Learning to Sample Using Stein Discrepancy

Dilin Wang **Yihao Feng** **Qiang Liu**
Department of Computer Science
Dartmouth College
Hanover, NH 03755
{dilin.wang.gr, yihao.feng.gr, qiang.liu}@dartmouth.edu

Abstract

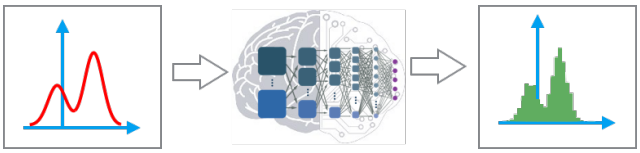
We propose a simple algorithm to train stochastic neural networks to draw samples from given target distributions for probabilistic inference. Our method is based on iteratively adjusting the neural network parameters so that the output changes along a Stein variational gradient [1] that maximumly decreases the KL divergence with the target distribution. Our method works for any target distribution specified by their unnormalized density function, and can train any black-box architectures that are differentiable in terms of the parameters we want to adapt. By allowing to "learn to draw samples", our method opens a host of applications. We present two examples in this paper: 1) we propose an *amortized MLE* method for training deep energy model, where a neural sampler is adaptively trained to approximate the likelihood function. Our method mimics an adversarial game between the deep energy model and the neural sampler, and obtains realistic-looking images competitive with the state-of-the-art results. 2) by treating stochastic gradient Langevin dynamics as a black-box sampler, we train it to automatically adjust its learning rate to maximize its convergence speed, and get better performances than the hand-designed learning rate schemes.

1 Introduction

Modern machine learning increasingly relies on highly complex probabilistic models to reason about uncertainty. A key computational challenge is to develop efficient inference techniques to approximate, or draw samples from complex distributions. Currently, most inference methods, including MCMC and variational inference, are hand-designed by researchers or domain experts. This makes it difficult to fully optimize the choice of different methods and their parameters, and exploit the structures in the problems of interest in an automatic way. The hand-designed algorithm can also be inefficient when it requires to make fast inference repeatedly on a large number of different distributions with similar structures. This happens, for example, when we need to reason about a number of observed datasets in settings like online learning, or need fast inference as inner loops for other algorithms such as maximum likelihood training. Therefore, it is highly desirable to develop more intelligent probabilistic inference systems that can adaptively improve its own performance to fully optimize computational efficiency, and generalize to new tasks with similar structures. Specifically, we study the following problem:

Problem 1. *Given a distribution with density $p(x)$ specified up to the normalization constant, and a function $f_\eta(\xi)$ with parameter η and random input ξ, for which we only have assess to draws of the random input ξ (without knowing its true distribution q_ξ), and the output values of $f_\eta(\xi)$ and its derivative $\partial_\eta f_\eta(\xi)$ given η and ξ. We want to find an optimal parameter η so that the density of the random output variable $x = f_\eta(\xi)$ with $\xi \sim q_\xi$ matches closely with the target density $p(x)$.*

Because we have no assumption on the structure of f_η and the distribution of random input, we can not directly calculate actual distribution of the output random variable $x = f_\eta(\xi)$; this makes



Given distribution *Black-box neural sampler* *Samples*

Figure 1: Our methods "learn to draw samples", constructing black-box neural samplers for given distributions. It allows us to automatize the hyper-parameter tuning of Bayesian inference, speed up the inference inner loops of learning algorithms, and eventually replace hand-designed inference algorithms with more efficiently one that is trained on past tasks and is improved adaptively over time.

it difficult to solve Problem 1 using the traditional variational inference (VI) methods. Recall that traditional VI approximates $p(x)$ using simple proposal distributions $q_\eta(x)$ indexed by parameter η, and finds the optimal η by minimizing KL divergence $\mathrm{KL}(q_\eta \,||\, p) = \mathbb{E}_{q_\eta}[\log(q_\eta/p)]$, which requires to calculate the density $q_\eta(x)$ or its derivative (even when Monte Carlo gradient estimation and reparameterization trick [2] are applied) that is not available by our assumption.

