Hierarchical Variational Policies for Reward-Guided Diffusion

Kushagra Pandey* **Farrin Marouf Sofian*** **Jan Niklas Groeneveld**
Felix Draxler **Stephan Mandt**

Department of Computer Science
University of California, Irvine

{pandeyk1, fmaroufs, jgroenev, fdraxler, mandt}@uci.edu

Abstract

Adapting pretrained diffusion models to downstream objectives such as inverse problems often requires expensive test-time guidance or optimization. We propose a principled framework for generating high-quality reward-aligned samples at substantially reduced inference cost. Our approach formulates test-time adaptation as a hierarchical variational model, where control is amortized into a lightweight yet expressive stochastic policy. This formulation naturally supports few-step diffusion sampling: large step sizes enable fast inference, while the learned policy maintains sample quality by providing structured per-step control. The resulting fully amortized sampler achieves a strong quality–speed tradeoff, matching or exceeding recent test-time scaling baselines while requiring significantly less compute. For example, on $4\times$ super-resolution, our method achieves better perceptual quality with more than **5$\times$ faster** inference compared to the best-performing baseline. We further extend our approach to a semi-amortized regime that combines cheap amortized proposals with limited test-time optimization, achieving state-of-the-art perceptual quality across several challenging inverse problems. Code and model checkpoints will be made available at `https://github.com/mandt-lab/hvp`.

1 Introduction

Diffusion models [21, 42] and related families [1, 29] are powerful priors for downstream generative tasks. Test-time adaptation of diffusion models steers the denoising process to solve novel tasks such as text-to-image alignment [30, 41, 48] and inverse problems [7, 31, 55, 33, 58, 18] without expensive training of conditional models: just specify the task, e.g., via a likelihood term or a learned reward [52, 51], and sample. However, most existing methods rely on repeated gradient evaluations or inner-loop optimizations that can be prohibitive for high-resolution or real-time scenarios.

A natural remedy is to amortize this cost: train a lightweight policy that *learns to steer* denoising, shifting expense from inference to a one-time training phase. However, such policies are difficult to learn, especially for the multimodal posteriors arising in ill-posed inverse problems or large-step diffusion [40, 44, 53, 17]. We address this by adding latent variables to the policy, yielding a more flexible hierarchical variational model [38]. We train these policies through a variational inference formulation [4, 56]: given a pretrained prior and a downstream reward, we approximate the posterior over denoising trajectories. Several recent methods [18, 33, 31] adopt a similar view but still require expensive optimization at every inference step. Our learned policy replaces that cost with a single forward pass, while also jointly optimizing control signals across all sampling steps.

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

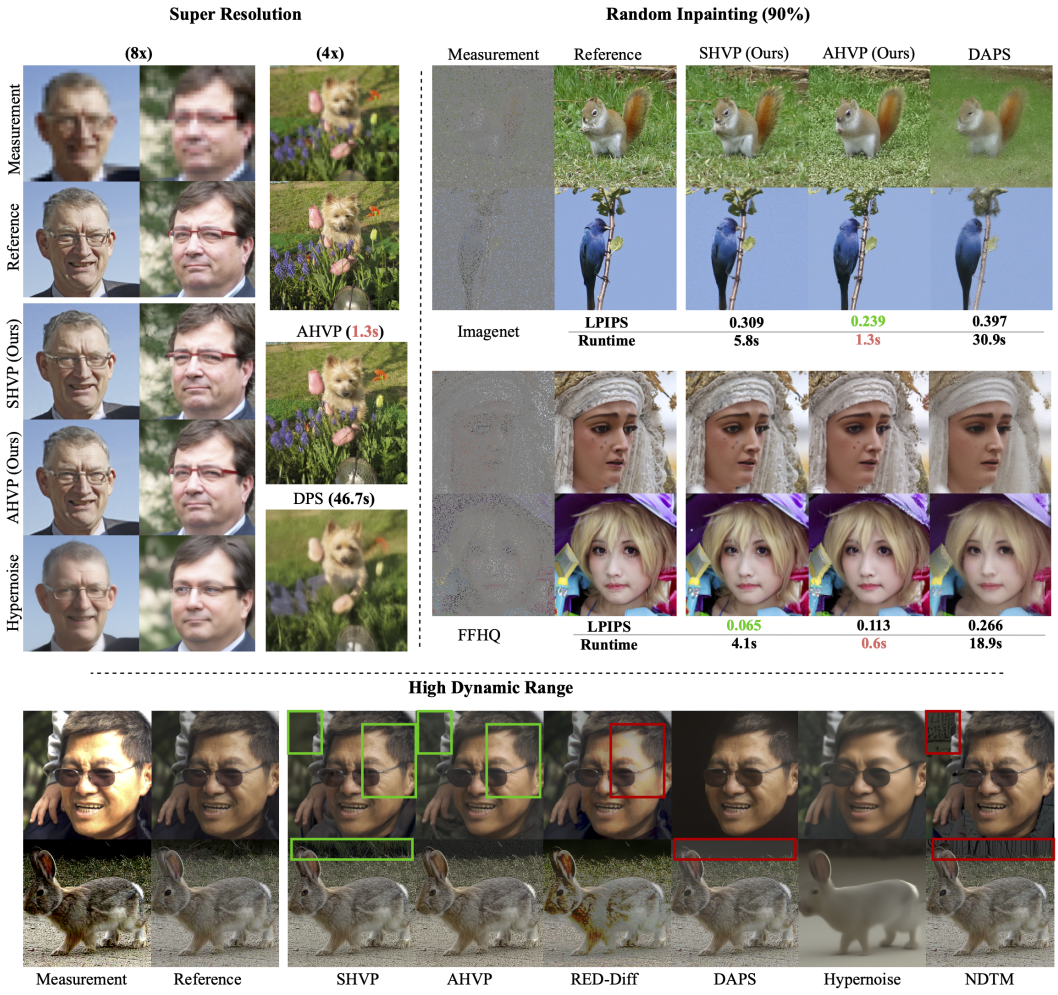
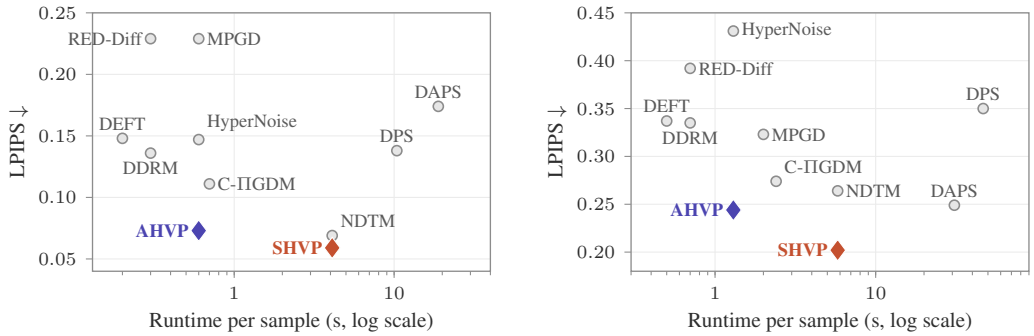


Figure 1: **AHVP and SHVP satisfy measurement constraints with high sample quality at reduced inference cost.** AHVP offers strong perceptual quality with fast inference, while SHVP typically improves quality further at a moderate additional cost. Shown: super-resolution ($\times 8$, $\times 4$), 90% random inpainting, and HDR. In the HDR panel, baselines show artifacts (red boxes) that our methods avoid (green boxes). Inline numbers denote LPIPS and per-sample runtime. Full results in Table 1.

Our framework provides a general and modular procedure for constructing amortized variational policies for a wide range of downstream tasks with differentiable likelihoods or rewards. As a concrete instantiation, we propose a two-stage procedure: first learn an initial noise distribution that maximizes the reward, then train per-step stochastic controllers that capture residual structure. We further extend the framework to a semi-amortized regime that blends cheap amortized proposals with additional test-time refinement. Empirically, on challenging inverse problems, our approach matches or exceeds state-of-the-art test-time scaling baselines while significantly reducing inference cost.

- We introduce a **unified framework** that recasts test-time guidance in diffusion models as inference over hierarchical variational policies, enabling principled amortized control across a broad class of downstream tasks with differentiable rewards.
- We develop **Amortized HVP (AHVP)**, a novel two-stage method that jointly learns an initial noise distribution and per-step stochastic policies, producing high-quality reward-aligned samples in a single forward rollout.
- We design **Semi-Amortized HVP (SHVP)**, which combines amortized proposals with lightweight test-time refinement and **achieves state-of-the-art perceptual quality** on several challenging inverse problems at modest additional cost.



(a) SR (×4) FFHQ-256 (b) SR (×4) ImageNet-256

Figure 2: **AHVP and SHVP achieve a superior quality–speed tradeoff.** LPIPS (lower is better) versus runtime per sample (log scale) for diffusion-based solvers on $\times 4$ super-resolution.

- We demonstrate a **superior quality–speed** tradeoff across four inverse problems on FFHQ-256 and ImageNet-256, with AHVP matching or exceeding the perceptual quality of leading test-time methods at **more than 5× faster inference** (Figs. 1 and 2).

2 Hierarchical Variational Policies for Reward Guidance

Intuitively, our method amortizes guidance by learning how to nudge an unconditional diffusion trajectory. At each denoising step, a lightweight controller observes the current state, timestep, and condition $\boldsymbol{y}$, predicts a control $\mathbf{u}_t$, and the pretrained denoiser is applied to the controlled state. The diffusion model remains fixed; task adaptation is carried by the learned initial-noise predictor and per-step controller. Together, the fixed denoiser and controller act like a task-conditional sampler, with the controller injecting information from $\boldsymbol{y}$ while leaving the pretrained diffusion model unchanged. Inference therefore becomes a simple conditional rollout rather than an inner-loop optimization. We train the per-step controller with a reward or likelihood objective using a Tweedie-estimate surrogate. Below we formalize this procedure as variational inference over controlled denoising trajectories: first as *hierarchical variational policies* (HVP) (section 2.1), then through a tractable learning objective (section 2.2), and finally through amortized parameterizations of the resulting policies (section 2.3).

