Learning Multi-layer Latent Variable Model via Variational Optimization of Short Run MCMC for Approximate Inference

Erik Nijkamp[1⋆], Bo Pang[1⋆], Tian Han[2], Linqi Zhou[1], Song-Chun Zhu[1], and Ying Nian Wu[1]

[1] University of California, Los Angeles
{enijkamp,bopang,linqi.zhou}@ucla.edu, {sczhu,ywu}@stat.ucla.edu
[2] Stevens Institute of Technology
than6@stevens.edu

Abstract. This paper studies the fundamental problem of learning deep generative models that consist of multiple layers of latent variables organized in top-down architectures. Such models have high expressivity and allow for learning hierarchical representations. Learning such a generative model requires inferring the latent variables for each training example based on the posterior distribution of these latent variables. The inference typically requires Markov chain Monte Caro (MCMC) that can be time consuming. In this paper, we propose to use noise initialized non-persistent short run MCMC, such as finite step Langevin dynamics initialized from the prior distribution of the latent variables, as an approximate inference engine, where the step size of the Langevin dynamics is variationally optimized by minimizing the Kullback-Leibler divergence between the distribution produced by the short run MCMC and the posterior distribution. Our experiments show that the proposed method outperforms variational auto-encoder (VAE) in terms of reconstruction error and synthesis quality. The advantage of the proposed method is that it is simple and automatic without the need to design an inference model.

1 Introduction

Deep generative models have seen many applications such as image and video synthesis, and unsupervised or semi-supervised learning. Such models usually consist of one or more layers of latent variables organized in top-down architectures. Learning such latent variable models from training examples is a fundamental problem, and this paper studies this problem for top-down models with multiple layers of latent variables. Such models have high expressivity and allow for learning hierarchical representations.

Learning latent variable models requires inferring the latent variables based on their joint posterior distribution, i.e., the conditional distribution of the latent

⋆ Equal contribution.

variables given each observed example. The inference typically requires Markov chain Monte Carlo (MCMC) such as Langevin dynamics [22] or Hamiltonian Monte Carlo (HMC) [24]. Such MCMC posterior sampling can be time consuming and difficult to scale up. The convergence of MCMC sampling in finite time is also questionable, especially if the posterior distribution is multi-modal.

An alternative to MCMC posterior sampling is variational inference, such as variational auto-encoder (VAE) [20, 29], which learns an extra inference network that maps each input example to the approximate posterior distribution. Despite the success of VAE, it has the following shortcomings. (1) It requires a separate inference model with a separate set of parameters. These parameters are to be learned together with the parameters of the generative model. (2) The design of the inference model is not automatic, especially for generative models with multiple layers of latent variables, which may have complex relationships governed by their joint posterior distribution. It is a highly non-trivial task to design an inference model to adequately capture the explaining-away competitions and bottom-up and top-down interactions between layers of latent variables [23, 32].

The goal of this paper is to completely do away with a separate inference model. Specifically, we propose to use noise initialized non-persistent short run MCMC [25], such as finite step Langevin dynamics, as an approximate inference engine. In the learning process, for each training example, we always initialize such a short run MCMC from the prior distribution of the latent variables, such as Gaussian or uniform noise distribution, and run a fixed finite number (e.g., 25) of steps. Thus the short run MCMC is non-persistent. In agreement with the philosophy of variational inference, we accept the approximate nature of short run MCMC, and we optimize the step size, or in general, algorithmic hyper-parameters of the short run MCMC, by minimizing the Kullback-Leibler divergence between the approximate distribution produced by the short run MCMC and the posterior distribution. This is a variational optimization, except that the variational parameter is the step size. Our experiments show that the proposed method outperforms VAE for multi-layer latent variable models in terms of reconstruction error and synthesis quality.

One major advantage of the proposed method is that it is simple and automatic. For models with multiple layers of latent variables that may be organized in complex top-down architectures, the gradient computation in Langevin dynamics is automatic on modern deep learning platforms. Such dynamics naturally integrates explaining-away competitions and bottom-up and top-down interactions between multiple layers of latent variables. It thus enables researchers to explore flexible generative models without dealing with the challenging task of designing and learning the inference models.

One class of generative models that are of particular interest are biologically plausible models, such as Boltzmann machine [1] and the generation model of the Helmholtz machine [15], where each node is a latent variable. With such a large number of latent variables, designing an inference network to regulate the bottom-up and top-down flows of information as well as lateral inhibitions

becomes a daunting task. However, short run MCMC is automatic, natural, and biologically plausible as it may be related to attractor dynamics [17, 2, 27].

2 Contributions and related work

The following are contributions of our paper.

- We propose short run MCMC for approximate inference of latent variables in deep generative models.
- We provide a method to determine the optimal step size, or in general, hyper-parameters of the short run MCMC.
- We demonstrate learning of multi-layer latent variable models with high quality samples and reconstructions.

The following are themes related to our work.

(1) *Variational inference.* As mentioned above, VAE [20, 29, 32, 9] is the prominent method for learning generator network. Our short run MCMC can be considered an inference model, except that it is intrinsic to the generative model in that it is based on the parameters of the generative model. Thus there is little mismatch between the inference process and the generative model, even at the beginning stage of the learning algorithm. Only algorithmic hyper-parameters such as step size are optimized by variational criterion. It is particularly convenient for models with multiple layers of latent variables, whereas designing variational inference models for such generative models can be a highly non-trivial task.

(2) *Alternating back-propagation.* [11] proposes to learn the generator network by maximum likelihood, and the learning algorithm iterates the following two steps: (a) inferring the latent variables by Langevin dynamics that samples from the posterior distribution of the latent variables. (b) updating the model parameters based on the inferred latent variables. Both steps involve gradient computations based on back-propagation. Similar training scheme has been developed and extended to model flexible latent prior as in [28, 26] and spatial-temporal data as in [10, 35]. [4] also leverages Langevin dynamics for posterior sampling which is however initialized from samples produced by an inference network. In the training stage, in step (a), the Langevin dynamics is initialized from the samples produced in the previous learning epoch. This is usually called persistent chain in the literature [34]. In our work, in step (a), we always initialize the finite-step (e.g., 25-step) Langevin updates from the prior noise distribution. This can be called non-persistent chain. The following are advantages of our method based on non-persistent short run MCMC as compared to methods based on persistent chain. (1) The short run MCMC can be viewed as an inference model whose hyper-parameters can be optimized based on variational criterion. This strikes a middle ground between MCMC and variational inference. (2) Theoretical underpinning of the learning method based on short run MCMC is much cleaner. (3) In both training and testing stages, the same short run MCMC is used.

