Approximate Probabilistic Inference with Composed Flows

Jay Whang
Department of Computer Science
University of Texas at Austin
jaywhang@cs.utexas.edu

Erik M. Lindgren
Google Research
erikml@google.com

Alexandros G. Dimakis
University of Texas at Austin
dimakis@austin.utexas.edu

Abstract

We study the problem of probabilistic inference on the joint distribution defined by a normalizing flow model. Given a pre-trained flow model $p(\boldsymbol{x})$, we wish to estimate $p(\boldsymbol{x}_2 \mid \boldsymbol{x}_1)$ for some partitioning of the variables $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$. We first show that this task is computationally hard for a large class of flow models. Motivated by this, we propose a framework for *approximate* probabilistic inference. Specifically, our method trains a new flow model with the property that its composition with the given model approximates the target conditional distribution. We describe how we can train this new model using variational inference and handle conditioning under arbitrary differentiable transformations. Experimentally, our approach outperforms Langevin Dynamics in terms of sample quality, while requiring much fewer parameters and training time compared to regular variational inference. We further validate the flexibility of our method on a variety of inference tasks with applications to inverse problems.

1 Introduction

Deep generative models have seen an unprecedented growth in the recent years. Among them, normalizing flow models [1] stand out due to their computational flexibility, as they offer efficient sampling, likelihood evaluation, and inversion. Even with such computational flexibility, however, efficient probabilistic inference on a flow model still remains largely unsolved. This question is becoming increasingly important as generative models increase in size and the computational resources necessary to train them from scratch are out of reach for many researchers and practitioners[1]. If it was possible to perform probabilistic inference on flow models, we could *re-purpose* these powerful pre-trained generators for numerous custom tasks.

We propose a novel method that leverages a powerful pre-trained flow model *by constructing carefully designed latent codes* to generate conditional samples. Specifically, we use variational inference to learn a distribution in the latent space of the given model such that, when fed into the pre-trained model, the output approximately matches the samples from the true conditional.

Our contributions:

[1] For example, Kingma and Dhariwal [2] report that their largest model had 200M parameters and was trained on 40 GPUs for a week.

NeurIPS 2020 Workshop on Deep Learning and Inverse Problems, virtual.

- We establish a hardness result that, even though flow models are designed to provide efficient inversion and sampling, *exact conditional sampling is provably computationally intractable* for a wide class of flow models. This motivates our approach of *approximate* inference.
- We develop a method to estimate the target conditional distribution by composing a second generative model (which we call the *pre-generator*) with the given model. Specifically, we show how to train the pre-generator to yield structured noise so that the composed model approximately matches the target conditional distribution.
- We experimentally show that our approach is competitive with MCMC methods in terms of speed and widely-used sample quality metrics. We also demonstrate that it achieves superior conditional likelihood estimation performance compared to regular variational inference. We further extend and validate the flexibility of our method on conditioning under arbitrary differentiable transformations with applications to inverse problems.

2 Hardness of Conditional Sampling on Flows

Before we present our method, we first establish a hardness result for exact conditional sampling for flow models that use additive coupling layer. Specifically, if an algorithm is able to efficiently sample from the conditional distribution of a flow model with additive coupling layers, then this algorithm can be used to solve NP-complete problems efficiently. Our hardness result holds even if we allow the conditional sampler to approximately match the conditional distribution. The formal statement of the below theorem and its corollary to the approximate case can be found in Appendix A.

Theorem 1. *(Informal) Suppose there is an efficient algorithm that can draw samples from the conditional distribution of a normalizing flow model implemented with additive coupling layers as defined in Dinh et al. [3]. Then $RP = NP$.*

3 Approximate Probabilistic Inference with Composed Flows

Notation. Let $p_f(\boldsymbol{x})$ denote the *base model*, a fixed flow model defined by the invertible mapping $f : \boldsymbol{z} \mapsto \boldsymbol{x}$. The *pre-generator* $p_{\hat{f}}(\boldsymbol{z})$ is similarly defined by the invertible mapping $\hat{f} : \boldsymbol{\epsilon} \mapsto \boldsymbol{z}$ and represents a distribution in the latent space of the base model. We assume that all flow models use standard Gaussian prior, i.e. $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, I_d) \to \boldsymbol{x} = f(\boldsymbol{z})$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I_d) \to \boldsymbol{z} = \hat{f}(\boldsymbol{\epsilon})$. By composing the base model and the pre-generator, we obtain the *composed model* $p_{f \circ \hat{f}}(\boldsymbol{x})$ whose samples are generated via $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I_d) \to \boldsymbol{x} = f(\hat{f}(\boldsymbol{\epsilon}))$.