In fact, it is this requirement of calculating $q_\eta(x)$ that has been the major constraint for the designing of state-of-the-art variational inference methods with rich approximation families; the recent successful algorithms [e.g., 3–5, to name only a few] have to handcraft special variational families to ensure the computational tractability of $q_\eta(x)$ and simultaneously obtain high approximation accuracy, which require substantial mathematical insights and research effects. One exception is a very recent paper [6] that avoids calculating $q_\eta(x)$ using an idea related to Stein discrepancy [7–10]. New methods that do not require to explicitly calculate $q_\eta(x)$ will significantly simplify the design and applications of VI methods, allowing practical users to focus more on choosing proposals that work best with their specific tasks. We will use the term *wild variational inference* to refer to new variants of variational methods that requires no tractability $q_\eta(x)$, to distinguish with the *black-box variational inference* [11] which refers to methods that work for generic target distributions $p(x)$ without significant model-by-model consideration (but still require to calculate the proposal density $q_\eta(x)$).

A similar problem also appears in importance sampling (IS), where it requires to calculate the IS proposal density $q(x)$ in order to calculate the importance weight $w(x) = p(x)/q(x)$. However, there exist methods that use no explicit information of $q(x)$. While, seemingly counter-intuitively, giving better asymptotic variance or converge rates than the typical IS that uses the proposal information [e.g., 12–15]. Discussions on this phenomenon dates back to O'Hagan [16], who argued that "Monte Carlo (that uses the proposal information) is fundamentally unsound" for violating the Likelihood Principle, and developed Bayesian Monte Carlo [17] as an example that uses no information on $q(x)$, yet gives better convergence rate than the typical Monte Carlo $O(n^{-1/2})$ rate [13]. Despite the substantial difference between IS and VI, these results intuitively suggest the possibility of developing efficient variational inference without calculating $q(x)$ explicitly.

2 Amortized SVGD

We approach Problem 1 by iteratively adjusting the network parameter η to make the network outputs mimic the dynamics of a recent Stein variational gradient descent (SVGD) method [1]. SVGD is a deterministic sampling method that iteratively updates a set of particles $\{x_i\}_{i=1}^n$ to approximate the target distribution $p(x)$ in the sense that $\sum_i f(x_i)/n \approx \mathbb{E}_p f$ for general test functions f. Each iteration of SVGD updates the current particles $\{x_i\}_{i=1}^n$ with a gradient-like update

$$x_i' = x_i + \epsilon \Delta x_i,$$

where ϵ is a small step size, and Δx_i is an optimal perturbation direction chosen to maximumly decrease the KL divergence between the distribution of the particles and the target distribution. The derivation in Liu and Wang [1] shows that Δx_i can be chosen to be

$$\Delta x_i = \hat{\mathbb{E}}_{x\sim\{x_i\}_{i=1}^n}[\nabla \log p(x) k(x, x_i) + \nabla_x k(x, x_i)], \tag{1}$$

where $\hat{\mathbb{E}}_{x\sim\{x_i\}_{i=1}^n}$ denotes the empirical averaging on the current particles $\{x_i\}_{i=1}^n$. The two terms in Δx_i play two different roles: the term with the gradient $\nabla_x \log p(x)$ drives the particles towards the high probability regions of $p(x)$, while the term with $\nabla_x k(x, x_i)$ serves as a repulsive force to

Algorithm 1 Amortized SVGD or KSD for Problem 1

for iteration t **do**

1. Draw random $\{\xi_i\}_{i=1}^n$, calculate $x_i = f_\eta(\xi_i)$, and the Stein variational gradient Δx_i in (1).
2. Update parameter η using (2) (for amortized SVGD), or (3) (for amortized KSD).

end for

encourage diversity; to see this, consider a stationary kernel $k(x, x') = k(x - x')$, then the second term reduces to $\hat{\mathbb{E}}_x \nabla_x k(x, x_i) = -\hat{\mathbb{E}}_x \nabla_{x_i} k(x, x_i)$, which can be treated as the negative gradient for minimizing the average similarity $\hat{\mathbb{E}}_x k(x, x_i)$ in terms of x_i.