(3) *Short run MCMC for energy-based model.* Recently [25] proposes to learn short run MCMC for energy-based model (EBM). An EBM is in the form of an unnormalized probability density function, where the log-density or the energy function is parametrized by a bottom-up neural network. [25] shows that it is possible to learn noise initialized non-persistent short run MCMC such as 100-step Langevin dynamics that can generate images of high synthesis quality. Our method follows a similar strategy, but it is intended for approximately sampling from the posterior distribution of latent variables.

(4) *Attractor dynamics.* In computational neuroscience, the dynamics of the neuron activities is often modeled by attractor dynamics [17, 2, 27]. However, the objective function of the attractor dynamics is often implicit, thus it is unclear what is the computational problem that the attractor dynamics is solving. For the attractor dynamics to be implemented in real time, the dynamics is necessarily a short run dynamics. Our short run MCMC is guided by a top-down model with a well-defined posterior distribution of the latent variables. It may be connected to the attractor dynamics and help us understand the latter. We shall explore this direction in future work.

3 Top-down model with multi-layer latent variables

3.1 Joint, marginal, and posterior distributions

Let x be the observed example, such as an image. Let z be the latent variables, which may consist of latent variables at multiple layers organized in a top-down architecture.

The joint distribution of (x, z) is $p_\theta(x, z)$, where θ consists of model parameters. The marginal distribution of x is $p_\theta(x) = \int p_\theta(x, z)dz$. Given x, the inference of z can be based on the posterior distribution $p_\theta(z|x) = p_\theta(x, z)/p_\theta(x)$.

The generator network assumes a d-dimensional noise vector z at the top-layer. The prior distribution $p(z)$ is known, such as $z \sim \mathcal{N}(0, I_d)$, where I_d is the d-dimensional identity matrix. Given z, $x = g_\theta(z) + \epsilon$, where $g_\theta(z)$ is a top-down convolutional neural network (sometimes called deconvolutional network due to the top-down nature), where θ consists of all the weight and bias terms of this top-down network. ϵ is usually assumed to be Gaussian white noise with mean 0 and variance σ^2. Thus $p_\theta(x|z)$ is such that $[x|z] \sim \mathcal{N}(g_\theta(z), \sigma^2 I_D)$, where D is the dimensionality of x. For this model

$$\log p_\theta(x, z) = \log[p(z)p_\theta(x|z)] \tag{1}$$

$$= -\frac{1}{2}\left[\|z\|^2 + \|x - g_\theta(z)\|^2/\sigma^2\right] + c, \tag{2}$$

where c is a constant independent of θ.

In this paper, we are mainly concerned with multi-layer generator network. While it is computationally convenient to have a single latent noise vector at the top layer, it does not account for the fact that patterns can appear at multiple layers of compositions or abstractions (e.g., face $\rightarrow$ (eyes, nose, mouth) $\rightarrow$ (edges,

corners) → pixels), where variations and randomness occur at multiple layers. To capture such a hierarchical structure, it is desirable to introduce multiple layers of latent variables organized in a top-down architecture. Specifically, we have $z = (z_l, l = 1, ..., L)$, where layer L is the top layer, and layer 1 is the bottom layer above x. For notational simplicity, we let $x = z_0$. We can then specify $p_\theta(z)$ as

$$p_\theta(z) = p_\theta(z_L) \prod_{l=0}^{L-1} p_\theta(z_l|z_{l+1}). \tag{3}$$

One concrete example is $z_L \sim \mathcal{N}(0, I)$, $[z_l|z_{l+1}] \sim \mathcal{N}(\mu_l(z_{l+1}), \sigma_l^2(z_{l+1}))$, $l = 0, ..., L-1$. where $\mu_l()$ and $\sigma_l^2()$ are the mean vector and the diagonal variance-covariance matrix of z_l respectively, and they are functions of z_{l+1}. θ collects all the parameters in these functions. $p_\theta(x, z)$ can be obtained similarly as in Equation (2).

3.2 Learning and inference

Let $p_{\text{data}}(x)$ be the data distribution that generates the example x. The learning of parameters θ of $p_\theta(x)$ can be based on $\min_\theta \text{KL}(p_{\text{data}}(x)\|p_\theta(x))$, where $\text{KL}(p\|q) = \text{E}_p[\log(p(x)/q(x))]$ is the Kullback-Leibler divergence between p and q (or from p to q since $\text{KL}(p\|q)$ is asymmetric). If we observe training examples $\{x_i, i = 1, ..., n\} \sim p_{\text{data}}(x)$, the above minimization can be approximated by maximizing the log-likelihood

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x_i), \tag{4}$$

which leads to the maximum likelihood estimate (MLE).

The gradient of the log-likelihood, $L'(\theta)$, can be computed according to the following identity:

$$\frac{\partial}{\partial \theta} \log p_\theta(x) = \frac{1}{p_\theta(x)} \frac{\partial}{\partial \theta} p_\theta(x) \tag{5}$$

$$= \int \frac{\partial}{\partial \theta} \log p_\theta(x, z) \frac{p_\theta(x, z)}{p_\theta(x)} dz \tag{6}$$

$$= \text{E}_{p_\theta(z|x)} \left[\frac{\partial}{\partial \theta} \log p_\theta(x, z) \right]. \tag{7}$$

The above expectation can be approximated by Monte Carlo samples from $p_\theta(z|x)$. The MLE learning can be accomplished by gradient descent. Each learning iteration updates θ by

$$\theta_{t+1} = \theta_t + \eta_t \frac{1}{n} \sum_{i=1}^{n} \text{E}_{p_{\theta_t}(z_i|x_i)} \left[\frac{\partial}{\partial \theta} \log p_\theta(x_i, z_i) \mid_{\theta=\theta_t} \right], \tag{8}$$

where η_t is the step size or learning rate, and $\text{E}_{p_{\theta_t}(z_i|x_i)}$ can be approximated by Monte Carlo sampling from $p_{\theta_t}(z_i|x_i)$.