Our method. Motivated by the hardness of exact conditional sampling, we propose to perform variational inference *in the latent space on a smoothed version of the problem* where we allow the observed variable to approximately match the given observation. We do this by creating a dummy variable $\tilde{\boldsymbol{x}}_1$ distributed according to $p_\sigma(\tilde{\boldsymbol{x}}_1 \mid \boldsymbol{x}_1) = \mathcal{N}(\tilde{\boldsymbol{x}}_1; \boldsymbol{x}_1, \sigma^2 I_d)$. Here, σ is the *smoothing parameter* that controls the tightness of this relaxation. The objective we minimize is the KL divergence between our variational distribution and the smoothed conditional density:

$$D_{\mathrm{KL}}(p_{f \circ \hat{f}}(\boldsymbol{x}_2) \parallel p_f(\boldsymbol{x}_2 \mid \tilde{\boldsymbol{x}}_1 = \boldsymbol{x}_1^*)). \tag{1}$$

For comparison, a direct application of variational inference in the image space (which we refer to as *Ambient VI*) would minimize the following objective:

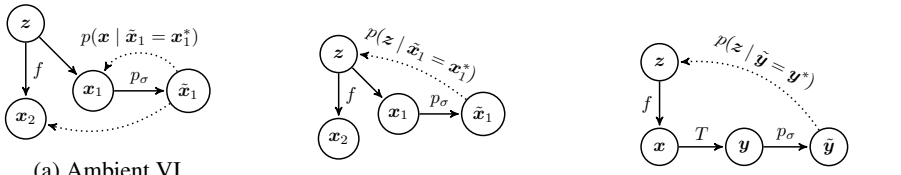
$$D_{\mathrm{KL}}(p_{\hat{f}}(\boldsymbol{x}_2) \parallel p_f(\boldsymbol{x}_2 \mid \tilde{\boldsymbol{x}}_1 = \boldsymbol{x}_1^*)), \tag{2}$$

where we write $p_{\hat{f}}$ to denote the variational distribution which directly models $f(\boldsymbol{x} \mid \tilde{\boldsymbol{y}} = \boldsymbol{y}^*)$.

Note that in our setting, we cannot directly model $p_f(\boldsymbol{x}_2 \mid \tilde{\boldsymbol{x}}_1 = \boldsymbol{x}_1^*)$ because we only have access to the joint distribution $p_f(\boldsymbol{x}_1, \boldsymbol{x}_2)$ through the base model. Thus we instead approximate the *joint* distribution conditioned on our observation, i.e. $p_f(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}_1 = \boldsymbol{x}_1^*)$. Figures 1a and 1b show the graphical models describing this formulation.

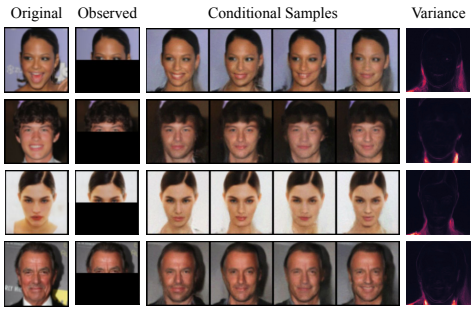
Conditioning under differentiable transformations. The flexibility of VI allows us to easily extend our method to conditioning under an arbitrary transformation. Concretely, let $T(\boldsymbol{x})$ be a differentiable function and $\boldsymbol{y}^*$ be some fixed observation in the range of T. We now observe $\boldsymbol{y} = T(\boldsymbol{x})$ instead, so we similarly define a dummy variable $\tilde{\boldsymbol{y}}$ distributed according to $p_\sigma(\tilde{\boldsymbol{y}} \mid \boldsymbol{y}) = \mathcal{N}(\tilde{\boldsymbol{y}}; \boldsymbol{y}, \sigma^2 I_d)$. We estimate the conditional density $p_f(\boldsymbol{x} \mid \tilde{\boldsymbol{y}} = \boldsymbol{y}^*)$ by minimizing the following objective:

$$\mathcal{L}_{\text{Ours}}(\hat{f}) = D_{\mathrm{KL}}(p_{\hat{f}}(\boldsymbol{z}) \parallel p_f(\boldsymbol{z})) + \mathbb{E}_{\boldsymbol{z} \sim p_{\hat{f}}} \left[\frac{1}{2\sigma^2} \left\| T(f(\boldsymbol{z})) - \boldsymbol{y}^* \right\|_2^2 \right], \tag{3}$$