Note that Δx_i reduces to the typical gradient $\nabla_x \log p(x_i)$ when there is only one single particle ($n = 1$), in which case SVGD reduces to the standard gradient ascent for maximizing $\log p(x)$ (i.e., maximum *a posteriori* (MAP)).

We propose to solve Problem 1 by "amortizing" SVGD to make the output of network $f_\eta(\xi)$ mimic the SVGD dynamics. This is done by iteratively adjusting the network parameter η such that the network outputs $x_i = f_\eta(\xi_i)$, $i = 1, \ldots, n$, change along with the optimal direction Δx_i given by SVGD. To be specific, we should update η via

$$\eta \leftarrow \arg\min_\eta \sum_{i=1}^n ||f_\eta(\xi_i) - x_i - \epsilon \Delta x_i||_2^2.$$

If we approximately solve this optimization with a single step of gradient update, we get a simpler update of η:

$$\eta \leftarrow \eta + \epsilon \sum_i \partial_\eta f_\eta(\xi_i) \Delta x_i, \tag{2}$$

which can be intuitively interpreted as a form of chain rule that *back-propagates the SVGD gradient to the network parameter* η. In fact, when we have only one particle, (2) reduces to the standard gradient ascent for $\max_\eta \log p(f_\eta(\xi))$, in which f_η is trained to "learn to optimize" [e.g., 18], instead of "learn to sample" $p(x)$. See Algorithm 1 for our full algorithm.

3 Amortized KSD

Amortized SVGD can be treated as minimizing the KL divergence objective, but avoids the need for explicitly evaluating $q(x)$ with the help of SVGD. Here we provide an alternative method based on minimizing a different kernelized Stein discrepancy (KSD) objective, which, thanks to its special form, can be minimized using typical gradient descent without needing to estimate $q(x)$ explicitly.

For two positive differentiable densities p and q on $\mathbb{R}^d$, their KSD $\mathbb{D}(q \,||\, p)$ is defined via

$$\mathbb{D}^2(q \,||\, p) = \mathbb{E}_{x,x' \sim q}[\kappa_p(x, x')],$$

where x, x' are i.i.d. draws from q and $\kappa_p(x, x')$ is a positive definite kernel related to p via

$$\kappa_p(x, x') = \nabla_x \log p(x) \nabla_x \log p(x') k(x, x') + \nabla_x \log p(x) \nabla_{x'} k(x, x') + \nabla_x \log p(x') \nabla_x k(x, x') + \nabla_x \cdot (\nabla_{x'} k(x, x')).$$

It can be shown that $\mathbb{D}(q \,||\, p) = 0$ if and only if $p = q$ under certain regularity conditions [1, 10].

Taking q_η to be the density of the random output $x = f_\eta(\xi)$ when $\xi \sim q_\xi$, then we want to find η to minimize $\mathbb{D}(q_\eta \,||\, p)$. With i.i.d. drawing ξ_i from q_ξ, we can approximate $\mathbb{D}^2(q_\eta \,||\, p)$ unbiasedly with a U-statistics:

$$\hat{\mathbb{D}}^2(q_\eta \,||\, p) = \frac{1}{n(n-1)} \sum_{i \neq j} \kappa_p(f_\eta(\xi_i),\ f_\eta(\xi_j)),$$

for which a standard gradient descent can be derived for optimizing η:

$$\eta \leftarrow \eta - \epsilon \frac{2}{n(n-1)} \sum_{i \neq j} \partial_\eta f_\eta(\xi_i) \nabla_{x_i} \kappa_p(x_i, x_j), \quad \text{where} \quad x_i = f_\eta(\xi_i). \tag{3}$$

