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 conjunctive normal form with m clauses, 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

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

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 9 for a plot of this function.

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.

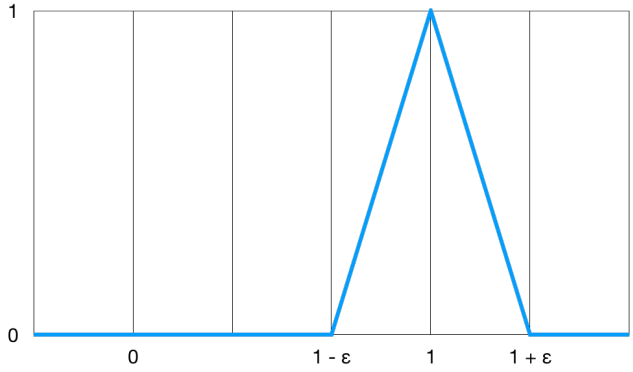

Figure 9: Plot of the scalar function used to construct an additive coupling layer that can generate samples of satisfying 3-SAT assignments.

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 $M \times \operatorname{ReLU}(\sum_j c_j - (m-1))$, 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,

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

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

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

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 the total variation (TV) distance is computationally hard. 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 $q(X_A) \geq 1/\mathrm{poly}(d)$. 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 $q(X_A) \leq 1/\mathrm{poly}(d)$, which implies $d_{\mathrm{TV}}(p(\cdot \mid y = M), q) \geq 1 - 1/\mathrm{poly}(d)$.

B. Missing Derivations

B.1. Derivation of Equation (4)

Here we present a detailed derivation of Equation (4). Note that this equality is true up to a constant w.r.t. $\hat{f}$.

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

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

B.2. Joint VI vs. Marginal VI

We also provide a justification for using the joint VI loss as discussed in Section 4. Specifically, we show that the joint VI loss in eq. (4) is an upper bound to the intractable marginal VI loss. Assuming the partitioning $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$,

we have:

$$
\begin{aligned}
&\text{(Joint KL)} \\
&= D_{\mathrm{KL}}(q_{\boldsymbol{x}}(\boldsymbol{x}) \parallel p_{\boldsymbol{x}}(\boldsymbol{x}|\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}^*)) \\
&= \mathbb{E}_{q_{\boldsymbol{x}}}\left[\log q_{\boldsymbol{x}}(\boldsymbol{x}_1, \boldsymbol{x}_2) - \log p_{\boldsymbol{x}}(\boldsymbol{x}_1, \boldsymbol{x}_2|\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}^*)\right] \\
&= \mathbb{E}_{q_{\boldsymbol{x}}}\Big[\log q_{\boldsymbol{x}}(\boldsymbol{x}_2) + \log q_{\boldsymbol{x}}(\boldsymbol{x}_1 \mid \boldsymbol{x}_2) \\
&\quad - \log p_{\boldsymbol{x}}(\boldsymbol{x}_2|\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}^*) - \log p_{\boldsymbol{x}}(\boldsymbol{x}_1 \mid \tilde{\boldsymbol{x}}_1 = \boldsymbol{x}^*, \boldsymbol{x}_2)\Big] \\
&= \mathbb{E}_{q_{\boldsymbol{x}}}\left[\log q_{\boldsymbol{x}}(\boldsymbol{x}_2) - \log p_{\boldsymbol{x}}(\boldsymbol{x}_2|\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}^*)\right] \\
&+ \mathbb{E}_{q_{\boldsymbol{x}}}\Big[\mathbb{E}_{q_{\boldsymbol{x}}(\boldsymbol{x}_1|\boldsymbol{x}_2)}\Big[\\
&\quad \log q_{\boldsymbol{x}}(\boldsymbol{x}_1 \mid \boldsymbol{x}_2) - \log p_{\boldsymbol{x}}(\boldsymbol{x}_1 \mid \tilde{\boldsymbol{x}}_1 = \boldsymbol{x}^*, \boldsymbol{x}_2)\Big]\Big] \\
&= D_{\mathrm{KL}}(q_{\boldsymbol{x}}(\boldsymbol{x}_2) \parallel p_{\boldsymbol{x}}(\boldsymbol{x}_2|\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}^*)) \\
&\quad + \mathbb{E}_{q_{\boldsymbol{x}}(\boldsymbol{x}_2)}\left[D_{\mathrm{KL}}(q_{\boldsymbol{x}}(\boldsymbol{x}_1 \mid \boldsymbol{x}_2) \parallel p_{\boldsymbol{x}}(\boldsymbol{x}_1|\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}^*, \boldsymbol{x}_2))\right] \\
&\geq D_{\mathrm{KL}}(q_{\boldsymbol{x}}(\boldsymbol{x}_2) \parallel p_{\boldsymbol{x}}(\boldsymbol{x}_2|\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}^*)) \\
&= \text{(Marginal KL)},
\end{aligned}
$$