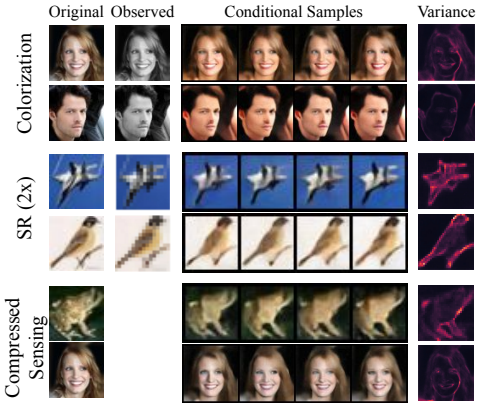


(a) Ambient VI (b) Latent VI (c) Latent VI with transformation

Figure 1: Graphical models depicting different ways we perform variational inference. Solid arrows represent the generative direction, and dotted arrows indicate the variational direction.



(a) Conditional samples generated by our method from observing the upper half of CelebA-HQ faces. We see that our approach is able to produce diverse completions with different jaw lines, mouth shapes, and facial expression.



(b) Results for various inverse problem tasks.

Figure 2: Image inpainting (left) and inverse problem (right) results. Best viewed electronically.

where $p_f(\boldsymbol{z})$ denotes the prior distribution of the base model, i.e. $\mathcal{N}(\mathbf{0}, I_d)$. We provide the derivation of eq. (3) in Appendix C. See Figure 1c for a comparison to eq. (1). Notice that the above expectation can be rewritten in terms of $\boldsymbol{\epsilon}$ to employ the *reparametrization trick* to obtain a low-variance gradient estimator for training.

4 Quantitative Experiments

We validate the efficacy of our proposed method in terms of both sample quality and likelihood on various inference tasks against three baselines: Ambient VI (as defined by the loss in eq. (2)), Langevin Dynamics, and PL-MCMC. We also conduct our experiments on three different datasets (MNIST, CIFAR-10, and CelebA-HQ) to ensure that our method works across a range of settings.

We report four different sample quality metrics: Frechet Inception Distance (FID), Learned Perceptual Image Patch Similarity (LPIPS), and Inception Score (IS) for CIFAR-10 [4–6]. While not strictly a measure of perceptual similarity, average mean squared error (MSE) is also reported for completeness.

For all our experiments, we use the multiscale RealNVP architecture [7] for both the base model and the pre-generator. We use Adam optimizer [8] to optimize the weights of the pre-generator using the loss defined in Equation (3). The images used to generate observations were taken from the test set and were not used to train the base models. We refer the reader to Appendix D for model hyperparameters and other details of our experiment setup.

Table 1: Sample quality metrics for image inpainting tasks on different datasets. The best value is bolded for each metric. As shown below, our method achieves superior sample quality to all baselines.

	MNIST			CIFAR-10 (5-bit)				CelebA-HQ (5-bit)		
	FID	MSE	LPIPS	FID	IS↑	MSE	LPIPS	FID	MSE	LPIPS
Ours	**4.11**	**21.67**	**0.074**	**41.14**	**7.189**	9.71	**0.176**	33.61	**223.06**	**0.208**
Langevin	14.34	36.51	0.135	47.53	6.732	**9.31**	0.201	**30.33**	323.47	0.229
Ambient VI	114.59	65.56	0.290	84.78	5.156	16.74	0.296	289.64	1060.66	0.587
PL-MCMC	21.20	59.89	0.190	N/A				N/A		

4.1 Image Inpainting

We perform inpainting tasks using our approach, where we sample missing pixels conditioned on the visible ones. We consider three different conditioning schemes: the bottom half (MNIST), the upper

half (CelebA-HQ), and randomly chosen subpixels (CIFAR-10). For MNIST, we use the smoothing parameter value of $\sigma = 0.1$ and for CIFAR-10 and CelebA-HQ, we use $\sigma = 0.05$.

In Figure 2a we see that our approach produces natural and diverse samples for the missing part of the image. The empirical pixelwise variance (normalized and averaged over the color channels) also confirms that, while the observation is not perfectly matched, most of the high-variance regions are in the unobserved parts as we expect.

We also quantitatively evaluate the quality of the generated samples using widely used sample quality metrics, as shown in Table 1. As we can see, our method outperforms the baseline methods on most of the metrics. Note that PL-MCMC results for CIFAR-10 and CelebA-HQ are omitted because it was prohibitively slow to run for hundreds of images, as each MCMC chain required over 20,000 proposals. Cannella et al. [9] also report using 25,000 proposals for their experiments.

Observing Table 1 closely, one might notice that Ambient VI performs significantly worse than other methods. While it may seem strange at first, this is actually a natural consequence of the Ambient VI formulation (see eq. (2)). We provide a detailed explanation for this in Appendix B.