Algorithm 2 Amortized MLE as Generative Adversarial Learning

Goal: MLE training for energy model $p(x|\theta) = \exp(-\phi(x,\theta) - \Phi(\theta))$.
Initialize η and θ.
for iteration t **do**
  Updating η: Draw $\xi_i \sim q_\xi$, $x_i = f_\eta(\xi_i)$; update η using (2) with $p(x) = p(x|\theta)$. Repeat several times when needed.
  Updating θ: Draw a mini-batch of observed data $\{x_{i,obs}\}$, and simulated data $x_i = f_\eta(\xi_i)$, update θ by

$$\theta \leftarrow \theta - \hat{\mathbb{E}}_{obs}[\nabla_\theta \phi(x,\theta)] + \hat{\mathbb{E}}_\eta[\nabla_\theta \phi(x,\theta)].$$

end for

This allows us to solve Problem 1 for wild variational inference by directly minimizing η with standard (stochastic) gradient descent.

However, (3) does not have the nice property of reducing to "learning to optimize" like (2). (3) is also less natural because (3) involves the Hessian matrix $\nabla_x^2 \log p(x)$. This makes it less convenient to implement (3), although it can be made easy with automatic differentiation tools. We also find in practice that (3) turns to be unstable sometimes, possibly because $\mathbb{D}(q \mid\mid p)$ forms a weaker discrepancy measure than KL divergence. More studies are needed to understand and improve amortized KSD.

4 Applications

Our method allows us to design efficient approximate sampling methods adaptively and automatically, and enables a host of novel applications. In this paper, we exploit two particular examples: (1) amortized MLE for training deep generative models, and (2) automatic hyper-parameter tuning for Bayesian inference.

4.1 Amortized MLE for Generative Adversarial Training

Maximum likelihood estimator (MLE) provides a fundamental approach for learning probabilistic models from data, but can be computationally prohibitive on distributions for which drawing samples or computing likelihood is intractable due to the normalization constant. Traditional methods such as MCMC-MLE uses hand-designed methods (e.g., MCMC) to approximate the intractable term but do not work efficiently in practice. We propose to adaptively train a generative neural network to draw samples from the distribution, which not only provides computational advantage, and also allow us to generate realistic-looking images competitive with, or better than the state-of-the-art generative adversarial networks (GAN) [19, 20] (see Figure 2-6).

To be specific, denote by $\{x_{i,obs}\}$ a set of observed data. We consider the maximum likelihood training of energy-based models of form

$$p(x|\theta) \propto \frac{1}{Z(\theta)} \exp(-\phi(x,\theta)), \qquad Z(\theta) = \int \exp(-\phi(x,\theta))dx,$$

where $\phi(x;\ \theta)$ is an energy function for x indexed by parameter θ and $Z(\theta)$ is the normalization constant. The maximum likelihood estimator of θ is based on maximizing the log likelihood function,

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p(x_{i,obs}|\theta),$$

whose gradient is

$$\nabla_\theta L(\theta) = -\hat{\mathbb{E}}_{obs}[\partial_\theta \phi(x;\theta)] + \mathbb{E}_\theta[\partial_\theta \phi(x;\theta)],$$

where $\hat{\mathbb{E}}_{obs}[\cdot]$ and $\mathbb{E}_\theta[\cdot]$ denote the empirical averaging on the observed data $\{x_{i,obs}\}$ and the expectation under model $p(x|\theta)$. The key computational difficulty is to approximate the model averaging $\mathbb{E}_\theta[\cdot]$. To address this problem, we use a generative neural network $x = f_\eta(\xi)$ trained by Algorithm 1 to approximately sample from $p(x|\theta)$,

$$\hat{\nabla}_\theta L(\theta) = -\hat{\mathbb{E}}_{obs}[\partial_\theta \phi(x;\theta)] + \hat{\mathbb{E}}_\eta[\partial_\theta \phi(x;\theta)],$$