4 Short run MCMC for approximate inference

4.1 Langevin dynamics

Sampling from $p_\theta(z|x)$ usually requires MCMC. One convenient MCMC is Langevin dynamics [22], which iterates

$$z_{k+1} = z_k + s\frac{\partial}{\partial z}\log p_\theta(z_k|x) + \sqrt{2s}\epsilon_k, \tag{9}$$

where $\epsilon_k \sim \mathcal{N}(0, I)$, k indexes the time step of the Langevin dynamics, and s is the step size. The Langevin dynamics consists of a gradient descent term on $-\log p(z|x)$. In the case of generator network, it amounts to gradient descent on $\|z\|^2/2 + \|x - g_\theta(z)\|^2/2\sigma^2$, which is penalized reconstruction error. The Langevin dynamics also consists of a white noise diffusion term $\sqrt{2s}\epsilon_k$ to create randomness for sampling from $p_\theta(z|x)$.

For small step size s, the marginal distribution of z_k will converge to $p_\theta(z|x)$ as $k \to \infty$ regardless of the initial distribution of z_0. More specifically, let $q_k(z)$ be the marginal distribution of z_k of the Langevin dynamics, then $\mathrm{KL}(q_k(z)\|p_\theta(z|x))$ decreases monotonically to 0, that is, by increasing k, we reduce $\mathrm{KL}(q_k(z)\|p_\theta(z|x))$ monotonically [5].

4.2 Noise initialized short run MCMC

It is impractical to run long chains to sample from $p_\theta(z|x)$. We thus propose the following short run MCMC as inference dynamics, with a fixed small K (e.g., $K = 25$),

$$z_0 \sim p(z), z_{k+1} = z_k + s\frac{\partial}{\partial z}\log p_\theta(z_k|x) + \sqrt{2s}\epsilon_k,\ k = 1, ..., K, \tag{10}$$

where $p(z)$ is the prior noise distribution of z.

We can write the above short run MCMC as

$$z_0 \sim p(z),\ z_{k+1} = z_k + sR(z_k) + \sqrt{2s}\epsilon_k,\ k = 1, ..., K, \tag{11}$$

$R(z) = \frac{\partial}{\partial z}\log p_\theta(z|x)$, where we omit x and θ in $R(z)$ for simplicity of notation. For finite K, this dynamics is a K-layer noise-injected residual network [12], or K-step noise-injected RNN [31, 16]. It may also be compared to flow-based inference model [6, 8, 7, 19, 21], except we do not learn a separate inference model.

To further simplify the notation, we may write the short run MCMC as

$$z_0 \sim p(z),\ z_K = F(z_0, \epsilon), \tag{12}$$

where $\epsilon = (\epsilon_k, k = 1, ..., K)$, and F composes the K steps of Langevin updates.

Let the distribution of z_K be $q_s(z)$, where we include the notation s to make it explicit that the distribution of z_K depends on the step size s. Recall that the distribution of z_K also depends on x and θ, so that in full notation, we may write $q_s(z)$ as $q_{s,\theta}(z|x)$.

For short run MCMC (10), the gradient term usually dominates the noise term, and most of the randomness comes from $z_0 \sim p(z)$. Given ϵ, z_K is a deterministic transformation of z_0. Assuming this transformation is invertible, and let $z_0 = F^{-1}(z_k, \epsilon)$. Let $q_s(z|\epsilon)$ be the conditional distribution of z_K given ϵ. By change of variable,

$$q_s(z|\epsilon) = p(F^{-1}(z, \epsilon))|\det(dF^{-1}(z, \epsilon)/dz)|. \tag{13}$$

Then

$$q_s(z) = \int q_s(z|\epsilon)p(\epsilon)d\epsilon = \mathrm{E}_{p(\epsilon)}[q_s(z|\epsilon)], \tag{14}$$

which can be approximated by Monte Carlo sampling from $p(\epsilon)$, i.e., the iid $\mathcal{N}(0, I)$ distribution.

For our method, we never need to compute F^{-1}, because we only need to compute $\mathrm{E}[h(z_K)] = \mathrm{E}_{q_s(z)}[h(z)]$ for a given function h, and

$$\mathrm{E}_{q_s(z)}[h(z)] = \mathrm{E}_{p(z_0)p(\epsilon)}[h(F(z_0, \epsilon))]. \tag{15}$$

In particular, we need to compute the entropy of $q_s(z)$ for variational optimization of step size s. The entropy is the negative of

$$\mathrm{E}_{q_s(z)}[\log q_s(z)] = \mathrm{E}_{p(z_0)p(\epsilon)}[\log \mathrm{E}_{p(\epsilon)}(q_s(F(z_0, \epsilon)|\epsilon))] \tag{16}$$

$$= \mathrm{E}_{p(z_0)p(\epsilon)}[\log \mathrm{E}_{p(\epsilon)}(p(z_0)/|\det(dF(z_0, \epsilon)/dz_0)|)], \tag{17}$$

where the expectations can be approximated by Monte Carlo sampling from the known prior distribution of z_0 and the known noise distribution of ϵ. In the above computation, we need to compute the determinant of the Jacobian $dF(z_0, \epsilon)/dz_0$. Fortunately, on modern deep learning platforms, such computation is easily feasible even if the dimension of z_0 is very high. Specifically, after computing the matrix $dF(z_0, \epsilon)/dz_0$, we can compute the eigenvalues of $dF(z_0, \epsilon)/dz_0$, so that the log-determinant is the sum of the log of the eigenvalues.