2.1 Guidance with Hierarchical Variational Policies (HVP)

We now present guidance in diffusion models through the lens of hierarchical variational approximations. Given an observed conditioning signal $\boldsymbol{y}$, we define a generative process that gradually converts noise into samples conditioned on $\boldsymbol{y}$. More specifically,

$$p(\mathbf{x}_{0:T}, \boldsymbol{y}) = p(\mathbf{x}_T) \left[\prod_{t=1}^{T} p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \right] p(\boldsymbol{y} \mid \mathbf{x}_0), \tag{1}$$

where $p(\mathbf{x}_T)$ denotes the prior over the initial noise (commonly a standard Gaussian distribution) and $p(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ intuitively infers a denoised state $\mathbf{x}_{t-1}$, given the previous noisy state $\mathbf{x}_t$ [21, 1, 29]. The distribution $p(\boldsymbol{y} \mid \mathbf{x}_0)$ represents the likelihood of observing $\boldsymbol{y}$ given the final denoised state $\mathbf{x}_0$. In practice, the form of the likelihood is application-specific. For instance, in inverse problems (like inpainting), it can be defined as $p(\boldsymbol{y} \mid \mathbf{x}_0) = \mathcal{N}(\boldsymbol{y}; \mathcal{A}(\mathbf{x}_0), \sigma_y^2 \boldsymbol{I}_d)$ where $\mathcal{A}$ *corrupts* the denoised sample $\mathbf{x}_0$. In this work, we assume that the likelihood distribution is **known** and **differentiable**.

Our goal is to infer the intermediate states $\mathbf{x}_{0:T}$ that maximize the likelihood of $\boldsymbol{y}$. Although the main object of interest is $p(\mathbf{x}_0 \mid \boldsymbol{y})$, approximating the full trajectory posterior $p(\mathbf{x}_{0:T} \mid \boldsymbol{y})$ allows us to reuse the diffusion generative dynamics for downstream tasks. Since this posterior is intractable, we introduce a variational distribution $q(\mathbf{x}_{0:T} \mid \boldsymbol{y})$. The resulting Evidence Lower Bound (ELBO) is

$$\log p(\boldsymbol{y}) = \log \int q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) \frac{p(\mathbf{x}_{0:T}, \boldsymbol{y})}{q(\mathbf{x}_{0:T} \mid \boldsymbol{y})} d\mathbf{x}_{0:T} \quad \geq \mathbb{E}_{q(\mathbf{x}_{0:T} \mid \boldsymbol{y})} \left[\log p(\mathbf{x}_{0:T}, \boldsymbol{y}) - \log q(\mathbf{x}_{0:T} \mid \mathbf{y}) \right]. \tag{2}$$

The variational distribution $q(\mathbf{x}_{0:T} \mid \boldsymbol{y})$ can be interpreted as a *stochastic policy* over diffusion trajectories. While Eq. 1 defines the unguided diffusion dynamics, optimizing the ELBO in Eq. 2 amounts to learning a controlled process that steers these dynamics toward states consistent with $\boldsymbol{y}$. To optimize the objective in eq. (2), we factorize the variational distribution as

$$q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) = q(\mathbf{x}_T \mid \boldsymbol{y}) \prod_{t=1}^{T} q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \boldsymbol{y}). \tag{3}$$

We now need to choose a parameterization for each $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \boldsymbol{y})$. We find that simple choices such as unimodal Gaussians are not sufficient: For example, in inverse problems like inpainting where $\boldsymbol{y}$ denotes an image with missing pixels, multiple plausible completions may exist for $\mathbf{x}_{t-1}$ given $\mathbf{x}_t$ and $\boldsymbol{y}$, inducing strong multimodality. This also holds for larger denoising step sizes which are known to induce more complex and potentially multimodal conditional denoising distributions [44, 17, 5, 59]. Therefore, for an expressive variational policy, we augment the variational distribution with additional latent variables $\mathbf{u}_t$ at each time step [38],

$$q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) = \int q(\mathbf{x}_{0:T}, \mathbf{u}_{1:T} \mid \boldsymbol{y}) \, d\mathbf{u}_{1:T}. \tag{4}$$

Through the lens of optimal control, these latent variables $\mathbf{u}_{1:T}$ can be interpreted as *stochastic controls*. Owing to this hierarchical structure with marginalized latent controls, we refer to the resulting formulation as *Hierarchical Variational Policies (HVP)*, which we define as,

$$q(\mathbf{x}_{0:T}, \mathbf{u}_{1:T} \mid \boldsymbol{y}) = q(\mathbf{x}_T \mid \boldsymbol{y}) \prod_{t=1}^{T} q(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}) q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{u}_t). \tag{5}$$

The *initial noise* distribution $q(\mathbf{x}_T \mid \boldsymbol{y})$ predicts the initial noise given the side information. The *per-step policy* $q(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y})$ outputs the controls at time t conditioned on the current state, side information, and past controls; in early experiments, we found that this conditioning improves training stability and performance (see App. C). Next, we define the *conditional state transition* as,

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{u}_t) = p(\mathbf{x}_{t-1} \mid \mathbf{x}_t + \gamma \mathbf{u}_t), \tag{6}$$

where $p(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is just the denoising distribution of the generative model, and γ is a tuning parameter. Thus, the controls $\mathbf{u}_t$ act as additive corrections on the states $\mathbf{x}_t$ that the diffusion model operates on [33]. This simple construction leaves us with only learning the policy and initial noise distributions. Next, we derive a tractable objective for learning such policies.

2.2 Policy Learning via Hierarchical Variational Bounds

In its naive form, variational inference becomes intractable when the variational distribution itself is defined as an integral over latent variables [4, 56]. In this case, the entropy of the resulting mixture distribution cannot be computed in closed form. Fortunately, advances in variational inference over the past decade provide a tractable alternative. Inspired by prior work [38], we adopt a less commonly used formulation of variational inference that derives a tractable objective by introducing a lower bound on the entropy term. Due to the length of the derivation, we present only the final result here; full details are in App. A. The hierarchical ELBO decomposes into four interpretable terms: a reward term and three regularizers that ensure samples are likely under the unguided diffusion model,

$$\begin{aligned}
\log p(\boldsymbol{y}) \geq \mathbb{E}_{\mathbf{x}_0 \sim q}\big[\log p(\boldsymbol{y} \mid \mathbf{x}_0)\big] &- \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T \mid \boldsymbol{y}) \parallel p(\mathbf{x}_T))}_{\mathcal{L}_1} \\
&- \sum_t \mathbb{E}_{\mathbf{x}_t, \mathbf{u}_t}\big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{u}_t) \parallel p(\mathbf{x}_{t-1} \mid \mathbf{x}_t))}_{\mathcal{L}_2}\big] \\
&- \sum_t \mathbb{E}_{\mathbf{x}_t, \mathbf{u}_{>t}}\big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}) \parallel \bar{r}(\mathbf{u}_t))}_{\mathcal{L}_3}\big].
\end{aligned} \tag{7}$$

The regularizer $\bar{r}(\mathbf{u}_t)$ over the controls is a design decision. We choose a standard Gaussian distribution $\bar{r}(\mathbf{u}_t) = \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$. From a controls perspective, the first term in eq. (7) encourages the guided process q to generate samples that maximize the likelihood $p(\boldsymbol{y} \mid \mathbf{x}_0)$. In applications

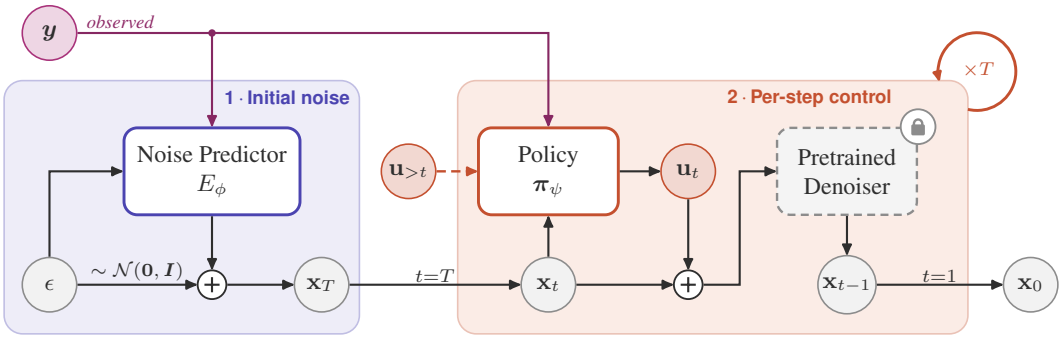


Figure 3: **AHVP samples in a single rollout, without expensive test-time optimization.** (1) The Stage-1 network E_ϕ predicts the initial noise $\mathbf{x}_T$. (2) For $t = T, \ldots, 1$, the Stage-2 policy $\boldsymbol{\pi}_\psi$ samples a control $\mathbf{u}_t$ that perturbs $\mathbf{x}_t$, steering the trajectory toward samples consistent with $\boldsymbol{y}$.

where the explicit form of the likelihood is not available, we can instead choose to maximize the expected reward $r(\mathbf{x}_0, \boldsymbol{y})$, which is a measure of the alignment between the generated sample $\mathbf{x}_0$ and the conditioning signal $\boldsymbol{y}$ (for instance, human preference scores given a text prompt and a generated sample [52, 51, 28]). The second term in Eq. 7 regularizes the noise policy with the unconditional noise prior $p(\mathbf{x}_T)$. The third term in Eq. 7 encourages the state transition distribution to stay *close* to the unguided dynamics of the pretrained iterative refinement model, implying high likelihood of the generated samples under the pretrained model. This regularizer is commonly used in RL-based finetuning of diffusion models [47, 46] and has shown to be useful in preventing *reward hacking*, i.e., overfitting to the reward without generating plausible samples. The last term in Eq. 7 encourages the per-step policy $q(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y})$ to be regularized by the prior over the controls $\bar{r}(\mathbf{u}_t)$. Next, we discuss parameterizations of different distributions in Eq. 7.