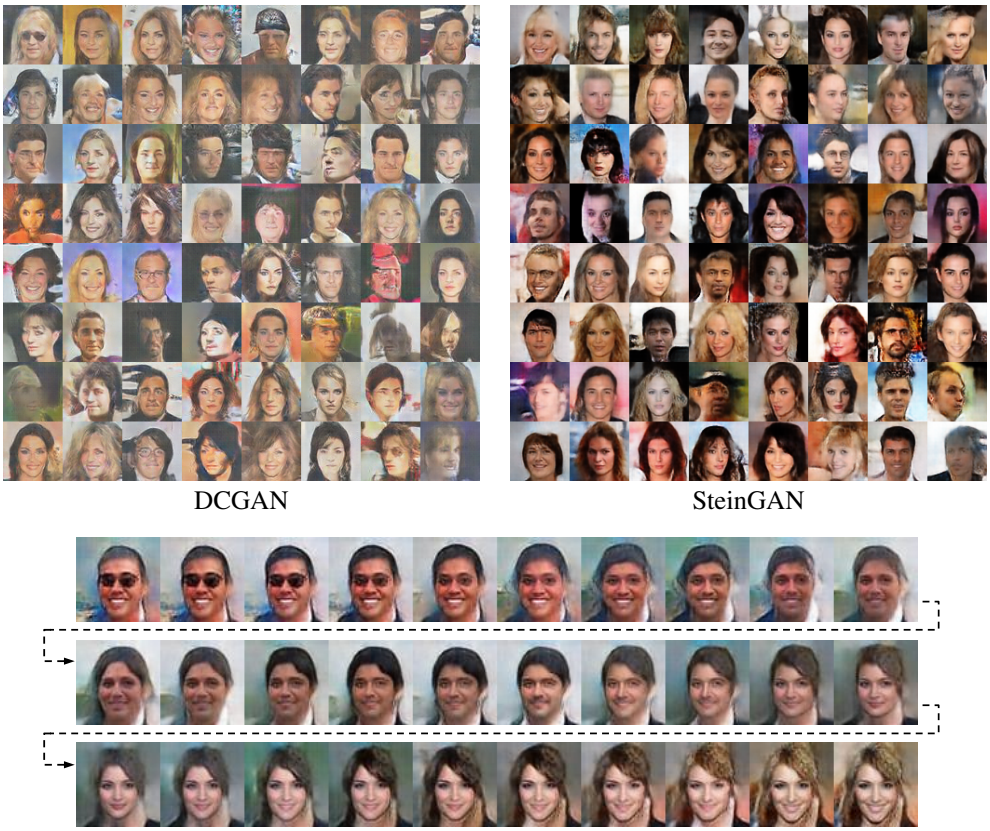


Figure 2: Results on CelebA. Upper: Images generated by DCGAN and our SteinGAN. Lower: images generated by SteinGAN when performing a random walk $\xi \leftarrow \xi + 0.01 \times \text{Uniform}([-1, 1])$ on the random input ξ.

where $\hat{\mathbb{E}}_\eta$ denotes the empirical averaging on $\{x_i\}$ where $x_i = f_\eta(\xi_i)$, $\{\xi_i\} \sim q_\xi$. As θ is updated by gradient ascent, η is successively updated via Algorithm 1 to *follow* $p(x|\theta)$. See Algorithm 2.

We call our method *SteinGAN*, because it can be intuitively interpreted as an adversarial game between the generative network f_η and the energy model $p(x|\theta)$ which serves as a discriminator: The MLE gradient update of p_η effectively decreases the energy of the training data and increases the energy of the simulated data from f_η, while the SVGD update of f_η decreases the energy of simulated data to fit better with $p(x|\theta)$. Meanwhile, our procedure is still a principled maximum likelihood procedure and can be more stable than the original GAN [19] that attends to find a Nash equilibrium. Compared with the traditional MCMC-MLE methods, we *amortize the sampler as we train*, which gives much faster speed and provides a high quality generator to generate realistic images simultaneously.