4.2 Likelihood Estimation

Next, we evaluate our method on the task of conditional likelihood estimation. By varying the size of the pre-generator, we observe the parameter efficiency of our method compared to Ambient VI. Results are shown in Table 2; we see that our method produces reasonable samples using only *about 1%* of the base model's parameters, confirming the effectiveness of inference in the latent space.

Table 2: Conditional likelihood estimation performance (measured in bits per dimension) for different pre-generator sizes on the MNIST imputation task. The first row shows the parameter count of the pre-generator relative to the base model.

Observations			
# Parameters	Ours	Ambient VI	Conditional Completions (ours)
1.2%	1.73	6.75	
3.2%	1.64	3.17	
10.6%	1.52	2.71	
39.1%	1.47	2.99	

5 Qualitative Experiments

Extracting Class-conditional Models: Here we present an interesting application of conditioning under a transformation T parameterized by a neural network. If T is a pre-trained binary classifier for a specific attribute, we can *extract* a model conditioned on the presence (or the absence) of that attribute from an unconditional base model. We test this idea on the MNIST dataset. We train 10 binary classifiers, one for each digit $k = 0, \ldots, 9$, to predict whether the given image is k or not. By setting T to be each of those classifiers, we are able to extract the class-conditional model $p_f(\boldsymbol{x} \mid \text{Label}(\boldsymbol{x}) \approx k)$. See Figure 3 for samples generated from the extracted models.

Inverse Problems: We additionally test the applicability of our method to linear inverse problems on CIFAR-10 and CelebA-HQ images. In Figure 2b, we show the conditional samples obtained by our method on three different tasks: image colorization, super-resolution ($2\times$), and compressed sensing with 500 random Gaussian measurements. We notice that the generated samples look natural, even when they do not match the original input perfectly, again showing our method's capability to generate semantically meaningful conditional samples and also provide sample diversity.

Figure 3: Each column contains uncurated samples from the posterior conditioned on a specific MNIST class.

6 Conclusion

We proposed a new inference algorithm for distributions parametrized by a flow. The need for approximate inference is motivated by the hardness of exact inference. We also presented a detailed empirical evaluation of our method with both quantitative and qualitative results on a wide range of tasks and datasets. Overall, we believe that the idea of a pre-generator creating structured noise is a useful and general method for leveraging pre-trained generators to solve new generative problems.

Acknowledgments

This research has been supported by NSF Grants CCF 1763702,1934932, AF 1901292, 2008710, 2019844 research gifts by Western Digital, WNCG IAP, computing resources from TACC and the Archie Straiton Fellowship.

References

[1] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

[2] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Neural Information Processing Systems*, pages 10215–10224, 2018.

[3] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. In *International Conference on Learning Representations 2015 workshop track*, 2015.

[4] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[5] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[6] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *International Conference on Learning Representations*, 2016.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Chris Cannella, Mohammadreza Soltani, and Vahid Tarokh. Projected latent markov chain monte carlo: Conditional sampling of normalizing flows, 2020.

[10] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013. ISBN 9780262018029 0262018020. URL `https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr_1_2?ie=UTF8&qid=1336857747&sr=8-2`.

A Proof of Hardness Results

A.1 Preliminaries

A Boolean variable is a variable that takes a value in $\{-1, 1\}$. A *literal* is a Boolean variable x_i or its negation $\neg x_i$. A *clause* is set of literals combined with the OR operator, e.g., $x_1 \vee \neg x_2 \vee x_3$. A *conjunctive normal form formula* is a set of clauses joined by the AND operator, e.g., $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_3 \vee x_4)$. A satisfying assignment is an assignment to the variables such that the Boolean formula is true.

The *3-SAT problem* is the problem of deciding if a conjunctive normal form formula with three literals per clause has a satisfying assignment. We will show that conditional sampling from flow models allows us to solve the 3-SAT problem.

We ignore the issue of representing samples from the conditional distribution with a finite number of bits. However the reduction is still valid if the samples are truncated to a constant number of bits.

A.2 Design of the Additive Coupling Network

Given a Boolean formula, we design a ReLU neural network with 3 hidden layers such that the output is 0 if the input is far from a satisfying assignment, and the output is about a large number M if the input is close to a satisfying assignment.

We will define the following scalar function

$$\begin{aligned}\delta_\varepsilon(x) =& \text{ReLU}\left(\frac{1}{\varepsilon}(x-(1-\varepsilon))\right) - \text{ReLU}\left(\frac{1}{\varepsilon}(x-(1-\varepsilon))-1\right)\\ &-\text{ReLU}\left(\frac{1}{\varepsilon}(x-1)\right) + \text{ReLU}\left(\frac{1}{\varepsilon}(x-1)-1\right).\end{aligned}$$

This function is 1 if the input is 1, 0 if the input x has $|x-1| \geq \varepsilon$ and is a linear interpolation on $(1-\varepsilon, 1+\varepsilon)$. Note that it can be implemented by a hidden layer of a neural network and a linear transform, which can be absorbed in the following hidden layer. See Figure 4 for a plot of this function.