2.3 Amortized Hierarchical Variational Policies (AHVP)

While the distributions defined in eq. (7) can be determined in a test-time scaling setup [33, 23], this is computationally expensive during inference. Instead, we propose to amortize the noise policy $q(\mathbf{x}_T \mid \boldsymbol{y})$ and the per-step policy $q(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y})$ using parametric estimators. A key advantage of amortization is that during inference, sampling requires a simple rollout through the guided process $q(\mathbf{x}_{0:T}, \mathbf{u}_{1:T} \mid \boldsymbol{y})$, therefore significantly improving runtime over test-time scaling approaches. We choose the following straightforward yet expressive parameterizations of these distributions:

For the **initial noise**, we implicitly model the distribution via a call to a DiT model E_ϕ [35]:

$$\mathbf{x}_T = \epsilon + E_\phi(\boldsymbol{y}, \epsilon), \quad \epsilon \sim \mathcal{N}(0, \boldsymbol{I}_d). \tag{8}$$

This is an expressive policy, and we follow [14] and approximate the corresponding regularizer $\mathcal{L}_1 \approx \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)}[\|E_\phi(\boldsymbol{y}, \epsilon)\|_2^2]$. We model the **per-step policy** based on the pretrained diffusion model, adding the controls $\mathbf{u}_t$ to the input of the pretrained denoising network $\boldsymbol{\mu}_{\text{pretrained}}$,

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{u}_t, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_{\text{pretrained}}(\mathbf{x}_t + \gamma \mathbf{u}_t, t), \sigma_{t-1}^2 \boldsymbol{I}_d). \tag{9}$$

For expressivity, the **latent control variates** follow a distribution. We choose a Gaussian,

$$q(\mathbf{u}_t | \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\pi}_\psi(\mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}, t), \bar{\sigma}_t^2 \boldsymbol{I}_d). \tag{10}$$

Similar to the noise predictor, we model $\boldsymbol{\pi}_\psi(.)$ using a lightweight DiT architecture. Since it acts on the input space to the pretrained model, it leverages the powerful representations of the pretrained network. With this setup, the remaining loss terms evaluate to $\mathcal{L}_2 \propto \|\boldsymbol{\mu}(\mathbf{x}_t + \gamma \mathbf{u}_t, t) - \boldsymbol{\mu}(\mathbf{x}_t, t)\|_2^2$ and $\mathcal{L}_3 \propto \|\boldsymbol{\pi}_\psi(\mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}, t)\|_2^2 = \|\mathbf{u}_t\|_2^2$. While the standard deviation $\bar{\sigma}_t$ could be learned, we set it heuristically to $\bar{\sigma}_t = \kappa \sigma_t$, where κ is a scalar hyperparameter, and σ_t is the standard deviation of the reverse denoising process at time t such that the noise level monotonically decreases in the denoising process. The intuition is that the controls encode more uncertainty during earlier denoising stages as compared to the end.

Together, this fully determines the loss function. Practically speaking, we first train the initial noise policy network E_ϕ by backpropagating through the diffusion chain. We then optimize the

per-step policy $\boldsymbol{\pi}_\psi$ over all steps in parallel by replacing the reward term with the Tweedie estimate $\log p(\mathbf{y} \mid \hat{\mathbf{x}}_0)$. This avoids backpropagation through the diffusion chain. Experimentally, we find that learning the initial noise can recover high-level structure, while learning the per-step policies recovers more granular details.

Semi-Amortized Hierarchical Variational Policies (SHVP). Training neural networks for amortization will result in a finite amortization error due to finite model capacity and training data. This amortization error can be reduced by spending a small amount of additional optimization, offering a simple way to trade off quality and speed. We further tune the controls with a few gradient steps:

$$\mathbf{u}_t^* = \arg\max_{\mathbf{u}_t} \mathbb{E}_q \Big[\log p(\boldsymbol{y} \mid \hat{\mathbf{x}}_0) - \lambda_2 \|\boldsymbol{\mu}(\mathbf{x}_t + \gamma \mathbf{u}_t, t) - \boldsymbol{\mu}(\mathbf{x}_t, t)\|_2^2 - \lambda_3 \|\mathbf{u}_t\|_2^2 \Big]. \tag{11}$$

This loss follows from reducing eq. (7) to the controls of the diffusion step under consideration [33]. Overall, our approach follows the core idea behind semi-amortized variational inference [26, 32]. Therefore, we denote this method as semi-amortized hierarchical variational policies (SHVP).

3 Experiments

While our framework can be applied to any differentiable reward, we demonstrate its empirical benefits on a suite of challenging *inverse problems*. Across both quantitative and qualitative evaluations, our approach achieves a superior perception–runtime tradeoff compared to recent state-of-the-art test-time scaling methods. In particular, the fully amortized sampler (AHVP) provides high-quality reconstructions with significantly reduced inference time, while the semi-amortized variant (SHVP) consistently achieves the best perceptual quality across tasks. We further investigate several key design choices underlying our framework. Through ablation studies, we analyze the effect of the policy parameterization and the impact of the proposed two-stage training procedure. We defer implementation details to App. B.

Reward Specification. For inverse problems, we define the likelihood as $p(\boldsymbol{y} \mid \mathbf{x}_0) = \mathcal{N}(\mathcal{A}(\mathbf{x}_0), \sigma_y^2 \mathbf{I}_d)$, recovering $\mathbf{x}_0 \sim p_{\text{data}}$ from degradations $\boldsymbol{y} = \mathcal{A}(\mathbf{x}_0) + \sigma_y \boldsymbol{z}$, $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Following [7, 33], we evaluate on linear tasks ($4\times$ and $8\times$ super-resolution; 90% random inpainting) and a non-linear task (HDR with oversaturation factor 2.0), fixing $\sigma_y = 0.01$ across all tasks. See App. B for the exact form of $\mathcal{A}$.

Models, Datasets, and Training. We experiment on FFHQ [24] and ImageNet [10] at 256×256, using publicly available unconditional pretrained checkpoints [7, 12]. Both the noise predictor and per-step policy use the DiT architecture, trained on 1k (FFHQ) and 10k (ImageNet) images per task. Evaluation uses a held-out set of 100 samples per dataset. Full hyperparameters are in App. B.

Baselines. We compare AHVP and SHVP against DAPS [55], NDTM [33], RED-Diff [31], C-ΠGDM [34], MPGD [19], DDRM [25], DPS [7], HyperNoise [14] (a noise-only variant of Eq. 7), and DEFT [11]. To assess test-time scaling under limited budgets, we reduce sampling steps by 5–6$\times$ (minimum 8) and tune all baselines for best low-step performance.

Evaluation Metrics. We report perceptual metrics (LPIPS [57], FID [20]) in the main text and reconstruction metrics (PSNR, SSIM) in App. C.

3.1 Main Results

Table 1 presents quantitative comparisons across four challenging inverse problems. Across all tasks and datasets, our methods consistently achieve strong perceptual quality. *Overall, AHVP provides the best quality–runtime tradeoff, while SHVP consistently achieves the highest perceptual quality.*

Perceptual Quality Comparisons. AHVP outperforms most test-time scaling baselines while maintaining competitive reconstruction quality. For example, on SR$\times$4, AHVP achieves better perceptual quality than all test-time scaling baselines on ImageNet while remaining competitive on FFHQ. Similarly, for random inpainting, AHVP achieves competitive performance on FFHQ and state-of-the-art LPIPS on ImageNet. Comparable trends are observed for the HDR task, demonstrating that AHVP achieves a favorable perception–distortion tradeoff. To further study the role of amortization for initializing test-time scaling methods, we evaluate semi-amortized hierarchical policies (SHVP). Across all four tasks, SHVP achieves the best perceptual quality in most settings, establishing new

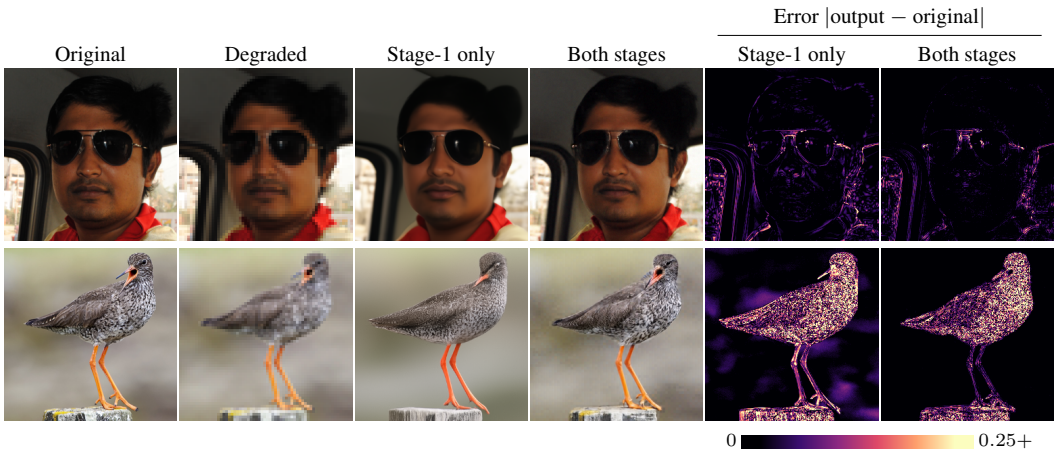


Figure 4: **Per-step controls recover fine details missed by the initial noise.** AHVP on SR×4 with only the Stage-1 noise predictor vs. both stages. Error maps show |output − original| on a shared scale: Stage-1 recovers coarse structure, and Stage-2 per-step controls further reduce the error. Top: FFHQ-256; bottom: ImageNet-256.

state-of-the-art results on several benchmarks. These results validate our hypothesis that pretrained amortized policies can serve as strong initialization for more powerful test-time scaling strategies. Overall, AHVP provides the best quality–runtime tradeoff, while SHVP consistently achieves the highest perceptual quality across tasks.

Runtime Comparisons. In addition to perceptual quality, it is important to evaluate the perception–runtime tradeoff. Table 1 reports the runtime of all competing baselines on both FFHQ and ImageNet. AHVP is significantly faster than most test-time scaling baselines while maintaining strong reconstruction quality. For instance, on the SR×4 task, AHVP is more than $5\times$ faster than NDTM while achieving similar or better perceptual quality. This tradeoff is further illustrated in Fig. 2. Although methods such as RED-Diff and DDRM provide fast inference, they produce substantially worse perceptual quality. Conversely, approaches such as DAPS and NDTM achieve strong quality but require significantly more computation. In contrast, AHVP achieves competitive perceptual quality with extremely efficient inference, yielding a superior overall perception–runtime tradeoff. This efficiency arises from amortization, which enables guided inference through a single pass of the controlled process q (Eq. 4) without expensive inner-loop optimization or gradient computations. Moreover, SHVP achieves higher perceptual quality than strong baselines such as NDTM at comparable inference cost, demonstrating that amortized policies provide an effective foundation for constructing efficient test-time samplers.

