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

ICLR

1. Introduction and Motivation

(1) An EBM parameterized by a ConvNet can be trained by MCMC-based maximum likelihood estimation (Xie et al. 2016)[1] .

(2) However, the Langevin MCMC sampling on a highly multi-modal energy function is generally not mixing, which leads to a short-run Langevin flow model (Nijkamp et al. 2019)[2] .

(3) Motivated by reducing the number of steps in the Langevin flow, this paper proposes the CoopFlow model that trains a Langevin flow jointly with a normalizing flow (Jonathan et al. 2019) [3] in a cooperative learning scheme (Xie et al. 2018)[4].

[1] Jianwen Xie, et al. "A Theory of Generative ConvNet." ICML, 2016.
[2] Erik Nijkamp, et al. "Learning Non-Convergent Non-Persistent Short-Run MCMC Toward Energy-Based Model." NeurIPS, 2019.
[3] Jonathan Ho, et al. "Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design." ICML, 2019
[4] Jianwen Xie, et al. "Cooperative Training of Descriptor and Generator Networks." TPAMI, 2018

2. Langevin Flow

Energy-based model

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

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

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

2. Langevin Flow

Maximum likelihood learning

Suppose we observe training examples $\{x_i, i = 1, ..., n\}$ from unknown data distribution $p_{\text{data}}(x)$, the energy-based model can be trained by maximum likelihood estimation. The learning gradient is given by

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

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 $p_\theta(x)$ via MCMC, respectively.

2. Langevin Flow

Langevin dynamics as MCMC

Generating synthesized examples from $p_\theta(x)$ can be accomplished with Langevin dynamics, which is applied as follows

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

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

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

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

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. According to the law of thermodynamics, $\mathbb{D}_{\text{KL}}(\tilde{p}_\theta(x)||p_\theta(x))$ decreases to zero monotonically as $T \to \infty$.

3. Normalizing Flow

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

$$x = g_\alpha(z); z \sim q_0(z),$$

where $q_0(z)$ is a Gaussian distribution, and g_α consists of a sequence of L invertible transformations.

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. The log-likelihood of x can be easily computed by

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

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

(1) We accept the fact that the short-run non-convergent MCMC is inevitable and more affordable in practice.

(2) We treat a non-convergent short-run Langevin flow as a generator

(3) We propose to jointly train the Langevin flow with a normalizing flow as a rapid initializer for more efficient generation.

(4) The resulting generator is called CoopFlow, which consists of a Langevin flow and a normalizing flow.

4. CoopFlow: Cooperative Training of Two Flows

The distribution of the CoopFlow

The density of the CoopFlow $\pi_{(\theta,\alpha)}(x)$ can be implicitly expressed by

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

where θ is the parameters of the EBM, and α is the parameters of the normalizing flow. $\mathcal{K}_\theta(x|x')$ is the transition kernel of T steps of Langevin dynamics that samples p_θ

4. CoopFlow: Cooperative Training of Two Flows

The CoopFlow Algorithm

At each iteration, we perform

(**Step 1**) 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 $\hat{x}_i = g_\alpha(z_i)$.

(**Step 2**) 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 $\tilde{x}_i$.

(**Step 3**) We update α of the normalizing flow by treating $\tilde{x}_i$ as training data.

(**Step 4**) We 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.

4. Understanding the Learned Two Flows

Moment Matching Estimator

Consider a simple EBM with $f_\theta(x) = \langle \theta, h(x) \rangle$, where $h(x)$ is the feature statistics. The CoopFlow π^* is a moment matching estimator, i.e., $\mathbb{E}_{p_{\text{data}}}[h(x)] = \mathbb{E}_{\pi^*}[h(x)]$.

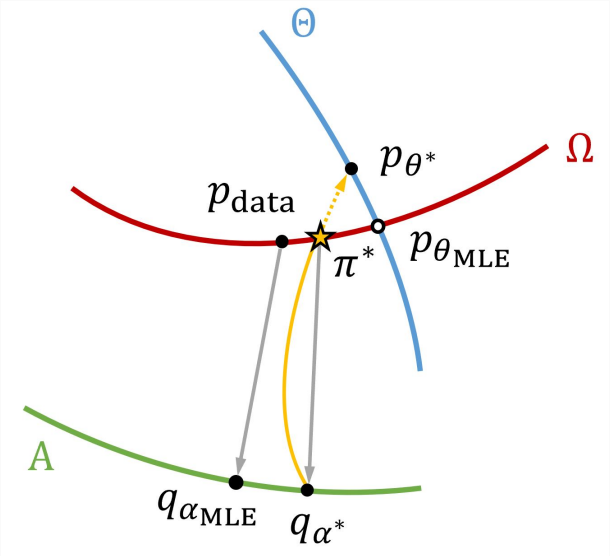

Figure 1: Illustration of convergence of the CoopFlow.