We tested our SteinGAN on four datasets, MNIST, CIFAR10, CelebA [21], and Large-scale Scene Understanding (LSUN)[22], on which we find our method tends to generate realistic-looking images competitive with DCGAN [20] (see Figure 2-Figure 4). In particular, we find we generate better images than DCGAN on CelebA (Figure 2), and our simulated CIFAR10 images achieves better testing classification accuracy when used as training data (see Figure 4). See Appendix A for more information. Our code is available at `https://github.com/DartML/SteinGAN`.

4.2 Hyper-parameter Optimization for Bayesian Inference

By treating the existing MCMC or variational methods as black-box procedures, we can apply our method to adaptively tune the hyper-parameters in these methods. This allows us to fully optimize the potential of existing Bayesian inference methods and also decreases the need of hyper-parameter tuning by human experts. As an example, we applied our method to adaptively learn the optimal learning rate for stochastic gradient Langevin dynamics (SGLD) [23], and find that significantly

outperforms hand-designed learning rates such as Adagrad [24] and RMSprop (See Figure 5). See Appendix B for more information.

5 Conclusion

We provide efficient algorithms for training neural samplers, together with a new SteinGAN method for generative adversarial training, and a hyperparameter optimization method for stochastic gradient Langevin dynamics. Future directions involve more applications and theoretical understandings for training neural samplers.

References

[1] Q. Liu and D. Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *arXiv preprint arXiv:1608.04471*, 2016.

[2] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[3] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[4] D. Tran, R. Ranganath, and D. M. Blei. Variational gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.

[5] R. Ranganath, D. Tran, and D. M. Blei. Hierarchical variational models. *arXiv preprint arXiv:1511.02386*, 2015.

[6] R. Ranganath, J. Altosaar, D. Tran, and D. Blei. Operator variational inference. 2016.

[7] J. Gorham and L. Mackey. Measuring sample quality with Stein's method. In *Advances in Neural Information Processing Systems (NIPS)*, pages 226–234, 2015.

[8] Q. Liu, J. D. Lee, and M. I. Jordan. A kernelized Stein discrepancy for goodness-of-fit tests. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

[9] C. J. Oates, M. Girolami, and N. Chopin. Control functionals for Monte Carlo integration. *Journal of the Royal Statistical Society, Series B*, 2017.

[10] K. Chwialkowski, H. Strathmann, and A. Gretton. A kernel test of goodness of fit. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

[11] R. Ranganath, S. Gerrish, and D. M. Blei. Black box variational inference. In *AISTATS*, pages 814–822, 2014.

[12] Q. Liu and J. D. Lee. Black-box importance sampling. *https://arxiv.org/abs/1610.05247*, 2016.

[13] F.-X. Briol, C. Oates, M. Girolami, M. A. Osborne, D. Sejdinovic, et al. Probabilistic integration: A role for statisticians in numerical analysis? *arXiv preprint http://arxiv.org/abs/1512.00933*, 2015.

[14] M. Henmi, R. Yoshida, and S. Eguchi. Importance sampling via the estimated sampler. *Biometrika*, 94(4): 985–991, 2007.

[15] B. Delyon and F. Portier. Integral approximation by kernel smoothing. *arXiv preprint arXiv:1409.0733*, 2014.

[16] A. O'Hagan. Monte Carlo is fundamentally unsound. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 36(2/3):247–249, 1987.

[17] A. O'Hagan. Bayes–hermite quadrature. *Journal of statistical planning and inference*, 29(3):245–260, 1991.

[18] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. *arXiv preprint arXiv:1606.04474*, 2016.

[19] 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.

[20] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[21] 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.

[22] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[23] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2011.

[24] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[25] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[26] D. Wang and Q. Liu. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016.

[27] Q. Liu and Y. Feng. Two methods for wild variational inference. *arXiv preprint arXiv:1612.00081*, 2016.