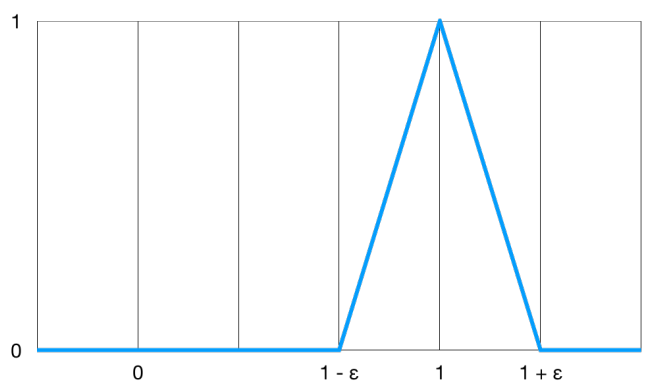


Figure 4: Plot of the scalar function used to construct an additive coupling layer that can generate samples of satisfying 3-SAT assignments.

For each variable x_i, we create a transformed variable $\tilde{x}_i$ by applying $\tilde{x}_i = \delta_\varepsilon(x_i) - \delta_\varepsilon(-x_i)$. Note that this function is 0 on $(-\infty, -1-\varepsilon] \cup [-1+\varepsilon, 1-\varepsilon] \cup [1+\varepsilon, \infty)$, -1 at $x_i = -1$, 1 at $x_i = 1$, and a smooth interpolation on the remaining values in the domain.

Every clause has at most 8 satisfying assignments. For each satisfying assignment we will create a neuron with the following process: (1) get the relevant transformed values $\tilde{x}_i, \tilde{x}_j, \tilde{x}_k$, (2) multiply each variable by $1/3$ if it is equal to 1 in the satisfying assignment and $-1/3$ if it is equal to -1 in the satisfying assignment, (3) sum the scaled variables, (4) apply the δ_ε function to the sum.

We will then sum all the neurons corresponding to a satisfying assignment for clause C_j to get the value c_j. The final output is the value $M \times \text{ReLU}(\sum_j c_j - (m-1))$, where M is a large scalar.

We say that an input to the neural network x corresponds to a Boolean assignment $x' \in \{-1, 1\}^d$ if for every x_i we have $|x_i - x'_i| < \varepsilon$. For $\varepsilon < 1/3$, if the input does not correspond to a satisfying assignment of the given formula, then at least one of the values c_j is 0. The remaining values of c_j are at most 1, so the sum in the output is at most $(m-1)$, thus the sum is at most zero, so the final output is 0. However, if the input is a satisfying assignment, then every value of $c_j = 1$, so the output is M.

A.3 Generating SAT Solutions from the Conditional Distribution

Our flow model will take in Gaussian noise $x_1, \ldots, x_d, z \sim N(0, 1)$. The values $x_1, \ldots, x_d$ will be passed through to the output. The output variable y will be $z + f_M(x_1, \ldots, x_d)$, where f_M is the neural network described in the previous section, and M is the parameter in the output to be decided later.

Let A be all the valid satisfying assignments to the given formula. For each assignment a, we will define X_a to be the region $X_a = \{x \in \mathbb{R}^d : \|a - x\|_\infty \leq \varepsilon\}$, where as above ε is some constant less than $1/3$. Let $X_A = \bigcup_{a \in A} X_a$.

Given an element $x \in X_a$, we can recreate the corresponding satisfying assignment a. Thus if we have an element of X_A, we can certify that there is a satisfying assignment. We will show that the distribution conditioned on $y = M$ can generate satisfying assignments with high probability.

We have that

$$p(X_A \mid y = M) = \frac{p(y = M, X_A)}{p(y = M, X_A) + p(y = M, \overline{X}_A)}$$

If we can show that $p(y = M, \overline{X}_A) \ll p(y = M, X_A)$, then we have that the generated samples are with high probability satisfying assignments.

Note that,

$$p(y = M, \overline{X}_A) = p(y = M \mid \overline{X}_A) P(\overline{X}_A) \leq p(y = M \mid \overline{X}_A).$$

Also notice that if $x \in \overline{X}_A$, then $f_M(x) = 0$. Thus $y \sim \mathcal{N}(0, 1)$ and $P(y = M \mid \overline{X}_A) = \Theta(\exp(-M^2/2))$.

