered the cooperatively synthesized examples by two models, and used to compute their learning gradients. Unlike other amortized sampling methods (e.g., Han et al. (2019); Grathwohl et al. (2021)) that uses variational learning, the EBM and normalizing flow in our framework don't 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 and have 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 distance between the EBM and the data distribution. Since the MCMC of EBM in our case is initialized by the normalizing flow, it follows a modified contrastive divergence algorithm. That is, at iteration t, it has the following objective,

$$\theta^{(t+1)} = \arg\min_\theta \mathbb{D}_{\text{KL}}(p_{\text{data}}||p_\theta) - \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}||p_\theta). \tag{11}$$

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 contrastive divergence has been studied by Hinton (2002); Carreira-Perpiñán & Hinton (2005).

The convergence of MLE learning of normalizing flow. The normalizing flow's objective is to learn to minimize the KL divergence between the normalizing flow and the Langevin flow because, in each learning iteration, the normalizing flow uses the EBM's synthesized examples as training data. At iteration t, it has the following objective

$$\alpha^{(t+1)} = \arg\min_{\alpha} \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}} || q_\alpha), \tag{12}$$

which is a convergent algorithm at each t. The convergence has been studied by Younes (1999).

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

$$\theta^* = \arg\min_{\theta} \mathbb{D}_{\text{KL}}(p_{\text{data}} || p_\theta) - \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^*} q_{\alpha^*} || p_\theta), \tag{13}$$

$$\alpha^* = \arg\min_{\alpha} \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^*} q_{\alpha^*} || q_\alpha). \tag{14}$$

The convergence of the cooperative learning framework (CoopNets) that integrates MLE algorithm of an EBM and MLE algorithm of a generic generator has been verified and discussed in Xie et al. (2020a). 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 discuss where the fixed point will be in our algorithm when the MCMC of the EBM is non-mixing and non-convergent. This will go beyond all the prior works about cooperative learning.

A.11 Discussion on the bias of the contrastive divergence objective for EBM

Bias in CD learning. The EBM trained by contrastive divergence (CD) still follows the "analysis by synthesis" scheme to update the model parameters θ. The learning gradient of θ is still computed by

$$L'(\theta) = \mathbb{E}_{p_{\text{data}}}[\nabla_\theta f_\theta(x)] - \mathbb{E}_{p_\theta}[\nabla_\theta f_\theta(x)] \approx \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta f_\theta(x_i) - \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta f_\theta(\tilde{x}_i), \tag{15}$$

except that the synthesized examples $\tilde{x}$ are sampled by a short-run MCMC initialized by the training data p_{data}. For example, the CD-1 is commonly used to train Restricted Boltzmann machines (Hinton, 2002), and "1" means the single-step MCMC. We denote $\tilde{x} \sim \mathcal{K}_\theta p_{\text{data}}(x)$, where $\mathcal{K}_\theta$ is the MCMC transition kernel of T steps of MCMC that samples p_θ, and $(\mathcal{K}_\theta p_{\text{data}})(x)$ is the distribution of x after T steps of MCMC update starting from q_{data}. Because $\tilde{x} \sim \mathcal{K}_\theta p_{data}(x) \neq p_\theta(x)$, therefore, the estimated θ using the above equation is biasd (i.e., it is not an MLE solution, but it still a convergent algorithm with a fixed point.) Hinton (2002) point out that the CD learning actually minimizes the following objective, which is a perturbation of the MLE,

$$\min_{\theta} \underbrace{\mathbb{D}_{\text{KL}}(p_{\text{data}} || p_\theta)}_{\text{MLE loss}} - \underbrace{\mathbb{D}_{\text{KL}}(\mathcal{K}_\theta p_{\text{data}} || p_\theta)}_{\text{bias due to the gap}}, \tag{16}$$

where the first term is the MLE due to the use of Eq. (15), the second term is the bia due to the gap between the actually used sampling distribution $\mathcal{K}_\theta p_{\text{data}}$ and the correct distribution $p_\theta(x)$ we should sample from. Only if $\mathbb{D}_{\text{KL}}(\mathcal{K}_\theta p_{\text{data}} || p_\theta) = 0$, the CD estimate will be unbiased. However, the original CD doesn't explicitly minimize the second term. Recently, Du et al. (2021) improves the CD algorithm by explicitly minimizing the second term that has been neglected in the original CD to further reduce the bias. We need to point out that, even though the CD is biased, it is still a convergent algorithm, whose convergence has been well studied in its original paper (Hinton, 2002). "Biasd" doesn't mean it is wrong. "Biasd" is just because $\mathbb{D}_{\text{KL}}(\mathcal{K}_\theta p_{data} || p_\theta) \neq 0$.

Bias in amortized inference. Actually, many useful algorithms and models using amortized sampling or amortized inference are biased (with respect to MLE). For example, variational auto-encoder (VAE) (Kingma & Welling, 2014), whose objective is also a perturbation of MLE,

$$\min_{\alpha,\phi} \underbrace{\mathbb{D}_{\text{KL}}(p_{\text{data}}(x)||q_\alpha(x))}_{\text{MLE loss}} + \underbrace{\mathbb{D}_{\text{KL}}(\pi_\phi(z|x)||q_\alpha(z|x))}_{\text{bias due to variational inference}} \tag{17}$$

where q_α is the distribution of the decoder/generator in VAE, and $\pi_\phi(z|x)$ is the encoder to approximate the true posterior $q_\alpha(z|x)$. Even though VAE explicitly minimizes the second term (i.e., bias) in its loss, but in practice, this term can not be zero due to the mean-field design of the inference model π_ϕ, the model α is still biased to MLE. Even though our model is not related to VAE, we just want to mention that "bias" is not a bad thing, it is just a trade-off between computation efficiency and the exact MLE solution due to the usage of amortized sampling/inference.