As to the invertibility of F, in our experience, the eigenvalues of $dF(z_0, \epsilon)/dz_0$ are always away from 0, suggesting that $z_K = F(z_0, \epsilon)$ is locally invertible. Moreover, different z_0 always lead to different $z_K = F(z_0, \epsilon)$, suggesting that F is globally invertible. Again, our method does not require inverting F.

4.3 Variational optimization of step size

We want to optimize the step size s so that $q_s(z)$ best approximates the posterior $p_\theta(z|x)$. This can be accomplished by

$$\min_s \mathrm{KL}(q_s(z)\|p_\theta(z|x)). \tag{18}$$

This is similar to variational approximation, with step size s being the variational parameter.

$$\mathrm{KL}(q_s(z)\|p_\theta(z|x)) = \mathrm{E}_{q_s(z)}[\log q_s(z) - \log p_\theta(x, z)] + \log p_\theta(x), \tag{19}$$

where the last term $\log p_\theta(x)$ is independent of s. The computation of the first two terms is explained in the previous subsection. See equations (15) and (17).

While we can optimize the step size s for each example x, in our work, we optimize over an overall s that is shared by all the examples. Reverting to the full notation $q_{s,\theta}(z|x)$ for $q_s(z)$, this means we minimizes

$$\frac{1}{n}\sum_{i=1}^{n} \mathrm{KL}(q_{s,\theta}(z_i|x_i)\|p_\theta(z_i|x_i)) \tag{20}$$

over s. The minimization can be accomplished by a grid search, or by gradient descent (the gradient is still computable on modern deep learning platforms).

Instead of using a constant step size s for all k, we may also optimize over varying step sizes $s_k, k = 1, ..., K$. We leave it to future work.

The main computational burden in optimizing algorithmic hyper-parameters such as step size comes from the computation of the entropy of $q_{s,\theta}(z_i|x_i)$. In this paper, we compute it rigorously to make the learning principled. In future work, we shall explore efficient approximate methods to optimize short run MCMC.

4.4 Learning with short run MCMC

A learning iteration consists of the following two steps. (1) Update s by minimizing (20). (2) Update θ by

$$\theta_{t+1} = \theta_t + \eta_t \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{q_{s,\theta_t}(z_i|x_i)}\left[\frac{\partial}{\partial\theta}\log p_\theta(x_i, z_i)\ |_{\theta=\theta_t}\right], \tag{21}$$

where η_t is the learning rate, $\mathrm{E}_{q_{s,\theta_t}(z_i|x_i)}$ (here we use the full notation $q_{s,\theta}(z|x)$ instead of the abbreviated notation $q_s(z)$) can be approximated by sampling from $q_{s,\theta_t}(z_i|x_i)$ using the noise initialized K-step Langevin dynamics. Compared to MLE learning algorithm (8), we replace $p_{\theta_t}(z|x)$ by $q_{s,\theta}(z|x)$, and fair Monte Carlo samples from $q_{s,\theta}(z|x)$ can be obtained by short run MCMC.

The learning procedure is summarized in Algorithm 1. Note, we only optimize s every T_s iterations, so that it does not incur much computational burden.

Algorithm 1: Learning with short run MCMC.

input : Training examples $\{x_i\}_{i=1}^n$, learning iterations T, step size updating interval T_s, learning rate η, initial parameters θ_0, batch size m, number of steps K, initial step size s.

output: Parameters θ_T.

for $t = 0 : T-1$ **do**

1. Draw observed examples $\{x_i\}_{i=1}^m$.
2. Draw latent vectors $\{z_{i,0} \sim p(z)\}_{i=1}^m$.
3. Infer $\{z_{i,K}\}_{i=1}^m$ by K steps of dynamics (10) with step size s.
4. Update θ according to (21).
5. Every T_s iterations, update s by minimizing (20).

4.5 Theoretical underpinning

Given θ_t, the updating equation (21) is a one step gradient ascent on

$$Q_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{q_{s,\theta_t}(z_i|x_i)} \left[\log p_\theta(x_i, z_i)\right]. \tag{22}$$

Compared to the log-likelihood in MLE learning, $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(x)$,

$$Q_s(\theta) = L(\theta) + \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{q_{s,\theta_t}(z_i|x_i)} \left[\log p_\theta(z_i|x_i)\right] \tag{23}$$

$$= L(\theta) - \frac{1}{n}\sum_{i=1}^{n} \mathrm{KL}(q_{s,\theta_t}(z_i|x_i)\|p_\theta(z_i|x_i)) \tag{24}$$

$$+ \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{q_{s,\theta_t}(z_i|x_i)}[\log q_{s,\theta_t}(z_i|x_i)]. \tag{25}$$

Since the last term has nothing to do with θ, gradient ascent on $Q_s(\theta)$ is equivalent to gradient ascent of $\tilde{Q}_s(\theta) = L(\theta) - \frac{1}{n}\sum_{i=1}^{n} \mathrm{KL}(q_{s,\theta_t}(z_i|x_i)\|p_\theta(z_i|x_i))$, which is a lower bound of $L(\theta)$. $\tilde{Q}_s(\theta)$ is a perturbation of $L(\theta)$. At θ_t, the optimization over s by minimizing (20) is to minimize this perturbation.

Thus a learning iteration can be interpreted as a joint maximization of $\tilde{Q}_s(\theta)$ over s and θ. Specifically, step (1) maximizes $\tilde{Q}_s(\theta)$ over s given $\theta = \theta_t$, and step (2) seeks to maximize $\tilde{Q}_s(\theta)$ over θ given s. This is similar to variational inference with s being the variational parameter.

The fixed point of the learning algorithm (21) solves the following estimating equation:

$$\frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{q_{s,\theta}(z_i|x_i)} \left[\frac{\partial}{\partial\theta} \log p_\theta(x_i, z_i)\right] = 0. \tag{26}$$

If we approximate $\mathrm{E}_{q_{s,\theta_t}(z_i|x_i)}$ by Monte Carlo samples from $q_{s,\theta_t}(z_i|x_i)$, then the learning algorithm becomes Robbins-Monro algorithm for stochastic approximation [30]. For fixed s, its convergence to the fixed point follows from regular conditions of Robbins-Monro. We expect that the optimized s will also converge to a fixed value.