Now consider any satisfying assignment x_a. Let X'_a be the region $X'_a = \{x \in \mathbb{R}^d : \|a - x\|_\infty \leq \frac{1}{2m}\}$. Note that for every x in this region we have $f_M(x) \geq M/2$. Additionally, we have that $P(X'_a) = \Theta(m)^{-d}$. Thus for any $x \in X'_a$, we have $p(Y = M \mid x) \gtrsim \exp(-M^2/8)$. We can conclude that

$$p(y = M, X_A) \geq p(Y = M, X'_a) = \int_{X'_a} p(Y = M \mid x) p(x)\, dx \gtrsim \exp(-M^2/8 - \Theta(d \log m)).$$

For $M = O(\sqrt{d \log m})$, we have that $p(y = M, \overline{X}_A)$ is exponentially smaller than $p(y = M, X_A)$. This implies that sampling from the distribution conditioned on $y = M$ will return a satisfying assignment with high probability.

A.4 Hardness of Approximate Sampling

Definition 2. The complexity class RP is the class of decision problems with efficient random algorithms that (1) output YES with probability $1/2$ if the true answer is YES and (2) output NO with probability 1 if the true answer is NO. It is widely believed that RP is a strict subset of NP.

A simple extension of the above theorem shows that even approximately matching the true conditional distribution in terms of TV distance is computationally hard. The total variation (TV) distance is defined as $d_{\mathrm{TV}}(p, q) = \sup_E |p(E) - q(E)| \leq 1$, where E is an event. The below corollary shows that it is hard to conditionally sample from a distribution that is even slightly bounded away from 1.

Corollary 3. *The conditional sampling problem remains hard even if we only require the algorithm to sample from a distribution q such that $d_{\mathrm{TV}}(p(\cdot \mid x = x^*), q) \leq 1 - 1/\mathrm{poly}(d)$, where d is the dimension of the distribution.*

We show that the problem is still hard even if we require the algorithm to sample from a distribution q such that $d_{\mathrm{TV}}(p(x \mid y = y^*), q) \geq 1/\mathrm{poly}(d)$.

Consider the event X_A from above. We saw that $p(X_A \mid y = M) \geq 1 - \exp(-\Omega(d))$. We have that $d_{\mathrm{TV}}(p(\cdot \mid y = M), q) \geq 1 - \exp(-\Omega(d) - q(X_A))$.

Suppose that the distribution q has $q(X_A) \geq 1/\text{poly}(d)$. Then by sampling a polynomial number of times from q we sample an element of X_A, which allows us to find a satisfying assignment. Thus if we can efficiently create such a distribution, we would be able to efficiently solve SAT and RP = NP. As we are assuming this is false, we must have $q(X_A) \leq 1/\text{poly}(d)$, which implies $d_{\text{TV}}(p(\cdot \mid y = M), q) \geq 1 - 1/\text{poly}(d)$.

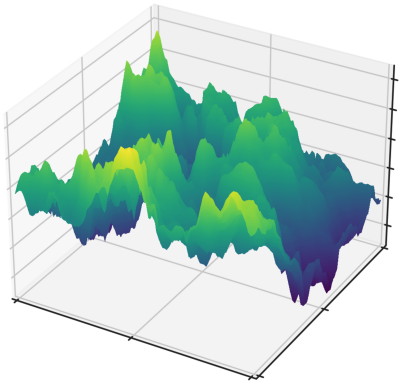
B Why Ambient VI Fails

From Table 1, notice that Ambient VI achieves significantly worse sample quality compared to other methods. The low-quality samples from the image inpainting task in Figure 5b further confirm that Ambient VI is unable to produce good conditional samples, even though the observation is matched well. This may seem initially surprising but is a natural consequence of the VI objective. Recall that our loss function decomposes into two terms: the KL term and the reconstruction term.

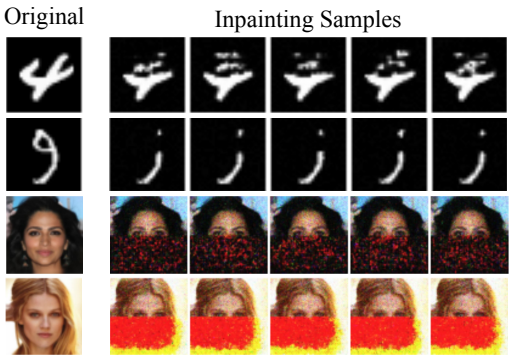
$$\mathcal{L}_{\text{Ours}}(\hat{f}) = D_{\text{KL}}(p_{\hat{f}}(\boldsymbol{z}) \parallel p_f(\boldsymbol{z})) + \mathbb{E}_{\boldsymbol{z} \sim p_{\hat{f}}} \left[\frac{1}{2\sigma^2} \|T(f(\boldsymbol{z})) - \boldsymbol{y}^*\|_2^2 \right].$$

