Neural MCMC via Adversarial Training

Zehao Yu
ShanghaiTech
yuzh@shanghaitech.edu.cn

Abstract

Our goal is to sample data from some high-dimensional distribution. We don't know the exact formulation of the distribution and what we have is some samples from this distrubution. Directly sampling from original data space is inefficient since high-dimensional data naturally lies in some low-dimensional manifolds. To address this problem, we propose Neural MCMC, a new method for drawing samples from high-dimensional distribution. We first learn the latent representation of the data using the auto-encoder framework without prior setting of the latent space. Then we train a MCMC kernels parametrized with a deep neural network to directly sample latent representation from the latent space via adversarial training. To avoid some trival solution and decrease the autocorrelation in MCMC kernels, we further train a pairwise discriminator to distinguish correlated samples of the consecutive MCMC outputs. With the sampled latent representations, we can decode them to original data space. We evaluate our method on three public datasets and show that our method can efficiently explore the target distribution and generate realistic results.

I. Introduction

Markov chain Monte Carlo (MCMC) has become a popular tool for many problems in the biological, natural, physical sciences, artificial intelligence and machine learning. The goal of MCMC is drawing samples $x \sim p(x|D)$ from a target posterior distribution given an observed dataset D using a Markov chain, and with those samples we can use Monte Carlo methods to efficiently approximate some desired quantities.

Over the last few decades, a variety of MCMC methods have been proposed to efficiently explore the posterior, such as Hamiltonian Monte Carlo (HMC) [18] and its Reimann manifold variants [6]. These samplers is based on a potential energy function in terms of the target posterior distribution. It then use some continuous dynamics to explore the energy landscape. However, when scale to large datasets, these methods inherently converge slowly in complex, high-dimensional models since it needs significant computational burden [15]. Recently, instead of using full dataset samplers, stochastic gradient variants of such continuous-dynamic samplers use data subsamples (mini-batches) to compute gradients of the target distribution and have shown promising results in scaling the aforementioned methods to large datasets [15]. Nevertheless, they still sample data point in the original high-dimensional data space and thus neglect the nature that high-dimensional data lies in some low-dimensional manifolds [7].

Recent progress in deep learning has vastly advanced in the field of representation learning and variational inference. Restricted Bltzman Machine (RBM) [17] and auto-encoder variants [11], [24] learn the low-dimensional representation of data via a reconstruction loss. Denoising auto-encoder [24] and generative stochastic networks (GSN)[2] learn a reconstruct distribution $P(X|\tilde{X})$ to recover clean data X from a corrupted data $\tilde{X} \sim P(\tilde{X}|X)$, where $P(\tilde{X}|X)$ is a predefined method of corruption. It then recontruct the data density $P(X)$ via Bayes relu. However, in order to generate samples, such method need to interatively reconstruction from corrupted data, corrupt the reconstructed data, then recontruct next sample, which is time-consuming. Since the reconstructed sample tend to be similar to the previous one, these methods struggle to "move between the modes" of the dataset.

In the other way, variational auto-Encoder (VAEs) [12] and generative adversarial networks (GANs) [8], [1] learn a latent variable model capable of generating data from a complex distributions. These method use a prior distribution $P(Z)$ or $Q(Z|X)$ of the latent space, and learn a mapping function $P(X|Z)$ from latent space to data space $P(X)$. $Q(Z|X)$, $P(X|Z)$ are encoding and decoding (generating) process parametrized by a neural networks. Although these methods can be used to generate realistic results [3], they rely on the choise of prior distrubtion $P(Z|X)$.

To address this limitations, we introduce Neural

MCMC, a new method for drawing samples from high-dimensional distribution. We learn the latent representation of the data using the auto-encoder framework without prior setting of the latent space. Then we train a MCMC kernels parametrized with a deep neural network to sample latent representation from the latent space directly via adversarial training. With the sampled latent representations, we can finally decode them to the original data space.

Different from previous methods that sample from the original high-dimensional data space [21], [18], [6], our method samples from the learned low-dimensional latent representation and thus can performance more efficiently and effectively. We also extend the pairwise discriminator proposed in [21] to the decrease the autocorrelation in MCMC kernels. Different from [21] which discriminate correlated samples in original data space, we distinguish correlated samples in the latent space. As a result, we can use a samller discriminator network to speed up training. Since the MCMC kernels in trained to move between different modes in the latent space, the resulting sampling process is much more efficient.