It is worth stressing that $q_{s,\theta_t}(z_i|x_i)$ is the distribution under the short run MCMC. Thus fair samples can be obtained from $q_{s,\theta_t}(z_i|x_i)$ by running K steps of short run MCMC. In contrast, the MLE estimating equation is $\frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{p_\theta(z_i|x_i)} \left[\frac{\partial}{\partial\theta} \log p_\theta(x_i, z_i)\right] = 0$, where $p_\theta(z_i|x_i)$ is the posterior distribution. The MLE learning algorithm (8) requires sampling from $p_{\theta_t}(z_i|x_i)$, which can be impractical, especially for multi-modal posterior distribution, where the mixing rate of MCMC can be very slow.

In our method, our estimate is defined by the solution to the estimating equation (26), which is a perturbation of the MLE estimating equation. We

accept this bias, so that the learning algorithm can be justified as a Robbins-Monro algorithm, whose convergence can be easily established. Thus both the target and the convergence of our learning algorithm are theoretically sound.

The bias of the learned θ based on short run MCMC relative to the MLE depends on the gap between $q_{s,\theta}(z|x)$ and $p_\theta(z|x)$. We suspect that this bias may actually be beneficial in the following sense. The gradient ascent of $Q_s(\theta)$ seeks to increase $L(\theta)$ while decreasing $\frac{1}{n}\sum_{i=1}^n \mathrm{KL}(q_{s,\theta_t}(z_i|x_i)\|p_\theta(z_i|x_i))$. The latter tends to bias the learned model so that its posterior distribution $p_\theta(z_i|x_i)$ is close to the short run MCMC $q_{s,\theta_t}(z_i|x_i)$, i.e., our learning method may bias the model to make inference by short run MCMC accurate.

5 Experiments

In this section, we will demonstrate (1) realistic synthesis, (2) faithful reconstructions of observed images, (3) inpainting of occluded images, (4) learning of hierarchical representations, (5) variational grid search and gradient descent on the step size, and, (6) ablation on latent layers and Langevin steps. The baselines are trained with ladder variational autoencoder [32] for multi-layer latent variable models. We refer to the Appendix and the reference implementation[3] for details.

5.1 Synthesis

We evaluate the learned generator $g_\theta(z)$ by examining the fidelity of generated examples quantitatively on various datasets. Figure 1 depicts generated samples by our method and Ladder-VAE of size 64×64 pixels on the CelebA dataset. Figure 2 depicts generated samples of size 32×32 pixels for various datasets with $K = 25$ short run MCMC inference steps. Table 1 compares the Fréchet Inception Distance (FID) [14] with Inception v3 classifier [33] on $40,000$ generated examples for the comparable multi-layer latent variable models Ladder-VAE [32] and Glow [21] for which levels may be comparable with layers of latent variables. Even though our method is specifically crafted for multi-layer latent-variable models, Table 2 compares short run MCMC on training single-layer latent-variable models with ABP [11], GLO [3], VAE [20], and VAE with MADE [8]. Despite its simplicity, short run MCMC is competitive with elaborate means of inference in VAE models and flow-based models, such as Glow [21].

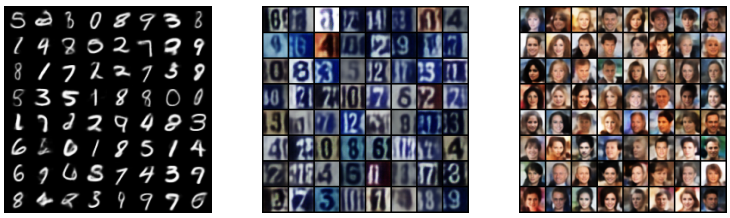
5.2 Reconstruction

We evaluate the accuracy of the learned short run MCMC inference dynamics $q_{s,\theta_t}(z|x_i)$ by reconstructing test images. In contrast to traditional MCMC posterior sampling with persistent chains, short run inference with small K allows not only for efficient learning on training examples, but also the same dynamics

[3] `https://enijkamp.github.io/project_short_run_inference/`

(a) Ladder-VAE with $L = 5$. (b) Short run inference with $K = 25$.

Fig. 1: Generated samples for models with $L = 5$ layers on CelebA ($64 \times 64 \times 3$).



(a) MNIST (28×28). (b) SVHN ($32 \times 32 \times 3$). (c) CelebA ($32 \times 32 \times 3$).

Fig. 2: Generated samples for $K = 25$ inference steps with $L = 5$ layers.

Models	MNIST		SVHN		CelebA	
	MSE	FID	MSE	FID	MSE	FID
Glow, $L = 3$	-	-	-	65.27	-	39.84
Ladder-VAE, $L = 1$	0.020	-	0.019	46.78	0.031	69.90
Ladder-VAE, $L = 3$	0.018	-	0.015	41.72	0.029	58.33
Ladder-VAE, $L = 5$	0.018	-	0.014	39.26	0.028	53.40
Ours, $L = 1$	0.019	-	0.018	44.86	0.019	45.74
Ours, $L = 3$	0.017	-	0.015	39.02	0.018	41.15
Ours, $L = 5$	0.015	-	0.011	35.23	0.011	36.84

Table 1: Comparison of generators $g_\theta(z)$ with latent layers L learned by Ladder-VAE and short run inference with respect to MSE of reconstructions and FID of generated samples for MNIST, SVHN, and CelebA ($32 \times 32 \times 3$).

	ABP [11]	GLO [3]	VAE [20]	VAE+IAF [8]	Ours
SVHN	49.71	65.52	46.78	50.41	44.86
CelebA	51.50	50.70	69.90	53.78	45.74

Table 2: Comparison of generators $g_\theta(z)$ with latent layers $L = 1$ with respect to FID of generated samples for SVHN and CelebA ($32 \times 32 \times 3$).