A Empirical Results of SteinGAN

In order to generate realistic-looking images, we define our energy model based on an autoencoder:

$$p(x|\theta) \propto \exp(-||x - \mathrm{D}(\mathrm{E}(x))||), \tag{4}$$

where x denotes the image. This choice is motivated by Energy-based GAN [25] in which the autoencoder loss is used as a discriminator but without a probabilistic interpretation. We assume $f_\eta(\xi)$ to be neural networks whose input ξ is a 100-dimensional random vector drawn by $\mathrm{Uniform}([-1, 1])$. The positive definite kernel is defined by the RBF kernel on hidden layer of the autoencoder, that is,

$$k(x, x') = \exp(-\frac{1}{h^2}||\mathrm{E}(x) - \mathrm{E}(x')||^2).$$

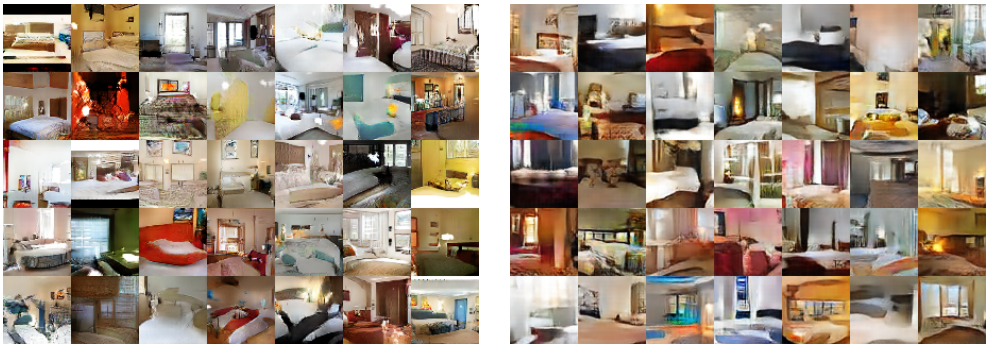
The kernel can act as a repulsive force to enforce diversity on the generated samples via the term $\nabla_x k(x, x')$ in (1). This is similar to the heuristic "repelling regularizer" in [25], but derived in a principled way. We take the bandwidth to be $h = 0.5 \times \mathrm{med}$, where med is the median of distances between $\mathrm{E}(x)$ on the observed images $\{x_{i,obs}\}_i$.

For MNIST and CIFAR-10, each image x also has a discrete label y, and we train a joint model on (x, y):

$$p(x, y|\theta) \propto \exp\{-||x - \mathrm{D}(\mathrm{E}(x))|| - \max[m,\ \ \sigma(y,\ \mathrm{E}(x))]\}, \tag{5}$$

where $\sigma(\cdot, \cdot)$ is the cross entropy loss function of a fully connected output layer. In this case, the generative network will first draw a y randomly according to the empirical counts in the dataset, and pass it into a neural network together with a 100 dimensional random vector to generate image x. This model allows us to generate images for each category.

We refer to Wang and Liu [26] for more details and discussions about SteinGAN.



DCGAN SteinGAN

Figure 3: Images generated by DCGAN and our SteinGAN on LSUN.

B Empirical Results for Hyper-parameter Optimization of Bayesian Inference

We applied our method to adaptively learn the optimal learning rate for stochastic gradient Langevin dynamics (SGLD) [23]. Denote by $D = \{z_j\}_{j=1}^N$ an observed dataset drawn i.i.d. by $p(z_i|x)$, and x is a random parameter with prior $p_0(x)$. The posterior distribution of x is

$$p(x|D) \propto p_0(x) \prod_{j=1}^{N} p(z_j|x).$$

SGLD draw approximate sample from $p(x|D)$ via iterative update of form

$$x^{t+1} \leftarrow x^t + \eta^t \cdot [\log p_0(x^t) + \frac{N}{|\mathbb{M}^t|} \sum_{j \in \mathbb{M}^t} \nabla_x \log p(D_j|x^t)] + \sqrt{2\eta^t} \cdot \xi^t,$$