$$\Omega = \{p : \mathbb{E}_p[h(x)] = \mathbb{E}_{p_{\text{data}}}[h(x)]\},$$
$$\Theta = \{p_\theta(x) = \exp(\langle \theta, h(x) \rangle)/Z(\theta), \forall \theta\},$$
$$A = \{q_\alpha, \forall \alpha\},$$

5. Experiments

Exp 1: Image Generation

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

Figure 2: Generated examples (32 $\times$ 32 pixels) by CoopFlow models trained from CIFAR-10, SVHN and Celeba datasets respectively. Samples are obtained from the setting of CoopFlow(pre).

5. Experiments

Exp 1: Image Generation

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., 2021)	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

Exp 2: Image Reconstruction

The CoopFlow model is a latent variable generative model:

$$z \sim p_0(z); \hat{x} = g_\alpha(z); x = F_\theta(\hat{x}, e),$$

where z is the latent variables, e are all injected noises in the Langevin flow, and F_θ is the mapping realized by a T-step Langevin flow.

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.

5. Experiments

Exp 2: Image Reconstruction

Figure 3: Reconstruction of images.

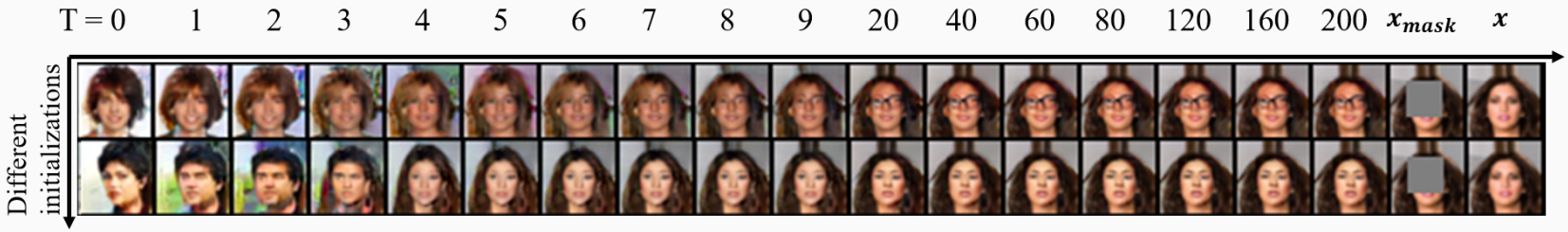

Figure 4: Image inpainting. Each row represents one different initialization. The last two columns display the masked and original images respectively.

5. Experiments

Exp 3: Interpolation in the Latent Space

The CoopFlow 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. We then infer z by the inversion of the normalizing flow $z^* = g_\alpha^{-1}(\hat{x}^*)$.

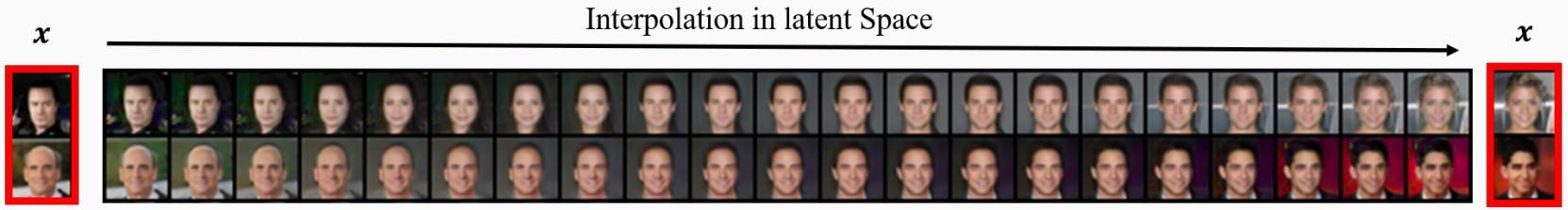

Figure 5: Image interpolation results on Celeba dataset (32×32). The leftmost and rightmost columns represent the images we observed. The columns in the middle represent the interploation results between the inferred latent vectors of the observed images.

6. Conclusion

(1) We study amortized sampling for training a short-run non-mixing Langevin sampler toward an EBM.

(2) 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.

(3) We provide both theoretical and empirical justifications for the proposed CoopFlow algorithm.