Bias in CoopFlow. Now let us go back to our CoopFlow algorithm. The update of the EBM in our CoopFlow can be considered a modified CD algorithm, because its MCMC starts from the normalizing flow q_α. At iteration t, the update of the EBM follows the following objective

$$\theta^{(t+1)} = \arg\min_\theta \underbrace{\mathbb{D}_{\text{KL}}(p_{\text{data}}||p_\theta)}_{\text{MLE loss}} - \underbrace{\mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}||p_\theta)}_{\text{bias due to amortized sampling}}, \tag{18}$$

which is biased with respect to MLE if $\mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}||p_\theta) \neq 0$. However, the CoopFlow algorithm has the other part that can handle the second term. It is the MLE learning of the normalizing flow $q_\alpha(x)$, with $\tilde{x} \sim \mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}$ as training examples. That is, at iteration t, we fix $\mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}$ and minimize the following objective

$$\alpha^{(t+1)} = \arg\min_\alpha \mathbb{D}_{\text{KL}}(\mathcal{K}_{\theta^{(t)}} q_{\alpha^{(t)}}||q_\alpha.) \tag{19}$$

Eq. (19) tries to reduce the KL distance between the flow and the EBM. In an ideal case where capacities of all models are large enough and the MCMC $\mathcal{K}_\theta$ is mixing and convergent MCMC, minimizing Eq. (19) will lead to $\mathbb{D}_{\text{KL}}(\mathcal{K}_\theta q_\alpha||q_\alpha) \to 0$, which means $q_\alpha = p_\theta$ because q_α has become the stationary distribution of $\mathcal{K}_\theta$. If $q_\alpha = p_\theta = 0$, then the second term of Eq. (18) will be vanishing because $\mathbb{D}_{\text{KL}}(\mathcal{K}_\theta q_\alpha||p_\theta) = \mathbb{D}_{\text{KL}}(\mathcal{K}_\theta p_\theta||p_\theta) = 0$, the Eq. (18) only has the first term left, which is the MLE solution. Since $q_\alpha = p_\theta$, thus α is also MLE solution. Therefore, our estimates of θ and α in the CoopFlow are unbiased in the ideal case.

However, in practice, the conditions, such as large enough capacity of models and convergent MCMC are very hard to get. Eventually, the CoopFlow might converge to a fixed point but not the fixed point of MLE. We have used information geometry to illustrate the fixed point in Figure 1 of the main text, and analyze the biased estimator in the paragraph "moment matching estimator" of section 4.2. That is, even though the CoopFlow is not MLE solution, but it is a moment matching estimator (MME), which is a perturbation of MLE shown in Figure 1. MLE is shown as $p_{\theta_{MLE}}$ in Figure 1, and the obtained MME solution is marked as a star π^*. Again, bias doesn't means wrong. $\pi_{\theta,\alpha}$ with MME is a valid generator.

A.12 Comparison between CoopFlow and short-run EBM via information geometry

Figure 14 illustrates the convergences of both 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 (Xie et al., 2016)

$$p_\theta(x) = \frac{1}{Z(\theta)} \exp[f_\theta(x)] q_0(x), \tag{20}$$

which is an exponential tilting of a known reference distribution $q_0(x)$. In general, the reference distribution can be either a Gaussian distribution or a uniform distribution. When the reference distribution is a uniform distribution, q_0 can be removed in Eq. (20). Since the initial distribution of the CoopFlow is a Gaussian distribution q_0, which is actually the prior distribution of the normalizing flow. For a convenient and fair comparison, we use Gaussian distribution as the reference distribution of the EBM in Eq. (20). And both CoopFlow and the baseline short-run EBM use the same EBM defined in Eq. (20) in their frameworks. We use $\bar{p}_{\bar{\theta}}$ to denote the baseline short-run EBM.

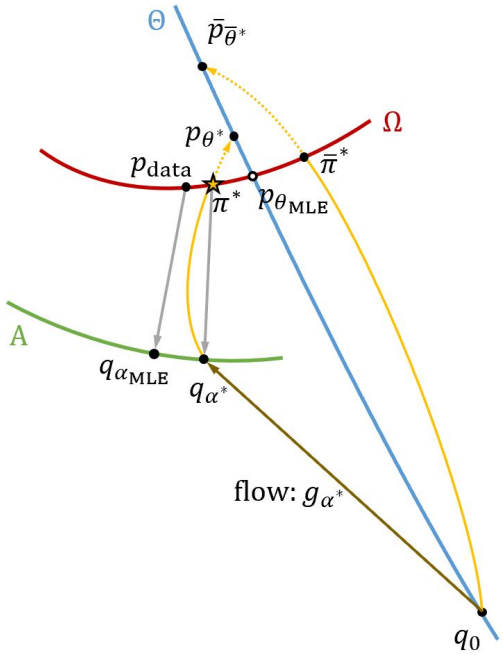


Figure 14: A comparison between the CoopFlow and the EBM with a pure short-run MCMC.

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 14, which is an extension of Figure 1 by adding the following elements. (i) q_0, which is the initial distribution for both CoopFlow and 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^*}. (ii) 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. (iii) 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. To compare the MCMC trajectories of the CoopFlow and the short-run EBM, we can find that CoopFlow has a much shorter MCMC trajectory than that of the short-run EBM, since the normalizing flow amortizes the sampling workload for the Langevin flow in the CoopFlow model.