If we alternatively derive the loss for Ambient VI, we arrive at an analogous objective:

$$\mathcal{L}_{\text{Ambient}}(\hat{f}) = D_{\text{KL}}(p_{\hat{f}}(\boldsymbol{x}) \parallel p_f(\boldsymbol{x})) + \mathbb{E}_{\boldsymbol{x} \sim p_{\hat{f}}} \left[\frac{1}{2\sigma^2} \|T(\boldsymbol{x}) - \boldsymbol{y}^*\|_2^2 \right].$$



(a) Contour plot of $p_f(\boldsymbol{x})$ around a random point in image space.



(b) Various failure modes exhibited by naive VI.

While these two loss functions seem like simple reparametrizations of each other via f, they behave very differently during optimization due to the KL term. Notice that for both loss functions, the first term is the *reverse* KL divergence between the variational distribution and the base distribution $D_{\text{KL}}(p_{\hat{f}} \parallel p_f)$. Because reverse KL divergence places no penalty whenever $p_{\hat{f}}$ is zero regardless of p_f, minimizing the reverse KL is known to have a *mode-seeking* behavior where $p_{\hat{f}}$ fits a single mode of p_f and ignores the rest of the support of p_f [10, Chapter 21.2.2]. In contrast, minimizing the forward KL has a *zero-avoiding* behavior and tries to cover all of p_f's support.

For our method, this is not a problem because the prior distribution of the base model $p_f(\boldsymbol{z})$ is a standard Gaussian and hence unimodal. However, for Ambient VI, $p_f(\boldsymbol{x})$ is the base model itself and is highly multimodal. This can be empirically seen by visualizing the landscape of $\log p_f(\boldsymbol{x})$ projected onto a random 2D subspace. In Figure 5a, we clearly see that $p_f(\boldsymbol{x})$ has numerous local maxima. For Ambient VI, the variational distribution collapses into one of these modes.

C Derivation of Equation (3)

Here we present a detailed derivation of Equation (3). Note that this equality is true *up to a constant w.r.t.* $\hat{f}$, which is fine as we use this as the optimization objective.

$$
\begin{aligned}
\mathcal{L}_{\text{Ours}}(\hat{f}) &\triangleq D_{\mathrm{KL}}(p_{f\circ\hat{f}}(\boldsymbol{x}) \parallel p_f(\boldsymbol{x} \mid \tilde{\boldsymbol{y}} = \boldsymbol{y}^*)) \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{f\circ\hat{f}}}\left[\log p_{f\circ\hat{f}}(\boldsymbol{x}) - \log p_f(\boldsymbol{x}, \tilde{\boldsymbol{y}} = \boldsymbol{y}^*)\right] + \log p_f(\tilde{\boldsymbol{y}} = \boldsymbol{y}^*) \\
&\overset{A}{=} \mathbb{E}_{\boldsymbol{x}\sim p_{f\circ\hat{f}}}\left[\log p_{f\circ\hat{f}}(\boldsymbol{x}) - \log p_f(\boldsymbol{x}) - \log p_\sigma(\tilde{\boldsymbol{y}} = \boldsymbol{y}^* \mid \boldsymbol{x})\right] \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{f\circ\hat{f}}}\left[\log p_{f\circ\hat{f}}(\boldsymbol{x}) - \log p_f(\boldsymbol{x})\right] + E_{\boldsymbol{x}\sim p_{f\circ\hat{f}}}\left[-\log p_\sigma(\tilde{\boldsymbol{y}} = \boldsymbol{y}^* \mid \boldsymbol{y} = T(\boldsymbol{x}))\right] \\
&= D_{\mathrm{KL}}(p_{f\circ\hat{f}}(\boldsymbol{x}) \parallel p_f(\boldsymbol{x})) + E_{\boldsymbol{x}\sim p_{f\circ\hat{f}}}\left[-\log p_\sigma(\tilde{\boldsymbol{y}} = \boldsymbol{y}^* \mid \boldsymbol{y} = T(\boldsymbol{x}))\right] \\
&\overset{B}{=} D_{\mathrm{KL}}(p_{\hat{f}}(\boldsymbol{z}) \parallel p_f(\boldsymbol{z})) + E_{\boldsymbol{z}\sim p_{\hat{f}}}\left[\frac{1}{2\sigma^2}\left\|T(f(\boldsymbol{z})) - \boldsymbol{y}^*\right\|_2^2\right]
\end{aligned}
$$

In (A), we drop the $\log p_f(\tilde{\boldsymbol{y}} = \boldsymbol{y}^*)$ term as it is constant w.r.t. $\hat{f}$.
In (B), we use the invariance of KL divergence under invertible transformation to rewrite the KL divergence in terms of $\boldsymbol{z}$.