We extensively evaluate our method on three public datasets: MNIST [13], Fashion-MNIST [26] and CelebA [14]. Our experiments show that our proposed method can efficiently explore the target distribution and generate realistic results.

II. Related Work

A. Markov Chain Monte Carlo

The goal of MCMC is drawing samples $x \sim p(x|D)$ from a target posterior distribution given an observed dataset D using a Markov chain. With those samples we can use Monte Carlo methods to efficiently approximate some desired quantities. Metropolis-Hastings (MH) algorithm [9] proceeds by randomly attempting to move about the sample space via a acceptance ratio. Gibbs sampling [5] generates an instance from the distribution of each variable in turn, conditional on the current values of the other variables. Hamiltonian Monte Carlo (HMC) [18] and its Reimann manifold variants [6] is based on a potential energy function in terms of the target posterior distribution and use some continuous dynamics to efficiently explore the energy landscape. Recently, stochastic gradient MCMC (SGMCMC) [15] is proposed to scale the aforementioned methods to large datasets. Instead of using full dataset samplers, SGMCMC use data subsamples (minibatches) to compute the gradient of the target posterior. Different with these methods, we use a nueral network to model the MCMC kernels then train it via an adversarial loss.

B. Dimension Reduction

The aim of dimensionality reduction is to preserve as much of the significant structure of the high-dimensional data as possible [16]. Principal component analysis (PCA) [25] is probably the most popular method for dimension reduction. It fits a low-dimensional affine subspace to a set of data points in a high-dimensional space. Classical scaling [23] finds the linear transformation that preserves the high-dimensional pairwise distances in low-dimensional space. Locally Linear Embedding (LLE) [20] is a non-linear method that computes low-dimensional, neighborhood-preserving embeddings of high-dimensional inputs. Stochastic Neighbor Embedding (SNE) [10] and t-SNE [16] aim to preserve pairwise conditional probabilities converted from high-dimensional Euclidean distances. In contrary, auto-encoder [11] use neural networks to learn the low-dimensional representation of data via a reconstruction loss. In this project, we also use auto-encoder to learn latent representation of high-dimensional datasets since the decoder can be used as a generative network.

C. Neural Networks for Posterior Approximation

Using neural networks to approximate the posterior is a general idea when directly inference is intractable. Denoising auto-encoder [24] and generative stochastic networks (GSN)[2] learn the transition operator of a Markov chain $P(X|\tilde{X})$ of recover clean data X from a corrupted data $\tilde{X} \sim P(\tilde{X}|X)$, where $P(\tilde{X}|X)$ is a predefined method of corruption. Therefore the data density $P(X)$ can be recontructed with Bayes relu. Variational auto-Encoder (VAE) [12]) use neural networks to efficient approximate posterior $P(Z|X)$ via a reconstruction loss and a regularization loss. The regularization loss encourages $P(Z|X)$ to be compatible with some prior $P(Z)$.

In contrary, generative adversarial networks (GANs) [8], [1] learn to directly generate data from a simple prior distribution to a complex distribution. It use a two player minimax game to train the generaing function G. The generator G generates samples from a noise variable, and a discriminator D is trained to distinguish between "fake" samples from generator and "real" samples from a given dataset. Then the generator is trained to "fool" the discriminator which means the generated samples is realistic in the view of discriminator. In this project, we focus on approximte the latent distribution using a MCMC kernel parametrized by a neural network. We use the adversarial loss in GANs to train the MCMC kernel so as to efficiently explore the latent space.