Inception Score				
	Real Training Set	500 Duplicate	DCGAN	SteinGAN
Model Trained on ImageNet	11.237	11.100	6.581	6.351
Model Trained on CIFAR-10	9.848	9.807	7.368	7.428

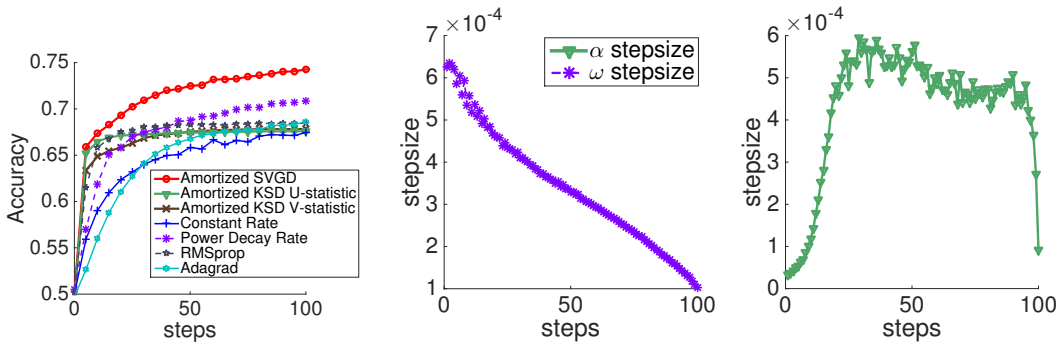
Testing Accuracy			
Real Training Set	500 Duplicate	DCGAN	SteinGAN
92.58 %	44.96 %	44.78 %	63.81 %

Figure 4: Results on CIFAR-10. "500 Duplicate" denotes 500 images randomly subsampled from the training set, each duplicated 100 times. Upper: images simulated by DCGAN and SteinGAN (based on joint model (5)) conditional on each category. Middle: inception scores for samples generated by various methods (all with 50,000 images) on inception models trained on ImageNet and CIFAR-10, respectively. Lower: testing accuracy on real testing set when using 50,000 simulated images to train ResNets for classification. SteinGAN achieves higher testing accuracy than DCGAN.

where ξ^t is a standard Gaussian random vector of the same size as x, and $\mathbb{M}^t$ is a random mini-batch selected at t-th iteration (we use a mini-batch size of 100), and η^t denotes a (vector) stepsize at t-th iteration. Consider running SGLD for $T = 100$ iterations, we can treat x^T as the output of a T-layer neural network parametrized the collection of stepsizes $\eta = \{\eta^t\}_{t=1}^T$, with x_0 and $\{\mathbb{M}^t, \xi^t\}_{t=1}^T$ as the random inputs. This allows us to apply amortized SVGD or KSD to adaptively estimate the optimal stepsize η.

We test our method with Bayesian logistic regression on the Covertype dataset. To demonstrate that our estimated learning rate can work well on new datasets never seen by the algorithm. We partition the dataset into mini-datasets of size $50,000$, and use 80% of them for training and 20% for testing. We adapt our amortized SVGD/KSD to train on the whole population of training mini-datasets by randomly selecting a mini-dataset at each iteration of Algorithm 1. Figure 5 reports the test accuracy when we apply the estimated step sizes on the 20% mini-datasets held for testing. We find that our method outperforms all the hand-designed learning rates, including Adagrad [24] and Rmsprop. We find that amortized KSD does not work as well as amortized SVGD, likely because KSD is a weaker discrepancy measure compared with KL divergence.

We refer to Liu and Feng [27] for more details and discussions about wild variational inference using Stein discrepancy.



(a) (b) (c)

Figure 5: (a) The testing accuracy of the first $T = 100$ iterations with learned and hand-designed step sizes. We find that our method outperforms all the hand-designed learning rates, as well as amortized KSD. (b)-(c) Examples of step sizes learned by amortized SVGD (for two different dimensions of x).

Figure 6: More images generated by SteinGAN on CelebA.