3.2 Ablation Results

Impact of Policy Type. We study the effect of modeling the per-step control policy as a stochastic distribution. In our method, the policy is parameterized as a unimodal Gaussian (Eq. 10). We compare against a deterministic policy baseline, $q(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}) = \delta(\mathbf{u}_t = \boldsymbol{\pi}_\psi(\mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}, t))$. Table 2 reports the perceptual quality of stochastic and deterministic policies on the SR×4 task for FFHQ and ImageNet. On FFHQ, both variants achieve comparable performance. However, on ImageNet the stochastic policy substantially improves perceptual quality, reducing LPIPS from 0.328 to 0.244 and improving FID from 117.66 to 84.44. These results suggest that stochastic policies provide a more flexible control mechanism for complex datasets. Exploring richer policy distributions, such as normalizing-flow-based policies, is an interesting direction for future work.

Impact of Two-Stage Training. Table 3 evaluates Stage-2 controls on top of the Stage-1 noise policy for SR×4. Stage-2 reduces the LPIPS of AHVP by nearly 50% and consistently improves SHVP. Fig. 4 shows that per-step controls recover sharp edges and textures missed by the initial noise policy. Thus, Stage-1 provides a strong trajectory initialization, while Stage-2 supplies fine-grained refinements along the rollout.

Table 1: **Quantitative comparison on diffusion-based inverse problems.** Perceptual quality (LPIPS, FID) and per-sample runtime on a single A6000 GPU; reconstruction metrics (PSNR/SSIM) are in Table 14. SHVP achieves the best perceptual quality across most tasks, while AHVP offers a strong quality–speed tradeoff. **Bold**/underline: best/second-best per metric; –: unstable runs.

Task	Method	FFHQ-256			ImageNet-256		
		Time (s) ↓	LPIPS ↓	FID ↓	Time (s) ↓	LPIPS ↓	FID ↓
SR (×4)	DAPS [55]	18.9	0.174	50.45	30.9	0.249	80.35
	NDTM [33]	4.1	0.069	41.22	5.8	0.264	84.15
	RED-Diff [31]	0.3	0.229	89.67	0.7	0.392	128.79
	DDRM [25]	0.3	0.136	68.02	0.7	0.335	93.99
	DPS [7]	10.4	0.138	72.18	46.7	0.350	130.09
	C-ΠGDM [34]	0.7	0.111	63.27	2.4	0.274	81.44
	MPGD [19]	0.6	0.229	64.59	2.0	0.323	95.48
	HyperNoise [14]	0.6	0.147	79.91	1.3	0.431	155.14
	DEFT [11]	**0.2**	0.148	63.58	**0.5**	0.337	95.76
	(Ours) AHVP	0.6	0.073	45.17	1.3	0.244	84.44
	(Ours) SHVP	4.1	**0.059**	**38.50**	5.8	**0.202**	**65.54**
SR (×8)	DAPS	18.9	0.284	91.01	30.9	0.421	182.04
	NDTM	4.1	0.145	69.86	5.8	**0.349**	**120.85**
	RED-Diff	0.3	0.363	122.74	0.7	0.566	214.96
	DDRM	0.3	0.212	89.83	0.7	0.516	155.31
	DPS	10.4	0.192	81.60	46.7	0.401	154.69
	C-ΠGDM	0.7	0.164	75.99	2.4	0.422	145.52
	MPGD	0.6	0.395	100.89	2.0	0.546	210.43
	HyperNoise	0.6	0.236	101.72	1.3	0.606	212.57
	DEFT	**0.2**	0.251	87.81	**0.5**	0.462	167.46
	(Ours) AHVP	0.6	0.219	88.33	1.3	0.514	192.07
	(Ours) SHVP	4.1	**0.134**	**62.48**	5.8	0.445	135.66
Random Inpainting (90%)	DAPS	18.9	0.266	84.43	30.9	0.397	222.05
	NDTM	4.1	0.107	67.91	5.8	0.296	132.33
	RED-Diff	0.3	0.827	305.02	0.7	0.929	327.46
	DDRM	0.3	0.743	244.08	0.7	0.893	346.67
	DPS	10.4	0.122	69.09	46.7	0.289	116.67
	C-ΠGDM	0.7	–	–	2.4	–	–
	MPGD	0.6	0.354	118.72	2.0	0.607	265.72
	HyperNoise	0.6	0.239	104.60	1.3	0.484	188.07
	DEFT	**0.2**	0.405	176.05	**0.5**	0.549	258.42
	(Ours) AHVP	0.6	0.113	80.21	1.3	**0.239**	111.27
	(Ours) SHVP	4.1	**0.065**	**44.53**	5.8	0.309	**100.55**
HDR	DAPS	18.9	0.145	**36.22**	30.9	0.186	47.95
	NDTM	4.1	0.093	49.51	5.8	0.142	55.08
	RED-Diff	0.3	0.138	72.01	0.7	0.204	84.50
	DPS	10.4	–	–	46.7	–	–
	HyperNoise	0.6	0.193	102.90	1.3	0.468	196.98
	DEFT	**0.2**	0.084	41.00	**0.5**	0.142	**44.78**
	(Ours) AHVP	0.6	0.089	45.76	1.3	0.138	61.01
	(Ours) SHVP	4.1	**0.073**	43.35	5.8	**0.135**	50.63

Policy Transfer. We test whether learned policies transfer across degradations on FFHQ, both within a degradation family (SR×4 ↔ SR×8) and across degradations (SR×4 ↔ HDR). Policies transfer within a family: applied to SR×8, an SR×4 policy attains an LPIPS of 0.243, close to the 0.223 of a policy trained on SR×8 (Table 4). This is intuitive, since similar tasks share inductive biases. However, they do not transfer across degradations with different inductive biases (SR×4 → HDR: 0.204 vs. 0.089; Table 5), which would require retraining. Instead, AHVP policies can be trained in a multi-task manner. As an illustration, we condition the AHVP controller on a task embedding, which adds negligible parameters, and train it jointly on SR×4 and HDR. It achieves competitive performance on both tasks compared to single-task policies (Table 5), showing that AHVP can amortize guidance across multiple tasks.

Table 2: **Effect of stochastic policies.** Impact of the policy distribution on SR×4 perceptual quality (LPIPS/FID).

Method	FFHQ-256		ImageNet-256	
	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓
AHVP (Deterministic)	0.081	**43.69**	0.328	117.66
AHVP (Stochastic)	**0.073**	45.17	**0.244**	**84.44**

Table 3: **Per-step controls (Stage-2) improve reconstruction quality.** Ablation on training stages for AHVP and SHVP on SR×4 for FFHQ-256 and ImageNet-256.

Method	Stage-1	Stage-2	FFHQ-256		ImageNet-256	
			LPIPS ↓	FID ↓	LPIPS ↓	FID ↓
AHVP	✓	✗	0.147	79.91	0.432	155.14
AHVP	✓	✓	**0.073**	**45.17**	**0.244**	**84.44**
SHVP	✓	✗	0.066	40.95	0.225	66.42
SHVP	✓	✓	**0.059**	**38.50**	**0.203**	**65.54**

Table 4: **Same-degradation transfer.** LPIPS↓ on FFHQ-256 of policies trained on one task (rows) and evaluated on another (columns).

	Evaluated on	
Trained on	SR×4	SR×8
SR×4	**0.073**	0.243
SR×8	0.114	**0.219**

Table 5: **Cross-degradation transfer and multitask training.** Same protocol as Table 4; the last row is a single task-conditioned controller trained jointly on both tasks.

	Evaluated on	
Trained on	SR×4	HDR
SR×4	**0.073**	0.204
HDR	0.189	**0.089**
SR×4 + HDR (joint)	<u>0.094</u>	<u>0.097</u>

Sensitivity to κ. Next, we assess the impact of the policy width κ (Eq. 10) on reconstruction quality. We retrain AHVP on SR×4 (FFHQ) with policy width $\kappa \in \{0.01, 0.05, 0.1, 0.2\}$ and use the same κ at inference. We find that the model is robust to variation in κ during training. Similarly, when trained with a fixed $\kappa = 0.05$, the model tolerates inference-time κ, (Table 6) but excess stochasticity in the policy starts to hurt at $\kappa = 0.2$ (LPIPS 0.091).

Cost of Amortization. Next we assess the cost of amortization. While amortization requires training the controllers, but this cost is smaller than it may appear. First, controller training is cheap: on SR×4 (FFHQ), the controller reaches within about 5% of its best LPIPS in $\sim$4 h (Table 7), roughly the cost of generating (3,600 samples with NDTM), and is quickly repaid whenever the sampler is reused. Second, AHVP has few hyperparameters: the controller architecture is identical across tasks, and only the number of DiT blocks changes between datasets (4 for FFHQ, 8 for ImageNet; Table 12), whereas training-free solvers require per-task tuning of step sizes and guidance weights. Third, a single task-conditioned controller trained jointly on several tasks (Table 5) shares the training cost across them.

4 Related Work

Test-time scaling in diffusion models has recently gained traction in settings ranging from text-to-image alignment [30] to inverse problems [9, 58]. Some earlier works rely on tractable approximations of the diffusion posterior [7, 43, 34, 6, 36] based on Tweedie's estimate for guidance in diffusion models. However, these approximations are often too simplistic, require costly Jacobian–vector products, or restrict applicability to certain downstream tasks or diffusion model architectures (latent vs. pixel space), leading to poor sample quality, instability, or limited general use. Consequently, some recent work [54, 2] adds a correction term to these approximations to better satisfy the constraints. In contrast, optimization-based methods [19, 39, 33, 23, 55] embed an inner optimization loop within the reverse diffusion process to estimate the diffusion posterior at each timestep. However, this can be expensive during inference as it may require backpropagation through the entire diffusion chain [15, 50]. Our method instead sidesteps such approximations and optimization loops and allows sample generation via a single rollout of the diffusion process.

Another line of work in test-time guidance focuses on non-differentiable rewards/constraints. For instance, SCG [22] leverages ideas from path integral control for symbolic music generation. Singhal et al. [41] propose steering diffusion models at test-time by sampling a collection of particles and

Table 6: **Sensitivity to κ** (policy width) on SR×4 (FFHQ-256). LPIPS lower is better.