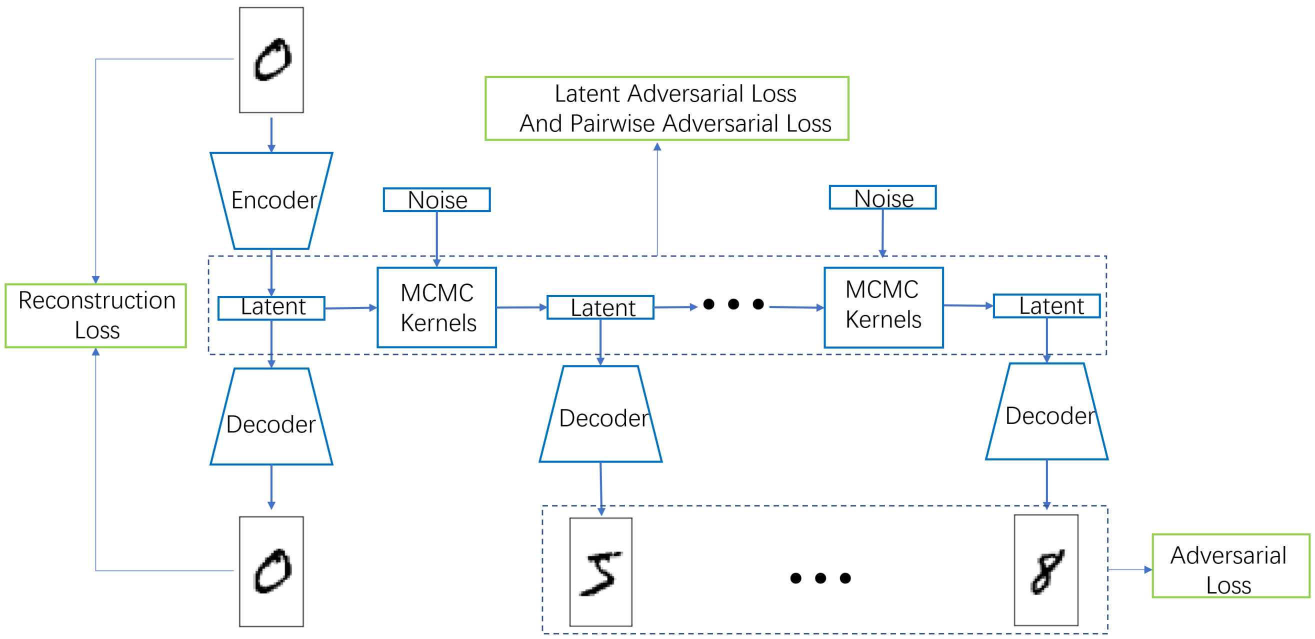


Fig. 1. Our proposed neural MCMC framework. We use an auto-encoder to learn the latent representation of the data space. We then use a neural network to model the MCMC kernels which take a latent code and a noise code as input and outputs the next latent code. Then we decode to latent code to original data space. The auto-encoder is trained with a reconstruction loss and the decoder (generator) is shared in every MCMC step and is also trained with a discriminator loss. The neural network parametrized MCMC kernels is trained with a latent representation discriminator loss to encourge the predicted latent code has the same distribution with input latent code, and a pairwise discriminator loss to distinguish correlated samples of the consecutive MCMC outputs and thus decrease autocorrelation.

III. Method

Our goal is to sample data from some high-dimensional distribution. We don't know the exact formulation of the distribution and what we have is some samples from this distrubution. Directly sampling from original data space is inefficient since high-dimensional data naturally lies in some low-dimensional manifolds [7]. To address this problem, we propose to learn the latent representation of the data using the auto-encoder framework without prior setting of the latent space. Then we train a MCMC kernels parametrized with a deep neural network to directly sample latent representation from the latent space via adversarial training. To avoid some trival solution and decrease the autocorrelation in MCMC kernels, we further train a pairwise discriminator to distinguish correlated samples of the consecutive MCMC outputs. With the sampled latent representations, we can decode them to original data space. Figure 1 shows the overall pipeline of our method.

A. Latent Representation

We are given a set of sample data points $X = \{x_1, x_2, ..., x_N\} \subset \mathbb{R}^D$ from some unknown high-dimentional distribution. We want to find a low-dimensional representation $Z = \{z_1, z_2, ..., z_N\} \subset \mathbb{R}^d$ of the original data points, where $d << D$. If X lie in some linear subspace, principle component analysis (PCA) could be directly applied, but it's not often the case. We therefore use an auto-encoder framework to resolve this problem.

The auto-encoder if of the form

$$\begin{aligned} z &= Encoder(x) \\ \tilde{x} &= Decoder(z) \end{aligned} \tag{1}$$

where $Encoder$ and $Decoder$ is parametrized with a neural network. We want the reconstucted data $\tilde{x}$ is the same as the original data x, thus mean square error (MSE) and bianary cross entropy (BCE) could be used as the reconstruction loss depend on the data type.

We note that our method is not restricted to auto-encoder, other methods for finding the low-dimensional latent space could be applied to. Although Non-linear Independent Components Estimation (NICE) [4] could transform the data to an independent latent variables, the transformed data is still in the high-dimensional space, while we focus on finding the low-dimensional representation.