where the last inequality is due to the nonnegativity of KL. Note that equality holds when

$$D_{\mathrm{KL}}(q_{\boldsymbol{x}}(\boldsymbol{x}_1 \mid \boldsymbol{x}_2) \parallel p_{\boldsymbol{x}}(\boldsymbol{x}_1|\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}^*, \boldsymbol{x}_2)) = 0,$$

i.e. when our variational posterior matches the true conditional.

C. Experiment Details

C.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, A: differentiable measurement function.
for $i = 1 \ldots$ `num_steps` **do**
  for $j = 1 \ldots m$ **do**
    Sample $\boldsymbol{\epsilon}^{(j)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\boldsymbol{z}^{(j)} \leftarrow \hat{f}(\boldsymbol{\epsilon}^{(j)})$ (reparametrization trick)
  end for
  $\mathcal{L} \leftarrow \frac{1}{m}\sum_{j=1}^{m}\Big[\log q_{\boldsymbol{z}}(\boldsymbol{z}^{(j)}) - \log p_{\boldsymbol{z}}(\boldsymbol{z}^{(j)}) + \frac{1}{2\sigma^2}\left\|A(f(\boldsymbol{z}^{(j)})) - \boldsymbol{y}^*\right\|_2^2\Big]$
  $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}$ (gradient step)
end for

C.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 & Dhariwal (2018). Additionally for CelebA-HQ, we used the same train-test split (27,000/3,000) of Kingma & Dhariwal (2018) and resized the images to 64×64 resolution. Uncurated samples from the base models are included for reference in Figure 10.

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
Test set bits-per-dim	1.053	1.725	1.268

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

C.3. Hyperparameters: Image Inpainting

We randomly chose 900/500/300 images from MNIST/CIFAR-10/CelebA-HQ test sets, applied masks defined in Section 6.1, and generated samples conditioned on the remaining parts. FID and other sample quality metrics were computed using 6 conditional samples per test image for all MNIST experiments, and 8 conditional samples for all CIFAR-10 and CelebA-HQ experiments.

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: 4000 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; 4000 for the others

For PL-MCMC

- Learning rate: $5\mathrm{e}{-4}$

Figure 10: Unconditional samples from the base models used for our experiments. From left: MNIST, 5-bit CIFAR-10, and 5-bit CelebA-HQ models.

- Length of chain: 2000 for MNIST
- $\sigma_a = 1\text{e}{-3}$, $\sigma_p = 0.05$

C.4. Hyperparameters: Compressed Sensing

For Ours and (Asim et al., 2019)

- Learning rate: $5\text{e}{-4}$
- Number of training steps: 4000
- For (Asim et al., 2019), we used the same training objective used in their Compressed Sensing experiments: $\arg\min_{\boldsymbol{z}} \|AG(\boldsymbol{z}) - \boldsymbol{y}^*\|_2^2$

For (Bora et al., 2017)

- Learning rate: 0.02
- Regularization coefficient: $\lambda = 0.1$
- Following (Bora et al., 2017), we repeated each run three times and initialized $\boldsymbol{z}_0$ using samples from $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ where $\sigma = 0.1$. Then we used the best result out of the three runs for evaluation.

C.5. Hyperparameters: Inverse Problems

Please see Table 5.

Table 5: Hyperparameters for the extra inverse problem experiments.

	Colorize	CS	CS	SR ($2\times$)
Dataset	CelebA-HQ		CIFAR-10	
Learning rate	$5\text{e}{-4}$	$5\text{e}{-4}$	$5\text{e}{-4}$	$5\text{e}{-4}$
σ	0.05	0.05	0.05	0.05
Batch size	8	8	32	32
Number of steps	1000	2000	1000	1000