Inference κ	0.01	0.05	0.1	0.2
Train $\kappa =$ inference κ	0.083	0.084	0.079	0.077
Train $\kappa = 0.05$	0.085	0.084	0.079	0.091

Table 7: **Controller training cost vs. quality** on SR×4 (FFHQ-256). LPIPS lower is better.

Training steps	1	100	500	1000	1700
LPIPS ↓	0.150	0.094	0.102	0.080	0.076
Time (h)	0.01	0.45	2.22	4.44	7.57

resampling them at intermediate steps based on their expected reward. Lastly, Geyfman et al. [18] propose the REINFORCE estimator for guidance via black-box rewards for Bayesian inference. In contrast, we focus only on differentiable terminal costs, and extending our framework to non-differentiable costs could be an important direction for future work.

Reward-based finetuning in diffusion models. Test-time scaling adapts the reverse diffusion process but can be computationally expensive. Inspired by reward-based finetuning in language models [60, 37], recent work explores reward-based finetuning for diffusion models [3, 16, 46]. We focus on differentiable rewards. DRaFT [8] directly optimizes such rewards while reducing cost through truncated backpropagation. Our method pursues a similar objective but includes additional regularization, improving training stability. Closely related is entropy-regularized finetuning of diffusion models [47, 45]. In particular, Uehara et al. [47] propose a two-stage framework similar to ours with two key differences. First, they infer the optimal initial noise using neural SDE solvers, which can be computationally expensive at inference time, whereas we directly predict the initial noise with a transformer. Second, our tractable objective (Eq. 7) introduces an additional regularization penalty on the per-step policy, leading to a different stochastic control formulation. Building on this objective, Domingo-Enrich et al. [13] propose a memoryless schedule for more efficient transport between base and target distributions. Another line of work [14, 49] removes amortization in latent space and derives a similar noise-prediction objective, though from a different theoretical perspective. Extending such scheduling ideas to our control formulation is an interesting future direction.

5 Conclusion

We introduced a principled framework for amortizing test-time guidance in diffusion models via hierarchical variational policies. By interpreting guidance as stochastic control over denoising trajectories, our method learns expressive policies that steer sampling while avoiding expensive test-time optimization. Empirically, the resulting fully amortized sampler (AHVP) achieves a strong perceptual quality–speed tradeoff across several challenging inverse problems, matching or exceeding recent test-time scaling baselines while requiring substantially less computation. Our semi-amortized extension (SHVP) further combines amortized proposals with lightweight test-time refinement, achieving state-of-the-art perceptual quality while maintaining efficient inference. More broadly, our results highlight amortized stochastic control as a promising framework for efficient test-time adaptation in diffusion models. Future work includes extending the framework to non-differentiable rewards, exploring richer hierarchical policies, and applying amortized guidance to broader generative tasks such as text-to-image alignment and video generation.

Acknowledgments and Disclosure of Funding

Stephan Mandt acknowledges funding from the National Science Foundation (NSF) through an NSF CAREER Award IIS-2047418, IIS-2611549, the NSF LEAP Center, the Chan Zuckerberg Initiative, SAP, and the Hasso Plattner Research Center at UCI.

References

[1] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

[2] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Roni Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

[3] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

[4] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017.

[5] Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions on Machine Learning Research*, 2025.

[6] Benjamin Boys, Mark Girolami, Jakiw Pidstrigach, Sebastian Reich, Alan Mosca, and Omer Deniz Akyildiz. Tweedie moment projected diffusions for inverse problems. *Transactions on Machine Learning Research*, 2024. Featured Certification.

[7] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2022.

[8] Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2024.

[9] Giannis Daras, Hyungjin Chung, Chieh-Hsin Lai, Yuki Mitsufuji, Jong Chul Ye, Peyman Milanfar, Alexandros G. Dimakis, and Mauricio Delbracio. A survey on diffusion models for inverse problems, 2024.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[11] Alexander Denker, Francisco Vargas, Shreyas Padhy, Kieran Didi, Simon V Mathis, Riccardo Barbano, Vincent Dutordoir, Emile Mathieu, Urszula Julia Komorowska, and Pietro Lio. DEFT: Efficient fine-tuning of diffusion models by learning the generalised h-transform. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[12] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[13] Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025.

[14] Luca Eyring, Shyamgopal Karthik, Alexey Dosovitskiy, Nataniel Ruiz, and Zeynep Akata. Noise hypernetworks: Amortizing test-time compute in diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[15] Luca Eyring, Shyamgopal Karthik, Karsten Roth, Alexey Dosovitskiy, and Zeynep Akata. Reno: Enhancing one-step text-to-image models through reward-based noise optimization. *Advances in Neural Information Processing Systems*, 37:125487–125519, 2024.

[16] Ying Fan and Kangwook Lee. Optimizing ddpm sampling with shortcut fine-tuning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[17] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[18] Daniel Geyfman, Felix Draxler, Jan Groeneveld, Hyunsoo Lee, Theofanis Karaletsos, and Stephan Mandt. Calibrated test-time guidance for bayesian inference, 2026.

[19] Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, and Stefano Ermon. Manifold preserving guided diffusion. In *The Twelfth International Conference on Learning Representations*, 2024.

[20] 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. *Advances in neural information processing systems*, 30, 2017.

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[22] Yujia Huang, Adishree Ghatare, Yuanzhe Liu, Ziniu Hu, Qinsheng Zhang, Chandramouli Shama Sastry, Siddharth Gururani, Sageev Oore, and Yisong Yue. Symbolic music generation with non-differentiable rule guided diffusion. In *ICML*, 2024.

[23] Yazid Janati, Badr MOUFAD, Alain Oliviero Durmus, Eric Moulines, and Jimmy Olsson. Divide-and-conquer posterior sampling for denoising diffusion priors. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[24] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[25] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in Neural Information Processing Systems*, 35:23593–23606, 2022.

[26] Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2678–2687. PMLR, 2018.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[28] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in neural information processing systems*, 36:36652–36663, 2023.

[29] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[30] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps, 2025.

[31] Morteza Mardani, Jiaming Song, Jan Kautz, and Arash Vahdat. A variational perspective on solving inverse problems with diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

[32] Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pages 3403–3412. PMLR, 2018.

[33] Kushagra Pandey, Farrin Marouf Sofian, Felix Draxler, Theofanis Karaletsos, and Stephan Mandt. Variational control for guidance in diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.

[34] Kushagra Pandey, Ruihan Yang, and Stephan Mandt. Fast samplers for inverse problems in iterative refinement models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[35] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, October 2023.

[36] Ashwini Pokle, Matthew J. Muckley, Ricky T. Q. Chen, and Brian Karrer. Training-free linear image inverses via flows. *Transactions on Machine Learning Research*, 2024.

[37] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[38] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 324–333, New York, New York, USA, 20–22 Jun 2016. PMLR.

[39] Litu Rout, Yujia Chen, Nataniel Ruiz, Abhishek Kumar, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. RB-modulation: Training-free stylization using reference-based modulation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[40] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

[41] Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.

[42] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[43] Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023.

[44] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 32211–32252, 2023.

[45] Wenpin Tang and Fuzhong Zhou. Fine-tuning of diffusion models via stochastic control: entropy regularization and beyond. *arXiv preprint arXiv:2403.06279*, 2024.

[46] Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review, 2024.

[47] Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *arXiv preprint arXiv:2402.15194*, 2024.

[48] Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xiner Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review, 2025.

[49] Siddarth Venkatraman, Mohsin Hasan, Minsu Kim, Luca Scimeca, Marcin Sendera, Yoshua Bengio, Glen Berseth, and Nikolay Malkin. Outsourced diffusion sampling: Efficient posterior inference in latent spaces of generative models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 61212–61239. PMLR, 13–19 Jul 2025.

[50] Hengkang Wang, Xu Zhang, Taihui Li, Yuxiang Wan, Tiancong Chen, and Ju Sun. DMPlug: A plug-in method for solving inverse problems with diffusion models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[51] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

[52] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

[53] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6613–6623, 2024.

[54] Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 23174–23184, October 2023.

[55] Bingliang Zhang, Wenda Chu, Julius Berner, Chenlin Meng, Anima Anandkumar, and Yang Song. Improving diffusion inverse problem solving with decoupled noise annealing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 20895–20905, 2025.

[56] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

[57] 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.

[58] Hongkai Zheng, Wenda Chu, Bingliang Zhang, Zihui Wu, Austin Wang, Berthy Feng, Caifeng Zou, Yu Sun, Nikola Borislavov Kovachki, Zachary E Ross, Katherine Bouman, and Yisong Yue. Inversebench: Benchmarking plug-and-play diffusion priors for inverse problems in physical sciences. In *The Thirteenth International Conference on Learning Representations*, 2025.

[59] Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. In *Forty-second International Conference on Machine Learning*, 2025.

[60] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Hierarchical Variational Policies

Here we present the proof of the bound in Eq. 7 in the main text.

Proof overview. We derive a variational lower bound (ELBO) on $\log p(\boldsymbol{y})$ by introducing a structured variational distribution $q(\mathbf{x}_{0:T} \mid \boldsymbol{y})$ augmented with auxiliary control variables $\{\mathbf{u}_t\}$. The proof proceeds in three steps:

1. We write the standard ELBO, decomposing it into an expected log-joint (energy) term and an entropy term.
2. We lower-bound the entropy by introducing a factored approximate posterior $\bar{r}(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y})$ over the controls, exploiting the non-negativity of the KL divergence.
3. We substitute the factored parameterizations of both the generative and variational processes to arrive at a per-timestep bound.