B. Neural MCMC

Once we have the low-dimensional latent representation $P(Z)$ of the data space, we can sample the latent representation then decode it to original data space. We can construct a Markov chain to approximate $P(Z)$, then we can run this chain in order to draw samples from $P(Z)$. However, how to construct a chain with high mixing rate is a difficult problem. Suppose we want to use the Metropolish-Hasting algorithm, then how do we choose a proposal distribution that lead to high accept rate? While using other MCMC methods are possible, we use an alternative approach inspired by recent succeed in GANs and A-NICE-MC [4].

In A-NICE-MC [4], the Markov chain is treated as a generative model and is parametrized with neural network. The network take the current sample and a random noise as input and outputs the next sample. Then it use the idea of GAN to optimize to Markov chain (the implicit generative model G) so that the generated samples has the same distribution with the real data (which means it can fool the discriminator). However, it use the NICE [4] to model the Markov chain and sample the data in the original data space which neglect the nature that high-dimensional data lies in some low-dimensional manifolds.

Different from A-NICE-MC[21], we levarage the expressive power of neural network to model the Markov chain in the learned low-dimensional latent space. Specifically, we use a neural network to model the MCMC kernel $T(z_{t+1}|z_t)$:

$$v \sim p(v) \quad z_{t+1} = G(z_t, v) \tag{2}$$

Where $p(v)$ is some random noise. Since we want the generated z_1 looks "real" in the view of latent space $p(z)$, we optimize the neural MCMC kernel using the following objective as in GAN [1], [8]:

$$\min_G \max_D \mathbb{E}_{z\sim p(z)}\left[D(z)\right] - \mathbb{E}_{v\sim p(v),\tilde{z}\sim p(\tilde{z})}\left[D(G(\tilde{z}, v))\right] \tag{3}$$

where $p(\tilde{z})$ can be the latent representation $p(z)$ or some other prior distribution and is used as the initialization input for the MCMC kernel G.

The outputs latent code z_t of G at every step ($t >= 1$) is the "fake" samples for the discriminator D, the learned latent code from the dataset is the "real" samples. Different from A-NICE-MC[21] which takes several step to generate one sample, we take only one step to generate the next sample.

Besides using discrinator in latent space, we employ a discriminator D_{orig} in origin data space because we also want the decoded data looks "real". The optimization objective is the following:

$$\min_G \max_{D_{orig}} \mathbb{E}_{x\sim p(x)}\left[D_{orig}(x)\right] - \mathbb{E}_{v\sim p(v),z\sim p(z),\tilde{z}\sim G(z,v)}\left[D_{orig}(Decoder(\tilde{z})\right]. \tag{4}$$

The proposed neural MCMC is shown in Figure 2.

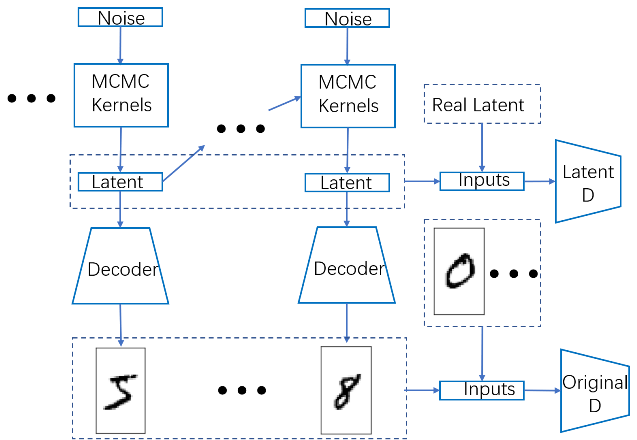


Fig. 2. Our neural MCMC Kernel. It is trained with two discriminator. Latent D encourges generated latent code to be same as learned latent code. Original D encourges generated latent code can be used to decode relealistic results.

C.Consecutive correlated samples discriminator

A trival solution for the previous introduced MCMC kernel will be an identity function that simply copy the input latent code z and ignore the noise part v. To avoid this trival solution, we add another discriminator $\tilde{D}$ to distinguish consecutive correlated samples as in [21]. To be specific, we use two samples (z_1, z_2) as input to $\tilde{D}$. Consecutive outputs $(\tilde{z}_1, \tilde{z}_2)$ from the generator is the "fake" samples and randomly sampled (z_1, z_2) from the dataset is the "real" samples. The pairwise discriminator $\tilde{D}$ should distinguishes correlated samples and therefore encourages the MCMC kernel (the generator G) to generate independent samples and thus can move between modes in latent space.