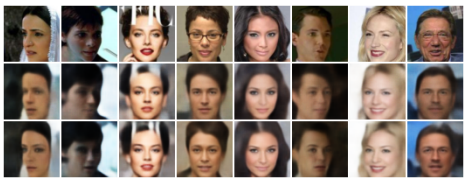


Fig. 3: Comparison of reconstructions between Ladder-VAE samples and our method on CelebA ($64 \times 64 \times 3$) with $L = 5$. *Top*: original test images. *Middle*: reconstructions from VAE. *Bottom*: reconstructions by short run inference.

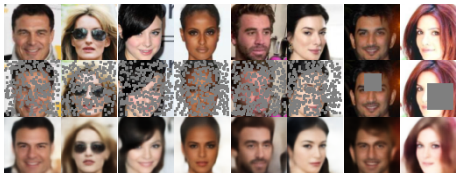


Fig. 4: Inpainting on CelebA ($64 \times 64 \times 3$) with $L = 5$ for varying occlusion masks. *Top*: original test images. *Middle*: occluded images. *Bottom*: inpainted test images by short run MCMC inference.

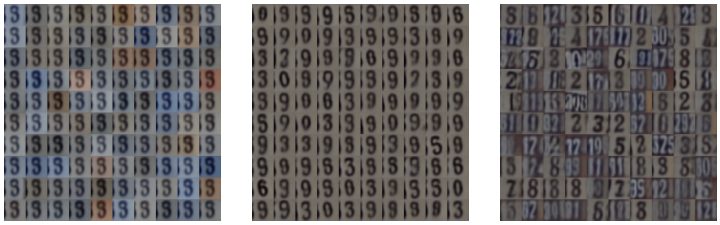
can be recruited for testing examples. Figure 3 compares the reconstructions of learned generators with $L = 5$ layers by Ladder-VAE and short run inference on CelebA ($64 \times 64 \times 3$). The fidelity of reconstructions by short run MCMC inference appears qualitatively improved over VAE, which is quantitatively confirmed by a consistently lower MSE in Table 1.

5.3 Inpainting

Our method can "inpaint" occluded image regions. To recover the occluded pixels, the only required modification of (10) involves the computation of $\|x - g_\theta(z)\|^2/\sigma^2$. For a fully observed image, the term is computed by the summation over all pixels. For partially observed images, we only compute the summation over the observed pixels. Figure 4 depicts test images taken from the CelebA dataset for which a mask randomly occludes pixels in various patterns.

5.4 Hierarchical representation

Multi-layer latent variable models not only demonstrate improved expressiveness over single-layer ones but also allow for learning the hierarchical structure. [36] argues that an alternative parameterization of the multi-layer generator promotes disentangled hierarchical features. We train a three-layer model with this parameterization using short run inference on SVHN. As shown in Figure 5,



(a) Bottom layer z_1. (b) Middle layer z_2. (c) Top layer z_3.

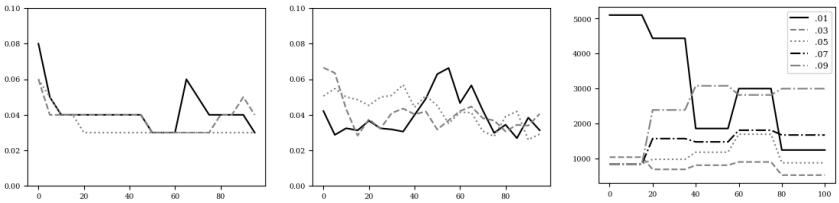
Fig. 5: Generated samples from a three-layer generator where each sub-figure corresponds to samples drawn when fixing the latent variables z of all layers except for one. (a) The bottom layer represents background color. (b) The second layer represents digit identity. (c) The top layer represents general structure.

the three-layer latent variables capture disentangled representations, which are background color, digit identity, general structure from bottom to top layer.

5.5 Variational optimization of step size

The step size s in (10) may be optimized such that $q_s(z)$ best approximates the posterior $p_\theta(z|x)$. That is, we can optimize the step size s by minimizing $\mathrm{KL}(q_s(z)\|p_\theta(z|x))$ via a grid search or gradient descent. As outlined in Section 4.3, we require $dF(z_0, \epsilon)/dz_0$. In reverse-mode auto-differentiation, we construct the Jacobian one row at a time by evaluating vector-Jacobian products. Then, we evaluate the eigenvalues of $dF(z_0, \epsilon)/dz_0$. As both steps are computed in a differentiable manner, we may compute the gradient with respect to s.

Figure 6a and 6b depict the optimal step size s over learning iterations t determined by grid-search with $s \in \{0.01, 0.02, \ldots, 0.15\}$ and gradient descent on (20). For both grid-search and gradient descent the step size settles near 0.05 after a few learning iterations. Figure 6c details the optimization objective of s, $\mathrm{E}_{q_s(z)}[\log q_s(z) - \log p_\theta(x, z)]$, with respect to individual step sizes s.



(a) Grid-search. (b) Gradient-descent. (c) Gradient-descent.

Fig. 6: (a) and (b) step size s over epochs T for three individual runs with varying random seed. (c) $\mathrm{E}_{q_s(z)}[\log q_s(z) - \log p_\theta(x, z)]$ for step sizes s over epochs T.

5.6 Influence of number of layers and steps

Tables 3a and 3b show the influence of the latent layers L for the generator network $g_\theta(z)$ and the number of steps K in the inference dynamics (10), respectively. Increasing L improves the quality of synthesis and reconstruction. Increasing K up to 25 steps results in significant improvements, while $K > 25$ appears to affect the scores only marginally.

	L		
	1	3	5
FID	61.03	52.19	47.95
MSE	0.020	0.018	0.015

(a) Varying L with $K = 25$.

	K				
	5	10	25	50	400
FID	82.79	67.38	36.84	35.39	35.16
MSE	0.045	0.037	0.011	0.010	0.010

(b) Varying K with $L = 5$.