Proof. Recall the generative process,

$$p(\mathbf{x}_{0:T}, \boldsymbol{y}) = p(\mathbf{x}_T) \left[\prod_{t=1}^{T} p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \right] p(\boldsymbol{y} \mid \mathbf{x}_0). \tag{12}$$

We introduce a variational distribution $q(\mathbf{x}_{0:T} \mid \boldsymbol{y})$ and apply Jensen's inequality to obtain the standard ELBO,

$$\begin{aligned} \log p(\boldsymbol{y}) &= \log \int \frac{p(\mathbf{x}_{0:T}, \boldsymbol{y})}{q(\mathbf{x}_{0:T} \mid \boldsymbol{y})} q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) \, d\mathbf{x}_{0:T} \\ &\geq \underbrace{\mathbb{E}_{q(\mathbf{x}_{0:T} \mid \boldsymbol{y})} \left[\log p(\mathbf{x}_{0:T}, \boldsymbol{y}) \right]}_{\text{Joint-Likelihood}} \underbrace{- \mathbb{E}_{q(\mathbf{x}_{0:T} \mid \boldsymbol{y})} \left[\log q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) \right]}_{\text{Entropy } (\mathcal{H}_q)}. \end{aligned} \tag{13}$$

We parameterize the variational distribution as a Markov chain over states:

$$q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) = q(\mathbf{x}_T \mid \boldsymbol{y}) \prod_t q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \boldsymbol{y}). \tag{14}$$

As in the main text, we augment this variational distribution with additional latent control variables $\{\mathbf{u}_t\}$ to make it more expressive,

$$q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) = \int q(\mathbf{x}_{0:T}, \mathbf{u}_{1:T} \mid \boldsymbol{y})\, d\mathbf{u}_{1:T}, \tag{15}$$

where we define the augmented joint as,

$$q(\mathbf{x}_{0:T}, \mathbf{u}_{1:T} \mid \boldsymbol{y}) = q(\mathbf{x}_T \mid \boldsymbol{y}) \prod_t q(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}) \prod_t q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{u}_t, \boldsymbol{y}). \tag{16}$$

Here $q(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y})$ denotes the *variational policy* that selects a control conditioned on the current state $\mathbf{x}_t$, the history of controls $\mathbf{u}_{>t}$, and the guidance signal $\boldsymbol{y}$, while $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{u}_t, \boldsymbol{y})$ is the controlled state transition. The joint-likelihood term in Eq. equation 13 maximizes the likelihood of the generative process in Eq. equation 12, while the entropy term encourages exploration in the variational distribution. We simplify both terms below, starting with the entropy.

Simplifying the entropy. We lower-bound $\mathcal{H}_q$ by introducing an approximate posterior $r(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y})$ over the controls and exploiting $D_{\mathrm{KL}}(\cdot \,\|\, \cdot) \geq 0$. Let $q(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y})$ denote the exact posterior over controls implied by Eqs. equation 15–equation 16. Then,

$$\begin{aligned}
\mathcal{H}_q &= -\mathbb{E}_{q(\mathbf{x}_{0:T}|\boldsymbol{y})}[\log q(\mathbf{x}_{0:T} \mid \boldsymbol{y})] \\
&= -\mathbb{E}_{q(\mathbf{x}_{0:T}|\boldsymbol{y})}[\log q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) + D_{\mathrm{KL}}(q(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y}) \,\|\, q(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y}))] \\
&\geq -\mathbb{E}_{q(\mathbf{x}_{0:T}|\boldsymbol{y})}[\log q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) + D_{\mathrm{KL}}(q(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y}) \,\|\, \bar{r}(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y}))]\,,
\end{aligned} \tag{17}$$

where the bound becomes exact when $\bar{r}$ matches q exactly. This can be further simplified as,

$$\mathcal{H}_q \geq -\mathbb{E}_{q(\mathbf{x}_{0:T}|\boldsymbol{y})}[\log q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) + D_{\mathrm{KL}}(q(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y}) \,\|\, \bar{r}(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y}))]\,, \tag{18}$$

$$= -\mathbb{E}_{q(\mathbf{x}_{0:T}|\boldsymbol{y})}\left[\log q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) + \mathbb{E}_{q(\mathbf{u}_{1:T}|\mathbf{x}_{0:T}, \boldsymbol{y})} \log \frac{q(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y})}{\bar{r}(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y})}\right] \tag{19}$$

$$= -\mathbb{E}_{q(\mathbf{x}_{0:T}, \mathbf{u}_{1:T}|\boldsymbol{y})}\left[\log q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) + \log \frac{q(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y})}{\bar{r}(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y})}\right] \tag{20}$$

$$= -\mathbb{E}_{q(\mathbf{x}_{0:T}, \mathbf{u}_{1:T}|\boldsymbol{y})}[\log q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) + \log q(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y}) - \log \bar{r}(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y})] \tag{21}$$

Expanding the KL and using $\log q(\mathbf{x}_{0:T}, \mathbf{u}_{1:T} \mid \boldsymbol{y}) = \log q(\mathbf{x}_{0:T} \mid \boldsymbol{y}) + \log q(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y})$, this simplifies to,

$$\mathcal{H}_q \geq -\mathbb{E}_{q(\mathbf{x}_{0:T}, \mathbf{u}_{1:T}|\boldsymbol{y})}[\log q(\mathbf{x}_{0:T}, \mathbf{u}_{1:T} \mid \boldsymbol{y}) - \log \bar{r}(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y})]\,. \tag{22}$$

We factorize the approximate posterior as a product of per-timestep terms,

$$\bar{r}(\mathbf{u}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{y}) = \prod_t \bar{r}(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}). \tag{23}$$

Substituting the parameterizations from Eqs. equation 16 and equation 23 into Eq. equation 22, and writing $\mathbb{E}_q$ as shorthand for $\mathbb{E}_{q(\mathbf{x}_{0:T}, \mathbf{u}_{1:T}|\boldsymbol{y})}$, we obtain,

$$\begin{aligned}
\mathcal{H}_q \geq &-\mathbb{E}_q\Big[\log q(\mathbf{x}_T \mid \boldsymbol{y})\Big] - \sum_t \mathbb{E}_q\Big[\log q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{u}_t, \boldsymbol{y})\Big] \\
&- \sum_t \mathbb{E}_{\mathbf{x}_t, \mathbf{u}_{>t}}\Big[D_{\mathrm{KL}}(q(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}) \,\|\, \bar{r}(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}))\Big],
\end{aligned} \tag{24}$$

where $\mathbb{E}_{\mathbf{x}_t, \mathbf{u}_{>t}}$ denotes the expectation obtained by rolling out a trajectory from $\mathbf{x}_T$ to $\mathbf{x}_t$, collecting all controls $\mathbf{u}_{>t}$ along the way.

Simplifying the joint likelihood. Since $p(\mathbf{x}_{0:T}, \boldsymbol{y})$ does not depend on $\mathbf{u}_{1:T}$, the energy term from Eq. equation 13 can be equivalently written under the augmented joint,

$$\mathbb{E}_{q(\mathbf{x}_{0:T}|\boldsymbol{y})}[\log p(\mathbf{x}_{0:T}, \boldsymbol{y})] = \mathbb{E}_q\left[\log p(\boldsymbol{y} \mid \mathbf{x}_0) + \log p(\mathbf{x}_T) + \sum_t \log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\right]. \tag{25}$$

Combining. Substituting Eqs. equation 24 and equation 25 into the ELBO (Eq. equation 13), we arrive at,

$$\begin{aligned}\mathcal{L} \geq \mathbb{E}_q\big[\log p(\boldsymbol{y} \mid \mathbf{x}_0)\big] - D_{\mathrm{KL}}(q(\mathbf{x}_T \mid \boldsymbol{y}) \,\|\, p(\mathbf{x}_T)) \\ - \sum_t \mathbb{E}_{\mathbf{x}_t,\mathbf{u}_t}\Big[D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{u}_t, \boldsymbol{y}) \,\|\, p(\mathbf{x}_{t-1} \mid \mathbf{x}_t))\Big] \\ - \sum_t \mathbb{E}_{\mathbf{x}_t,\mathbf{u}_{>t}}\big[D_{\mathrm{KL}}(q(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}) \,\|\, \bar{r}(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}))\big]. \end{aligned} \tag{26}$$

Lastly, we model the approximate posterior over controls as independent standard Gaussians,

$$\bar{r}(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}) = \bar{r}(\mathbf{u}_t) = \mathcal{N}(0, \boldsymbol{I}_d), \tag{27}$$

which gives the final bound,

$$\begin{aligned}\mathcal{L} \geq \mathbb{E}_q\big[\log p(\boldsymbol{y} \mid \mathbf{x}_0)\big] - D_{\mathrm{KL}}(q(\mathbf{x}_T \mid \boldsymbol{y}) \,\|\, p(\mathbf{x}_T)) \\ - \sum_t \mathbb{E}_{\mathbf{x}_t,\mathbf{u}_t}\Big[D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{u}_t, \boldsymbol{y}) \,\|\, p(\mathbf{x}_{t-1} \mid \mathbf{x}_t))\Big] \\ - \sum_t \mathbb{E}_{\mathbf{x}_t,\mathbf{u}_{>t}}\big[D_{\mathrm{KL}}(q(\mathbf{u}_t \mid \mathbf{x}_t, \mathbf{u}_{>t}, \boldsymbol{y}) \,\|\, r(\mathbf{u}_t))\big]. \end{aligned} \tag{28}$$

This concludes the proof. □

B Implementation Details

B.1 Reward Specification

Here we describe the form of the likelihood $p(\boldsymbol{y} \mid \mathbf{x}_0)$ in more detail. For inverse problems, the likelihood can be specified as,

$$p(\boldsymbol{y}|\mathbf{x}_0) = \mathcal{N}(\mathcal{A}(\mathbf{x}_0), \sigma_y^2 \boldsymbol{I}_d) \tag{29}$$

Super-Resolution (SR): Following [7], the degradation operator for super-resolution can be specified as

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{y}|\boldsymbol{S}^f \mathbf{x}, \sigma_y^2 \boldsymbol{I}), \tag{30}$$

where $\boldsymbol{S}^f$ represents the bicubic downsampling matrix with downsampling factor f. In this work, we choose $f = \{4, 8\}$ for both datasets.

Random Inpainting (90%). We use random inpainting with a dropout probability of 0.9 (or 90%). For this task, the forward model can be specified as

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{y}|\boldsymbol{M}\boldsymbol{x}, \sigma_y^2 \boldsymbol{I}_d) \tag{31}$$

where $\boldsymbol{M} \in \{0, 1\}^{d \times d}$ is the masking matrix.

High Dynamic Range (HDR). We use the HDR setup from [7]. More specifically, the HDR degradation operator is given by

$$\boldsymbol{y} = \texttt{clip}(\alpha \mathbf{x}_0 + \beta, 0, 1.0) \tag{32}$$

Depending on the choice of the scalar coefficients α and β, the resulting degradation can be oversaturated or undersaturated. Following prior work [55], we choose $\alpha = 2.0$ and $\beta = 0$.