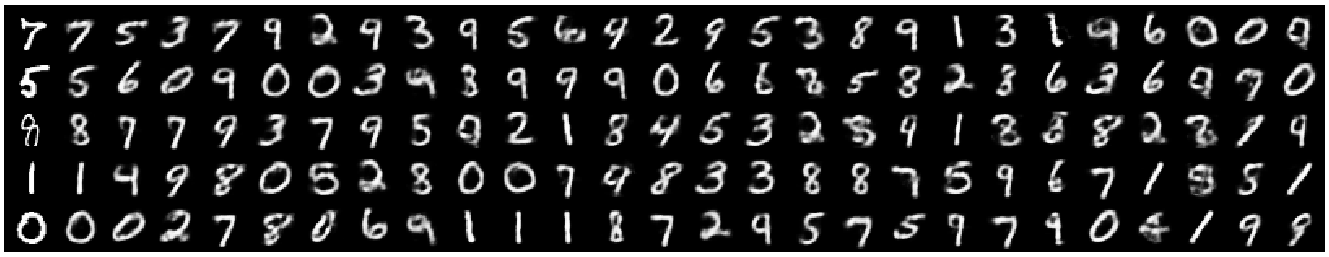
Different from A-NICE-MC [21] which use $\tilde{D}$ to distinguish similar samples in the original data space, we distinguish correlated samples in latent space, thus we can use a samller discriminator network to speed up training. Since the MCMC kernels in trained to move between different modes in the latent space, the resulting sampling process is much more efficient.

IV. Experiments

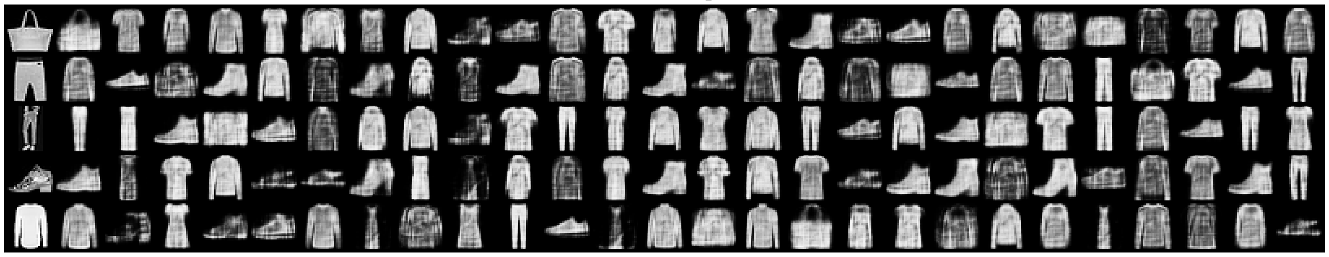
In this section, we conducate experiments to evaluate the performance of the proposed method on three public datasets: MNIST [13], Fashion-MNIST [26] and CelebA [14]. We first introduce the datasets we used, then describe the implementation details and show the experiment results.

A.Datasets

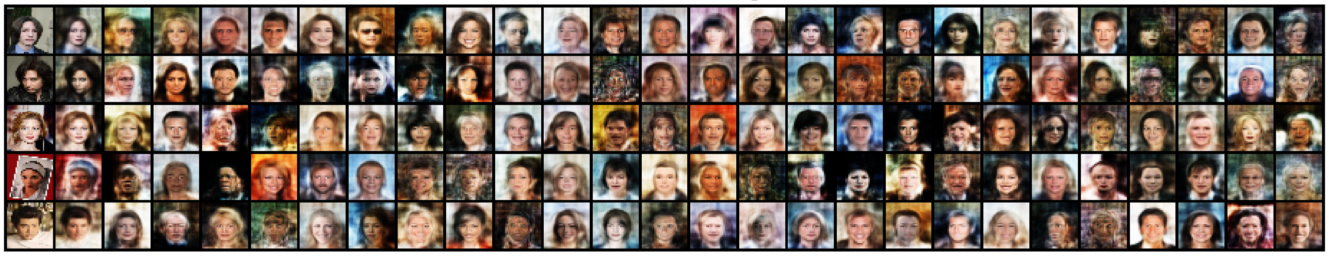
The MNIST dataset [13] contains handwritten digits. It has a training set of 60,000 examples, and a test set



MNIST samples



Fashion-MNIST samples



CelebA samples

Fig. 3. Our neural MCMC samples on three datasets. First two columns show the input images and reconstructed images respectively. From third column to last column show the consecutive samples from our neural MCMC using latent code of the image in first column as initializing input.