Table 3: Influence of number of layers L and number of short run inference steps K on (a) CelebA ($64 \times 64 \times 3$) and (b) CelebA ($32 \times 32 \times 3$).

6 Conclusion

This paper proposes to use short run MCMC to infer latent variables in deep generative models, where the tuning parameters such as step size of the short run MCMC are optimized by a variational criterion. It thus combines the strengths of both MCMC and variational inference. Unlike variational auto-encoder, there is no need to design an extra inference model, which is usually a challenging task for models with multiple layers of latent variables.

The short run MCMC is easily affordable on the current computing platforms and can be easily scaled up to big training data. It will enable the researchers to develop more sophisticated latent variable models, such as biologically plausible models where each node is a latent variable and the short run MCMC can be compared to attractor dynamics in neuroscience.

This paper lays the foundation for short run MCMC for approximate inference in complex generative models, where the short run MCMC is optimized in a principled way. In our further work, we shall explore more efficient approximate methods for optimizing or learning more general forms of short run inference dynamics.

Acknowledgments The work is supported by NSF DMS-2015577, DARPA XAI N66001-17-2-4029, ARO W911NF1810296, ONR MURI N00014-16-1-2007, and XSEDE grant ASC170063. We thank NVIDIA for the donation of Titan V GPUs. We thank Eric Fischer for the assistance with experiments.

References

1. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for boltzmann machines. Cognitive Science **9**(1), 147–169 (1985). https://doi.org/10.1207/s15516709cog0901_7, https://doi.org/10.1207/s15516709cog0901_7
2. Amit, D.J.: Modeling brain function: the world of attractor neural networks, 1st Edition. Cambridge Univ. Press (1989), http://www.worldcat.org/oclc/19922497
3. Bojanowski, P., Joulin, A., Lopez-Paz, D., Szlam, A.: Optimizing the latent space of generative networks. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 599–608. PMLR (2018), http://proceedings.mlr.press/v80/bojanowski18a.html
4. Chen, C., Li, C., Chen, L., Wang, W., Pu, Y., Duke, L.C.: Continuous-time flows for efficient inference and density estimation. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 824–833. PMLR, Stockholmsmssan, Stockholm Sweden (10–15 Jul 2018), http://proceedings.mlr.press/v80/chen18d.html
5. Cover, T.M., Thomas, J.A.: Elements of information theory (2. ed.). Wiley (2006), http://www.elementsofinformationtheory.com/
6. Dinh, L., Krueger, D., Bengio, Y.: NICE: non-linear independent components estimation. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings (2015), http://arxiv.org/abs/1410.8516
7. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real NVP. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings (2017), https://openreview.net/forum?id=HkpbnH9lx
8. Germain, M., Gregor, K., Murray, I., Larochelle, H.: Made: Masked autoencoder for distribution estimation. In: International Conference on Machine Learning. pp. 881–889 (2015)
9. Gregor, K., Danihelka, I., Graves, A., Rezende, D.J., Wierstra, D.: DRAW: A recurrent neural network for image generation. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. pp. 1462–1471 (2015), http://proceedings.mlr.press/v37/gregor15.html
10. Han, T., Lu, Y., Wu, J., Xing, X., Wu, Y.N.: Learning generator networks for dynamic patterns. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 809–818. IEEE (2019)
11. Han, T., Lu, Y., Zhu, S., Wu, Y.N.: Alternating back-propagation for generator network. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA. pp. 1976–1984 (2017), http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14784
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 770–778. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.90, https://doi.org/10.1109/CVPR.2016.90

13. Hendrycks, D., Gimpel, K.: Bridging nonlinearities and stochastic regularizers with gaussian error linear units. CoRR **abs/1606.08415** (2016), http://arxiv.org/abs/1606.08415
14. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA. pp. 6626–6637 (2017)
15. Hinton, G.E., Dayan, P., Frey, B.J., Neal, R.M.: The wake-sleep algorithm for unsupervised neural networks. Science **268**(5214), 1158–1161 (1995)
16. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735, https://doi.org/10.1162/neco.1997.9.8.1735
17. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. In: Proceedings of the national academy of sciences. vol. 79, pp. 2554–2558. National Acad Sciences (1982)
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
19. Kingma, D.P., Salimans, T., Welling, M.: Improving variational inference with inverse autoregressive flow. CoRR **abs/1606.04934** (2016), http://arxiv.org/abs/1606.04934
20. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014), http://arxiv.org/abs/1312.6114
21. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. In: Advances in neural information processing systems. pp. 10215–10224 (2018)
22. Langevin, P.: On the theory of Brownian motion (1908)
23. Maaløe, L., Fraccaro, M., Liévin, V., Winther, O.: Biva: A very deep hierarchy of latent variables for generative modeling. In: Advances in neural information processing systems. pp. 6551–6562 (2019)
24. Neal, R.M.: MCMC using hamiltonian dynamics. Handbook of Markov Chain Monte Carlo **2** (2011)
25. Nijkamp, E., Hill, M., Zhu, S.C., Wu, Y.N.: Learning non-convergent non-persistent short-run MCMC toward energy-based model. Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, Canada (2019)
26. Pang, B., Han, T., Nijkamp, E., Zhu, S.C., Wu, Y.N.: Learning latent space energy-based prior model. arXiv preprint arXiv:2006.08205 (2020)
27. Poucet, B., Save, E.: Attractors in memory. Science **308**(5723), 799–800 (2005)
28. Qiu, Y., Wang, X.: Almond: Adaptive latent modeling and optimization via neural networks and langevin diffusion. Journal of the American Statistical Association pp. 1–13 (2019)
29. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014. pp. 1278–1286 (2014), http://proceedings.mlr.press/v32/rezende14.html

30. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics pp. 400–407 (1951)
31. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. nature **323**(6088), 533–536 (1986)
32. Sønderby, C.K., Raiko, T., Maaløe, L., Sønderby, S.K., Winther, O.: Ladder variational autoencoders. In: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. pp. 3738–3746 (2016), http://papers.nips.cc/paper/6275-ladder-variational-autoencoders
33. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 2818–2826 (2016). https://doi.org/10.1109/CVPR.2016.308, https://doi.org/10.1109/CVPR.2016.308
34. Tieleman, T.: Training restricted boltzmann machines using approximations to the likelihood gradient. In: Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008. pp. 1064–1071 (2008). https://doi.org/10.1145/1390156.1390290, https://doi.org/10.1145/1390156.1390290
35. Xie, J., Gao, R., Zheng, Z., Zhu, S., Wu, Y.N.: Learning dynamic generator model by alternating back-propagation through time. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. pp. 5498–5507 (2019). https://doi.org/10.1609/aaai.v33i01.33015498, https://doi.org/10.1609/aaai.v33i01.33015498
36. Zhao, S., Song, J., Ermon, S.: Learning hierarchical features from deep generative models. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 4091–4099. PMLR (2017), http://proceedings.mlr.press/v70/zhao17c.html

7 Appendix

7.1 Experiment details

All the training image datasets are resized and scaled to $[-1, 1]$ with no further pre-processing. We train the models with $T = 3 \times 10^5$ parameter updates optimized by Adam [18]. The learning rate η decays step-wise (1×10^{-4}, 5×10^{-5}, 1×10^{-5}) for each 1×10^5 iterations. If not stated otherwise, we use $K = 25$ short run inference steps and σ is gradually annealed to 0.15.

7.2 Model specification

For the multi-layer generator model, we have $z = (z_l, l = 1, \dots, L)$ for which layer L is the top layer, and layer 1 is the bottom layer close to x. For simplicity, let $x = z_0$. Then, $p_\theta(z) = p_\theta(z_L) \prod_{l=0}^{L-1} p_\theta(z_l \mid z_{l+1})$. In our case, we have $z_L \sim \mathcal{N}(0, I)$, $[z_l|z_{l+1}] \sim \mathcal{N}(\mu_l(d_l(p_l(z_{l+1}))), \sigma_l^2(d_l(p_l(z_{l+1}))))$, $l = 0, ..., L-1$. where $\mu_l()$ and $\sigma_l^2()$ are the mean vector and the diagonal variance-covariance matrix of z_l respectively, and they are functions of $d_l(p_l(z_{l+1}))$ where d_l are deterministic layers and p_l are projection layer to preserve dimensionality. d_l is defined as two subsequent *conv*2*d* layers with *GeLU* [13] activation functions and skip connection. p_l is a linear layer with subsequent *transpose_conv*2*d*. μ_l and σ_l are a pair of *conv*2*d* and *linear* layers to project to dimensionality of z_l. Then, $z_l = \mu_l(d_l(p_l(z_{l+1}))) + \sigma_l(d_l(p_l(z_{l+1}))) \otimes \epsilon_l$ where $\epsilon_l \sim \mathcal{N}(0, I_{d_l})$. The final deterministic block o_0 is a *transpose_conv*2*d* layer projecting to the desired dimensionality of x. The range of x is bounded by $tanh()$.

Table 4 illustrates a specification with $L = 3$ latent layers, latent dimensions $d_3 = 32$, $d_2 = 64$, $d_1 = 128$ for z_3, z_2, z_1, respectively, and $n_f = 64$ channels.

l	operation	dimensions
3	$z_3 \sim N(0, I_{d_3})$	$[n, d_3, 1, 1]$
2	$z_{3,p} = p_2(z_3)$	$[n, n_f, 16, 16]$
2	$z_{3,d} = d_2(z_{3,p})$	$[n, n_f, 16, 16]$
2	$z_2 = \mu_2(z_{3,d}) + \sigma_2(z_{3,d}) \otimes \epsilon_2$	$[n, d_2, 1, 1]$
1	$z_{2,p} = p_1(z_2)$	$[n, n_f, 16, 16]$
1	$z_{2,d} = d_1(z_{2,p}) + z_{3,d}$	$[n, n_f, 16, 16]$
1	$z_1 = \mu_1(z_{2,d}) + \sigma_1(z_{2,d}) \otimes \epsilon_1$	$[n, d_1, 1, 1]$
0	$z_{1,p} = p_0(z_1)$	$[n, n_f, 16, 16]$
0	$z_{1,d} = d_0(z_{1,p}) + z_{2,d}$	$[n, n_f, 16, 16]$
0	$x = tanh(o_0(z_{1,d}))$	$[n, 3, 32, 32]$

Table 4: Specification of multi-layer generator model with $L = 3$ layers, latent dimensions $d_3 = 32$, $d_2 = 64$, $d_1 = 128$ for z_3, z_2, z_1, respectively, and $n_f = 64$ channels.

7.3 Training of baselines

For ladder variational autoencoder [32], the generator model is defined in Table 4. The training follows the one outlined in [32]. We train the model with $T = 3 \times 10^5$ parameter updates optimized by Adam [18].

For GLO [3] and ABP [11], our model in Table 4 was reduced to a single-layer variational autoencoder.

For GLO, we used a re-implementation[4] in PyTorch. As outlined in [3], after training the model, the inferred latent vectors, z, were used to fit a multivariate Gaussian distribution from which z was drawn for sampling. The hyperparameters are as follows: $code\_dim = 128$, $n\_pca = 64 * 64 * 3 * 2$, $loss = l2$.

For ABP, 40 steps of persistent Markov Chains were used. The hyper-parameters are as follows: 40 MCMC steps, Langevin discretization step size of 0.3, $\sigma = 0.3$, Adam [18] optimizer.

For Glow [21], the model was trained using the official code[5] with our datasets and the evaluation was performed with our implementation of the Frchet Inception Distance (FID) [14] with Inception v3 classifier [33] on $40,000$ generated example. The hyperparameters are as follows: $dal = 0$, $n\_batch\_train = 64$, $optimizer = adamax$, $n\_levels = 3$, $width = 512$, $depth = 16$, $n\_bits\_x = 8$, $learntop = False$, $flow\_coupling = 0$.

[4] `https://github.com/tneumann/minimal_glo`
[5] `https://github.com/openai/glow`