B.2 Baseline Hyperparameters

We optimize all baselines for the best perceptual sample quality. We use the official code implementation from `https://github.com/czi-ai/oc-guidance`. Below, we highlight the key hyperparameters for different baselines.

Table 8: Hyperparameters of C-ΠGDM used for different super-resolution tasks.

	FFHQ			ImageNet		
Task	λ	w	τ	α	w	τ
SR ($\times 4$)	-0.3	4.0	0.4	-0.4	4.0	0.4
SR ($\times 8$)	-0.2	4.0	0.4	-0.5	4.0	0.4

Table 9: RED-Diff hyperparameters used for different tasks.

	FFHQ		ImageNet	
Task	lr	λ	lr	λ
SR ($\times 4$)	0.2	0.1	0.2	0.1
SR ($\times 8$)	0.1	0.1	0.2	0.1
Inpainting	0.2	0.1	0.2	0.1
HDR	0.5	0.1	0.2	0.5

Note on the number of reverse diffusion sampling steps. As mentioned in the main text, we mainly focus on conditional sampling under low inference budgets and therefore reduce the sampling budget of all competing baselines by a factor of 5–6$\times$, with a minimum number of reverse diffusion steps set to 8. We elaborate more below.

DPS [7]. Across all tasks and datasets, we use DPS with the DDIM sampler and $\eta = 0.5$. We reduce the number of reverse diffusion steps in DPS from 1000 [7] to 200. We use the adaptive step size proposed in [7] and set it to

$$\zeta^{'} = \frac{\zeta}{\|\boldsymbol{y} - \mathcal{A}(\hat{\mathbf{x}}_0)\|_2^2} \tag{33}$$

where we fix $\zeta = 1.0$.

DDRM [25]. Across all tasks and datasets, following [25], we set $\eta_b = 0.85$ and $\eta = 1.0$, which we found robust across tasks. We reduce the sampling budget of DDRM from 20, as proposed in [25], to 10. DDRM can only be applied to linear inverse problems.

C-ΠGDM [34]. For all tasks and datasets, we fix the number of diffusion steps to 10 using the noise conditioned C-ΠGDM sampler. Here w and τ represent the hyperparameters for projection to a conjugate space while τ represents the start-time for reverse diffusion sampling. We find that C-ΠGDM is unstable for random inpainting. A similar observation was made by [33]. C-ΠGDM only works for linear inverse problems.

DAPS [55]. For all tasks and datasets, we use the DAPS-200 configuration from [55] without modification. The number of reverse diffusion steps is set to 2, with 100 MCMC steps between each step.

MPGD [19]. For all tasks and datasets, we reduce the number of steps for MPGD from 100 [19] to 20. We set DDIM $\eta = 0.5$ and the scale parameter to 20.0, which we found to work best across tasks.

RED-Diff [31]. For all tasks and datasets, we reduce the number of steps for RED-Diff to 10. We use the DDIM sampler with $\eta = 0.0$. We highlight other important hyperparameters, such as the learning rate and gradient-term weight, in Table 9.

NDTM. We reduce the number of reverse diffusion sampling steps in NDTM from 50 [33] to 8. Moreover, we fix the weighting of different loss components in NDTM to a terminal cost weight of $w_T = 50$, score regularizer, and control weights set to `ddim`. Refer to Eq. 17 in [33] for more details

Table 10: NDTM hyperparameters used for different tasks.

	FFHQ					ImageNet				
Task	N	γ	η	τ	lr	N	γ	η	τ	lr
SR ($\times 4$)	5	1.0	0.0	400	0.05	2	2.0	0.1	600	0.05
SR ($\times 8$)	5	1.0	0.0	500	0.05	2	1.0	0.0	600	0.07
Inpainting	2	6.0	0.0	500	0.05	2	1.0	0.0	600	0.1
HDR	5	1.0	0.2	400	0.05	5	1.0	0.0	400	0.05

Table 11: Loss weighting for Stage-1 training.

	FFHQ		ImageNet	
Task	w_T	w_{control}	w_T	w_{control}
SR ($\times 4$)	50.0	1.0	50.0	1.0
SR ($\times 8$)	50.0	1.0	50.0	1.0
Inpainting	50.0	1.0	50.0	1.0
HDR	10.0	1.0	1.0	1.0

Table 12: Architecture details of the latent and per-step controllers.

	Latent Controller		Per-step Controller	
	FFHQ	ImageNet	FFHQ	ImageNet
In-Channels	6	6	9	9
Out-Channels	3	3	3	3
Embed Dim.	768	768	768	768
Num Heads	8	8	8	8
Patch Size	4	4	4	4
Num DiT Blocks	4	8	4	8
Dropout	0.0	0.0	0.0	0.0
MLP Ratio	4.0	4.0	4.0	4.0
Parameters	51M	92M	51M	92M

Table 13: Loss weighting for Stage-2 training.

	FFHQ			ImageNet		
Task	w_T	w_{control}	w_{score}	w_T	w_{control}	w_{score}
SR ($\times 4$)	50.0	`ddim`	`ddim`	50	`ddim`	`ddim`
SR ($\times 8$)	50.0	`ddim`	`ddim`	50.0	`ddim`	`ddim`
Inpainting	25.0	`ddim`	`ddim`	25.0	`ddim`	`ddim`
HDR	1.0	`ddim`	`ddim`	1.0	`ddim`	`ddim`

on the specific form of the loss weighting. We tune NDTM for best performance with all task-specific hyperparameters in Table 10.

B.3 Training Hyperparameters

For modeling the noise predictor and the per-step controller, we use the DiT [35] architecture. See Table 12 for more details. During training, we fix the learning rate to 1e-4 for both stages (noise prediction and per-step controls) and use the Adam [27] optimizer. We highlight the loss weights, i.e., the terminal loss (w_T) and the control loss (w_{control}), for different tasks for Stage-1 training in Table 11. For Stage-2 training, we mostly adopt the weights of terminal loss (w_T), control loss (w_{control}), and score loss (w_{score}) from [33] and specify them in Table 13. During training, we set $\kappa = 0.05$ as the multiplicative factor for the standard deviation of the stochastic controller (see Eq. 10 in the main text).

Table 14: Reconstruction metrics for different inverse problems. -: unstable runs.

Task	Method	FFHQ-256		ImageNet-256	
		PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
SR (×4)	DAPS-200 [55]	**30.19**	0.794	**25.30**	0.671
	NDTM [33]	29.73	**0.849**	22.88	0.676
	RED-Diff [31]	26.43	0.720	23.04	0.672
	DDRM [25]	29.49	0.834	<u>23.94</u>	<u>0.692</u>
	DPS [7]	26.89	0.748	22.06	0.587
	C-ΠGDM [34]	28.10	0.765	23.30	0.630
	MPGD [19]	27.16	0.791	23.62	0.656
	HyperNoise [14]	27.86	0.780	21.60	0.571
	DEFT [11]	28.96	0.815	23.77	**0.710**
	(Ours) AHVP	29.33	0.836	23.38	0.665
	(Ours) SHVP	<u>29.90</u>	<u>0.842</u>	23.67	0.648
SR (×8)	DAPS-200	**26.32**	0.691	**22.29**	0.523
	NDTM	25.35	<u>0.708</u>	20.35	0.514
	RED-Diff	25.16	0.696	20.59	0.513
	DDRM	<u>25.78</u>	**0.726**	<u>21.27</u>	**0.550**
	DPS	23.36	0.628	19.69	0.467
	C-ΠGDM	25.09	0.674	20.90	0.487
	MPGD	21.01	0.561	18.35	0.392
	HyperNoise	23.60	0.649	19.22	0.457
	DEFT	24.83	0.669	20.74	0.522
	(Ours) AHVP	25.09	0.697	20.75	0.371
	(Ours) SHVP	24.95	0.677	21.16	<u>0.526</u>
Random Inpainting (90%)	DAPS-200	25.72	0.707	<u>21.62</u>	0.539
	NDTM	<u>27.21</u>	<u>0.822</u>	20.55	**0.614**
	RED-Diff	13.24	0.218	13.20	0.197
	DDRM	15.41	0.263	14.61	0.187
	DPS	26.94	0.776	**21.74**	<u>0.608</u>
	C-ΠGDM	-	-	-	-
	MPGD	21.34	0.632	14.97	0.268
	HyperNoise	23.26	0.648	19.55	0.466
	DEFT	16.20	0.421	15.35	0.310
	(Ours) AHVP	26.32	0.788	20.74	0.598
	(Ours) SHVP	**28.56**	**0.835**	18.96	0.531
HDR	DAPS-200	**27.15**	0.869	**24.65**	0.813
	NDTM	25.42	0.874	22.40	<u>0.828</u>
	RED-Diff	24.80	0.803	20.57	0.689
	DPS	-	-	-	-
	HyperNoise	22.92	0.774	19.33	0.521
	DEFT	24.72	**0.906**	19.76	**0.842**
	(Ours) AHVP	24.30	0.893	21.93	0.799
	(Ours) SHVP	<u>25.52</u>	<u>0.902</u>	<u>22.65</u>	0.804

C Extended Results

C.1 Reconstruction metrics.

We provide PSNR and SSIM comparisons between different baselines in Table 14. Our method performs comparably to most baselines across different tasks.

C.2 Extended Qualitative Results.

We provide additional qualitative comparisons between our proposed methods (AHVP, SHVP) and competing baselines in Figs. 5, 6, and 7.