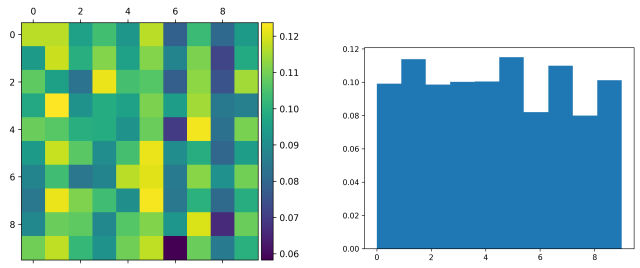


Fig. 4. Left figure shows the transition matrix of our neural MCMC between 10 digits of MNIST dataset. The right figure shows the stationary distribution of 10 digits.

of 10,000 examples. The digits have been size-normalized and centered in a 28x28 image.

Fashion-MNIST [26] is a MNIST-like fashion product database consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes (*i.e.*, Dress, Coat). It has the same image size and structure of training and testing splits with the original MNIST dataset. This dataset is more challenging then the original MNIST dataset.

CelebFaces Attributes Dataset (CelebA) [14] is a large-scale face attributes dataset with more than 200K celebrity images including 10,177 number of identities. The images in this dataset cover large pose variations and background clutter. We use the aligned and cropped images in this dataset and resize to 32x32.

B.Implementation Details

We implement our method with PyTorch [19]. The encoder , decoder, MCMC kernel and the discriminators are parametrized with multi-layers fully connected networks. We use leaky relu [27] as activation function and BCE loss as reconstruction loss. RMSProp optimizer [22] is used for optimization.

We adopt two stages training strategy. We first train the encoder, decoder and image discriminator for 500 epochs with batch size 128. Then we train the MCMC kernel, latent discriminator and pairwise discriminator for another 500 epochs. The learning rate in both stages is set to 0.00005.

C.Results

We visualize the samples from our neural MCMC on three datasets in Figure 3. As one can see, our method can easily move between the mode and thus can efficiently explore the target posterior.

For quantitative evaluation, we train a classifier on the MNIST dataset and use it to classify the generated samples. We show the transition matrix and the stationary distribution in Figure 4. Well, the results show that the transition matrix and stationary distribution is approximately uniform which further reveal the effectiveness and effiency of our method.

V. Conclusion

In this project, we propose Neural MCMC, a new method for drawing samples from high-dimensional distribution. We first learn the latent representation of the data using the auto-encoder framework. Then we train a MCMC kernels parametrized with a deep neural network to directly sample latent representation from the latent space via adversarial training. To avoid some trival solution and decrease the autocorrelation in MCMC kernels, we further train a pairwise discriminator to distinguish correlated samples of the consecutive MCMC outputs. With the sampled latent representations, we can decode them to original data space. We evaluate our method on three public datasets and show that our method can efficiently explore the target distribution and generate realistic results.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[2] Y. Bengio, E. Thibodeau-Laufer, and J. Yosinski. Deep generative stochastic networks trainable by backprop. Technical Report arXiv:1306.1091, Universite de Montreal, 2013.

[3] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[4] L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[5] A. E. Gelfand and A. F. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.

[6] M. Girolami and B. Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[7] Y. Goldberg, A. Zakai, D. Kushnir, and Y. Ritov. Manifold learning: The price of normalization. *Journal of Machine Learning Research*, 9(Aug):1909–1939, 2008.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[10] G. E. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.

[11] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[12] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[15] Y.-A. Ma, T. Chen, and E. Fox. A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems*, pages 2917–2925, 2015.

[16] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[17] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[18] R. M. Neal et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[20] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[21] J. Song, S. Zhao, and S. Ermon. A-nice-mc: Adversarial training for mcmc. *arXiv preprint arXiv:1706.07561*, 2017.

[22] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[23] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[24] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408, 2010.

[25] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[26] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[27] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.