D Experiment Details

D.1 Our Algorithm

Algorithm 1 Training the pre-generator for a given observation under transformation. We assume that $\hat{f}$ is an invertible neural network with parameters θ.

Input: $\boldsymbol{y}^*$: observation we are conditioning on, $T(\boldsymbol{x})$: differentiable transformation of $\boldsymbol{x}$.
for $i = 1 \ldots$ `num_steps` **do**
  for $j = 1 \ldots m$ **do** ▷ generate m latent codes from $p_{\hat{f}}(\boldsymbol{z})$
    Sample $\boldsymbol{\epsilon}^{(j)} \sim \mathcal{N}(\mathbf{0}, I_d)$
    $\boldsymbol{z}^{(j)} \leftarrow \hat{f}(\boldsymbol{\epsilon}^{(j)})$
  end for
  $\mathcal{L} \leftarrow \frac{1}{m}\sum_{j=1}^{m}\left[\log p_{\hat{f}}(\boldsymbol{z}^{(j)}) - \log p_{\text{normal}}(\boldsymbol{z}^{(j)}) + \frac{1}{2\sigma^2}\left\|T(f(\boldsymbol{z}^{(j)})) - \boldsymbol{y}^*\right\|_2^2\right]$
  $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}$ ▷ gradient step
end for

D.2 Hyperparameters: Base Model and Pre-generator

See Table 3 and Table 4 for the hyperparameters used to define the network architectures train them. For the color datasets CIFAR-10 and CelebA-HQ, we used 5-bit pixel quantization following Kingma and Dhariwal [2]. Additionally for CelebA-HQ, we used the same train-test split (27,000/3,000) of Kingma and Dhariwal [2] and resized the images to 64×64 resolution.

Table 3: Hyperparameters used to train the base models used in our experiments.

Base Models	MNIST	CIFAR-10	CelebA-HQ
Image resolution	28×28	32×32	64×64
Num. scales	3	6	6
Res. blocks per scale	8	12	10
Res. block channels	32	64	80
Bits per pixel	8	5	5
Batch size	128	64	32
Learning rate	0.001	0.001	0.001
Num. epochs	200		

Table 4: Hyperparameters used to define and train the pre-generator for each of our experiments.

Base Models	MNIST	CIFAR-10	CelebA-HQ
Image resolution	28×28	32×32	64×64
Num. scales	3	4	3
Res. blocks per scale	3	4	3
Res. block channels	32	48	48
Batch size	64	32	8

D.3 Hyperparameters: Image Inpainting

We randomly chose 900/500/300 images from MNIST/CIFAR-10/CelebA-HQ test sets, applied masks defined in Section 4.1, and generated samples conditioned on the remaining parts. FID and other sample quality metrics were computed using 32 conditional samples per test image for VI methods (ours and Ambient VI), 8 for Langevin Dynamics, and 6 for PL-MCMC. We note that using more samples for VI methods do not unfairly affect the result of sample quality evaluation, i.e. there was no appreciable difference when using 8 vs. 32 samples to compute FID. We used more samples simply because it is much cheaper for VI methods to generate samples compared to MCMC methods.

For VI Methods (Ours & Ambient VI)

- Learning rate: $1\mathrm{e}{-3}$ for MNIST; $5\mathrm{e}{-4}$ for the others
- Number of training steps: 2000 for CelebA-HQ; 1000 for the others

For Langevin Dynamics

- Learning rate: $5\mathrm{e}{-4}$ for all datasets
- Length of chain: 1000 for CIFAR-10; 2000 for the others

For PL-MCMC

- Learning rate: $5\mathrm{e}{-4}$
- Length of chain: 2000 for MNIST
- $\sigma_a = 1\mathrm{e}{-3}$, $\sigma_p = 0.05$

D.4 Hyperparameters: Inverse Problems

	Colorization	Compressed Sensing	Compressed Sensing	Super-resolution
Learning rate	$5\mathrm{e}{-4}$	$5\mathrm{e}{-4}$	$5\mathrm{e}{-4}$	$5\mathrm{e}{-4}$
σ	0.05	0.05	0.05	0.05
Dataset	CelebA-HQ	CelebA-HQ	CIFAR-10	CIFAR-10
Batch size	8	8	32	32
Number of steps	1000	2000	1000	1000