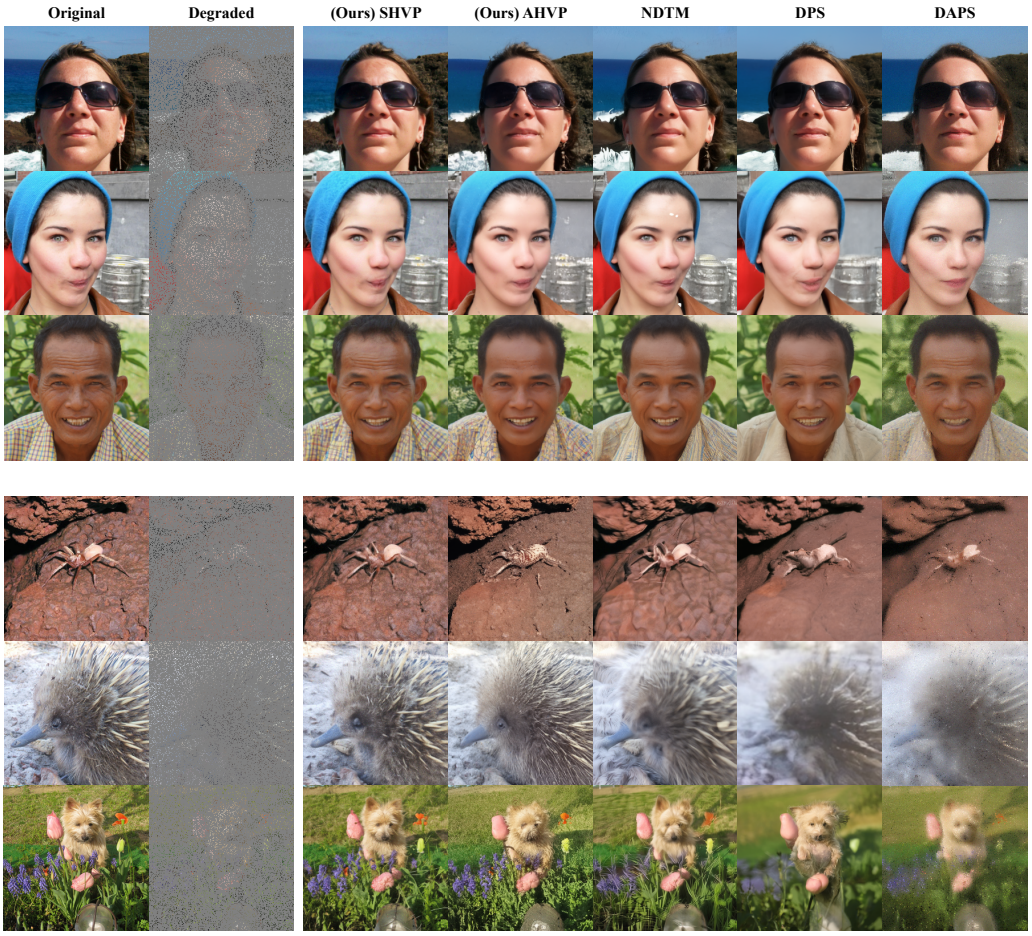


Figure 5: **Qualitative comparison on 90% random inpainting.** Top panel: FFHQ-256; bottom panel: ImageNet-256. Our methods (AHVP, SHVP) capture fine-grained details better than competing baselines.

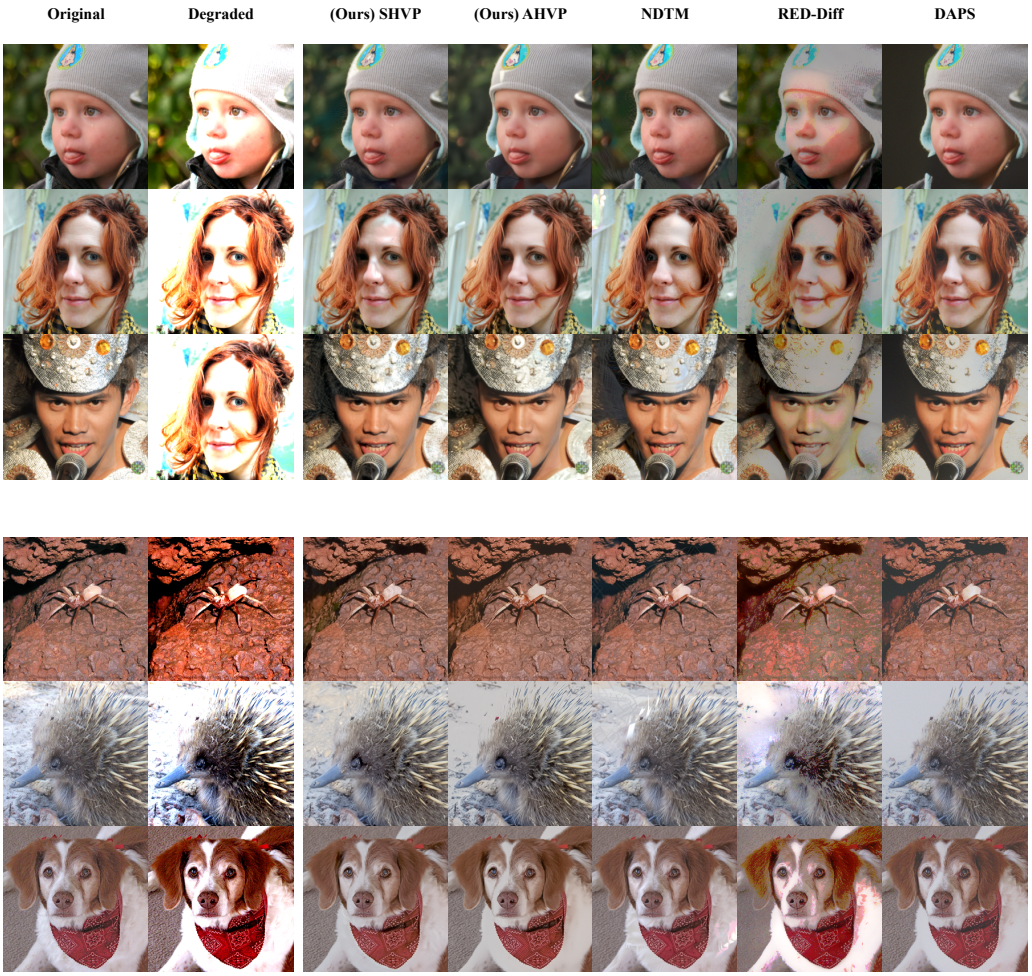


Figure 6: **Qualitative comparison on HDR.** Top panel: FFHQ-256; bottom panel: ImageNet-256. Our methods (AHVP, SHVP) preserve the overall color profile better than competing baselines.

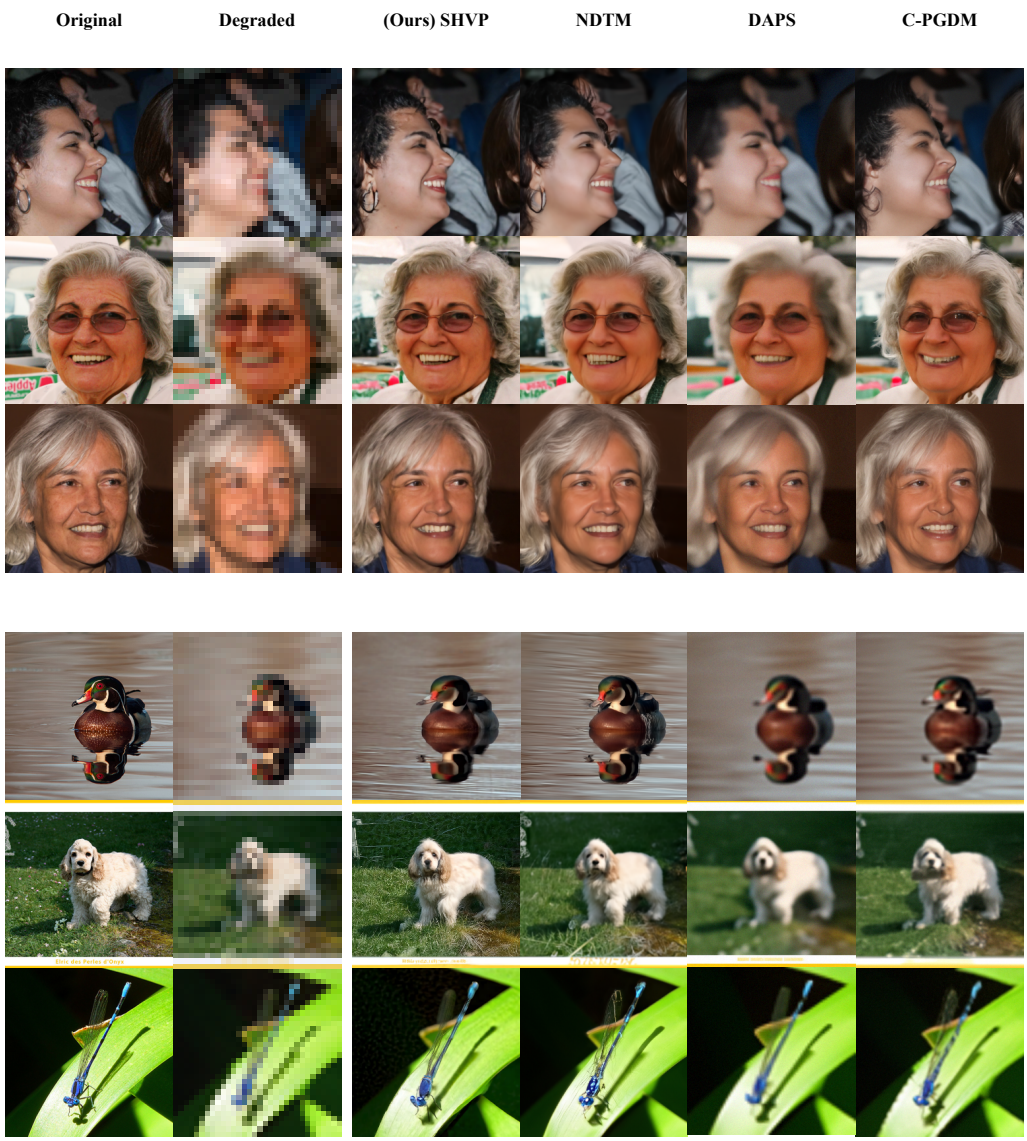


Figure 7: **Qualitative comparison on SR$\times$8.** Top panel: FFHQ-256; bottom panel: ImageNet-256. SHVP (ours) captures details better than competing baselines.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claim in the abstract and introduction accurately reflects the contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations and future work in the conclusion

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Yes, the paper provides the full set of assumptions and a proof in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All the details for training and inference are discussed in the Appendix and the main text.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will provide access to our code upon acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all these details in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report error bars as different runs are expensive to repeat

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide all the details in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes we conform with the NeurIPS code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We acknowledge the code resources used wherever needed

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not release any new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not involve crowdsourcing or working with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The paper did not use any LLMs for core methodological development.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.